FIREWALL MONITORING USING INTRUSION DETECTION SYSTEMS

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

MASTER OF SCIENCE

in Computer Software

by
Şükran ASARCIKLI

October 2005
İzmir

We approve the thesis of **Şükran ASARCIKLI**

Date of Signature

……………………………………

17.10.2005

Assist. Prof. Dr. Tuğkan TUĞLULAR
Supervisor
Department of Computer Engineering
İzmir Institute of Technology

……………………………………

17.10.2005

Assoc. Prof. Dr. Ahmet KOLTUKSUZ
Department of Computer Engineering
İzmir Institute of Technology

……………………………………

17.10.2005

Prof. Dr. Şaban EREN
Department of Computer Engineering
Ege University

……………………………………

17.10.2005

Prof. Dr. Kayhan ERCİYEŞ
Head of Department
İzmir Institute of Technology

....

Assoc. Prof. Semahat ÖZDEMİR
Head of the Graduate School

ABSTRACT

Most organizations have intranet, they know the benefits of connecting their private LAN to the Internet. However, Internet is inherently an insecure network. That makes the security of the computer systems an imported problem. The first step of network security is firewalls. Firewalls are used to protect internal networks from external attacks through restricting network access according to the rules. The firewall must apply previously defined rules to each packet reaching to its network interface. If the application of rules are prohibited due to malfunction or hacking, internal network may be open to attacks and this situation should be recovered as fast as possible. In order to be sure about the firewall working properly, we proposed to use Intrusion Detection Systems (IDS) to monitor firewall operation. The architecture of our experimental environment is composed of a firewall and two IDSs. One IDS is between external network and firewall, while the other is between firewall and private network. Those two IDSs are invisible to the both networks and they send their information to a monitoring server, which decides, based on two observations, whether the firewall is working properly or not.

ÖZET

Günümüzde birçok kurum kendi yerel ağına sahip ve yerel ağını Internet'e bağlamanın avantajlarının da farkında. Bunun yanında Internet güveni olmayan bir ortam. Bu durum bilgisayar sistemlerinin güvenliğini önemli bir problem olarak karşımıza çıkarmaktadır. Ateş duvarları ağ güvenliğinde ilk adımdır, yerel ağları Internet'ten gelecek saldırılara karşı korumak için kullanılırlar. Bunu ağ trafiğini önceden belirlenmiş kurallara göre kısıtlayarak gerçekleştirirler. Ateş duvarları kendisine ulaşan her ağ paketine önceden tanımlanmış olan kuralları uygulamalıdır. Herhangi bir nedenden dolayı kuralların uygulanması engellenirse, yerel ağ saldırılara açık olacaktır. Bu durum en kısa sürede düzeltilmelidir. Bu çalışmada, ateş duvarlarının doğru çalışıp çalışmadığından emin olmak için Nüfuz Tespit Sistemleri'nin kullanılması önerilmektedir. Önerilen yapı bir ateş duvarı ve iki Nüfuz Tespit Sisteminden oluşmaktadır. Nüfuz Tespit Sistemlerinden biri Internet ile ateş duvarı arasındaki ağ trafiğini izlerken, diğeri de ateş duvarı ile yerel ağ arasındaki ağ trafiğini izlemektedir. Her iki Nüfuz Tespit Sistemi de ağlar tarafından görülmemektedir. Nüfuz Tespit Sistemleri izledikleri ağ trafiğinin denetim sonuçlarını bir denetim sunucusuna gönderirler. Sunucu her iki Nüfuz Tespit Sisteminden gelen denetim sonuçlarına göre ateş duvarının doğru çalışıp çalışmadığına karar verir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The improvements in modern technology have enabled the use of computer systems in conducting business and in gathering and sharing information in corporations and academic institutions using the Internet. Today it is possible to make many banking operations and shopping via Internet. Many companies have been presented on the Internet. The data that is shared on the Internet and the company should be saved from attacks.

The primary focus when TCP/IP was first developed was to ensure reliable communications between groups of networks. At that time, security was not an issue because the size of this Internet was small and most of the users knew each other. The base technologies used to construct this network contained many insecurities, most of which continue to exist today. Due to a number of well-reported attacks on private networks originating from the Internet, security is now a primary concern when an organization connects to the Internet. Organizations need to conduct their business in a secure manner and to protect their data and computing resources from attack. Such needs are heightened as businesses link geographically distant parts of the organization using private networks based on TCP/IP.

Nowadays, securing private networks is very important. An organization implementing a secure network must first develop a network security policy that specifies the organization’s security requirements for their Internet connection. A network security policy specifies what connections are allowed between the private and external networks and the actions to take in the event of a security breach. A firewall placed between the private network and the Internet, enforces the security policy by controlling what connections can be established between the two networks. All network traffic must pass through the firewall, which ensures that only permitted traffic passes. The main purpose of a firewall system is to control access to or from a protected network.

Using only a firewall concentrates security in one point. It should not be forgotten firewall itself also is open to attacks. Any problem with the operation of the firewall, due to malfunction or hacking could be disastrous to other less protected systems on the internal network. This case should be detected and recovered as fast as possible.

This thesis is organized in six chapters. Chapter 2 provides background information about firewalls, their types, architectures and role in network security. In Chapter 3, intrusion detection systems (IDSs) are defined, the classification of IDSs is provided. Also it contains the information about the detection methods of IDSs and their response after detecting an intrusion. Chapter 4 defines basic rules for firewall testing and presents the needs for monitoring a firewall. Chapter 5 explains the proposed architecture to monitor firewall, a working example of the proposed architecture and installation of the experiment environment is in laboratory.

CHAPTER 2

FIREWALLS

This chapter describes what firewalls can do for network security, various types of firewalls technologies according to their protection mechanisms, their properties and firewall architectures and details of building those firewalls architectures.

2.1 What is a Firewall?

Firewalls are usually the first component of network security. They separate networks in different security levels, by utilizing network access control policies. The major function of the firewall is to protect the private network from non-legitimate traffic.

Firewalls are located between the Internet and private network. They can monitor the outgoing and incoming traffic; also they can prevent the harmful traffic and attacks from Internet. They also can stop the non-legitimate outgoing traffic. If a computer from the local network is attacked by an intruder and generates non-legitimate traffic, the firewall can prevent and detect the computer. Firewall can detect such succeeded attack, so it can be recovered.

A firewall is the most effective way to connect a network to the Internet and still protect that network (Chapman and Zwicky 1995). Firewalls create a separation between public networks (Internet) and private networks by examining the traffic according to the predefined policy, and allowing only legitimate traffic to pass between the public and private network. They help implementing a larger security policy that defines the services and access to be permitted. It is an implementation of that policy in terms of a network configuration, one or more host systems and routers, and other security measures such as advanced authentication in place of static passwords.

A firewall system can be a router, a personal computer, a host, or a collection of hosts and/or routers, set up specifically to shield a site or subnet from protocols and services that can be abused from hosts outside the subnet (Wack and Carnahan 1995).

Firewalls must be installed at the choke points to control network traffic and implement network security policy of the organization. Firewalls achieve this by examining the all incoming and outgoing network traffic according to the predefined firewall policy. All network traffic must pass through the firewall, which ensures that only permitted traffic are allowed through (WEB_1 2005). Firewalls have some advantage and disadvantages they are summarized below.

Advantages:

- Firewalls can stop non-legitimate traffic at first point,
- Firewalls can filter protocols and services that are either not necessary or that cannot be adequately secured from exploitation (NIST 2002),
- A firewall can "hide" names of internal systems and internal network schema, thereby revealing less information to outside hosts (NIST 2002),
- Firewalls can concentrate extended logging of network traffic on one system.

Disadvantages:

- Firewalls utilize manually configured set of rules to differentiate legitimate traffic from non-legitimate traffic,
- Once a static policy is defined, the firewall can't react to a network attack – nor can it initiate effective counter-measures (NIST 2002),
- Firewalls only examine network packets that pass through them, do not examine network traffic between any two inside hosts,
- Most firewalls do not analyze the contents of the data packets that make up network traffic,
- Firewall policies can vary in effectiveness, depending on the expertise of the security manager and the complexity of the network environment.

2.2 Definitions About Firewalls

These are very basic definitions related with firewalls.

Firewall: A component or set of components that restricts access between a protected network and the Internet, or between other sets of networks, according to firewall policy.

Host: A computer system attached to a network.

Bastion host: A computer system that must be highly secured because it is vulnerable to attack, usually because it is exposed to the Internet and is a main point of contact for users of internal networks.

Dual-homed host: A general-purpose computer system that has at least two network interfaces (or homes)

Packet: The fundamental unit of communication on the Internet.

Packet filtering: The action a device takes to selectively control the flow of data to and from a network. Packet filters allow or block packets, usually while routing them from one network to another. To accomplish packet filtering, you set up a set of rules that specify what types of packets are to be allowed and what types are to be blocked. Packet filtering may occur in a router, in a bridge, or on an individual host. It is sometimes known as screening (Chapman and Zwicky 1995).

Perimeter network: A network added between a protected network and an external network, in order to provide an additional layer of security. A perimeter network is sometimes called a DMZ, which stands for De-Militarized Zone.

Proxy server: A program that deals with external servers on behalf of internal clients. Proxy clients talk to proxy servers, which relay approved client requests on to real servers, and relay answers back to clients.

2.3 Firewalls in Network Security

Firewalls can control all incoming and outgoing traffic because they are placed at the choke point between the public network and private network. Firewalls act like traffic cops and perform the critical task of filtering traffic crossing the network boundary according to the predefined security rules (firewall policy).These rules can be specified at the network layer and/or at the application layer. Firewalls utilize these static, manually configured, security policies to differentiate legitimate traffic from non-legitimate traffic.

Typical reasons for using a firewall to protect a private network include the following (Shay 2000):

- To prevent unauthorized external users from accessing computing resources on the internal network. This is necessary because it is extremely difficult and costly to attempt to secure all the hosts within a private network,

- To control internal user access to the external network to prevent the export of proprietary information,
- To avoid the negative public relations impact of a break in,
- To provide a dependable and reliable connection to the Internet, so that employees do not implement their own insecure private connections.

Firewalls can do a lot for your site's security. In fact, some advantages of using firewalls extend even beyond security, as described below.

A firewall is a focus for security decisions: Think of a firewall as a choke point. All traffic in and out must pass through this single, narrow checkpoint. A firewall gives you an enormous amount of advantages for network security because it lets you concentrate your security measures on this choke point: the point where your network connects to the Internet.

Focusing your security in this way is far more efficient than spreading security decisions and technologies around, trying to cover all the bases in a piecemeal fashion. Although firewalls can cost tens of thousands of dollars to implement, most sites find that concentrating the most effective security hardware and software at the firewall is less expensive, more effective than other security measures, and certainly less expensive than having inadequate security.

A firewall can enforce security policy: Many of the services that people want from the Internet are inherently insecure. The firewall is the traffic cop for these services. It enforces the site's security policy, allowing only "approved" services to pass through and those only within the rules set up for them.

For example, one site might decide that only one internal system can communicate with the outside world. Still another site might decide to allow access from all systems of a certain type, or belonging to a certain group; the variations in site security policies are endless.

A firewall may be called upon to help enforce policies that are more complicated. For example, perhaps only certain systems within the firewall are allowed to transfer files to and from the Internet; by using other mechanisms to control which users have access to those systems, you can control which users have these capabilities. Depending on the technologies you choose to implement your firewall, a firewall may have a greater or lesser ability to enforce such policies.

A firewall can log Internet activity efficiently: Because all traffic passes through the firewall, the firewall provides a good place to collect information about system and network use and misuse. As a single point of access, the firewall can record what occurs between the protected network and the external network (Chapman and Zwicky 1995).

A firewall limits your exposure: Sometimes, a firewall will be used to keep one section of your site's network separate from another section. By doing this, you keep problems that affect one section from spreading through the entire network. In some cases, you will do this because one section of your network may be more trusted than another; in other cases, because one section is more sensitive than another. Whatever the reason, the existence of the firewall limits the damage that a network security problem can do to the overall network

The main purpose of a firewall system is to control access to or from a protected network. It implements a network access policy by forcing connections to pass through the firewall, where they can be examined and evaluated.

2.4 Firewalls Not Enough

Although firewalls are an important part of network security, they do not provide airtight perimeter protection, they are not a complete security solution. The average firewall is designed to deny clearly suspicious traffic – such as an attempt to Telnet to a device when corporate security policy forbids Telnet access completely – but is also designed to allow some traffic through – Web traffic to an internal Web server, for example. Certain threats are outside the control of the firewall. Some of the weaknesses of firewalls are (Chapman and Zwicky 1995):

A firewall cannot protect you against malicious insiders: A firewall might keep a system user from being able to send proprietary information out of an organization over a network connection; so would simply not have a network connection. However, that same user could copy the data onto disk, tape, or paper and carry it out of the building in his or her briefcase.

Once a back door has been installed on a server, the attacker has ensured that he will have unfettered access to that machine at any point in the future (NA 2003). That is to say, firewalls can prevent most of the malicious access attempts but they cannot stop attacks if they are not specified in their rule sets.

A firewall cannot protect you against connections that don't go through it: A firewall can effectively control the traffic that passes through it; however, there is nothing a firewall can do about traffic that doesn't pass through it. For example, if the site allows dial-in access to internal systems behind the firewall, it has absolutely no way of preventing an intruder from getting in through such a modem.

Sometimes, technically expert users or system administrators set up their own "back doors" into the network, either temporarily or permanently, because they chafe at the restrictions that the firewall places upon them and their systems. The firewall can do nothing about this. It is really a people-management problem, not a technical problem.

A firewall cannot protect against completely new threats: A firewall is designed to protect against known threats. Firewall monitors traffic according to predefined rules and decide whether to allow or prevent the connection. No firewall can automatically defend against every new threat that arises. A well-designed one may also protect against new threats. (For example, by denying any but a few trusted services, a firewall will prevent people from setting up new and insecure services.)

A firewall cannot protect against viruses: Firewalls examine certain parts of the network packets mostly for source and destination addresses and port numbers, not for the details of the data. They cannot prevent from the attacks that are in the other parts of the packets.

2.5 Network Packets, Protocols and Services

Firewall makes decisions based on the network packets content. Content of the packets depends on the network protocols. The most popular one is TCP/IP protocol suite. The function of the TCP/IP protocol stack, or suite, is the transfer of information from one network device to another. TCP/IP protocol suits layers are (CNAPC 2005):

- Application Layer (Layer 4)
- Transport Layer (Layer 3)
- Internet Layer (Layer 2)
- Network Access Layer (Layer 1)

Application Layer: The application layer supports addressing protocols and network management. It also has protocols for file transfer, e-mail, and remote login. Examples of these protocols are: *DNS* (Domain Name System), *WINS* (Windows Internet Naming

Service), *POP3* (Post Office Protocol), *SMTP* (Simple Mail Transport Protocol), *SNMP* (Simple Network Management Protocol), *FTP* (File Transfer Protocol), *TFTP* (Trivial File Transfer Protocol) and *HTTP* (Hypertext Transfer Protocol).

Transport Layer: The transport layer enables a user's device to segment several upper-layer applications for placement on the same transport layer data stream, and enables a receiving device to reassemble the upper-layer application segments (CNAPC 2005). The transport layer data stream is a logical connection between the endpoints of a network, and provides transport services from a host to a destination. This service is sometimes referred to as end-to-end service. The two protocols found at the transport layer are **TCP (Transmission Control Protocol)** and **UDP (User Datagram Protocol)**. Either of these two protocols are used by the application layer process, the choice depends on the application's transmission reliability requirements.

- *TCP (*Transmission Control *Protocol):* is a connection-oriented, reliable protocol. TCP provides flow control by using sliding windows and ensures reliability by using sequence numbers and acknowledgments. TCP re-sends anything that is not received correctly and supplies a virtual circuit between end-user applications. The advantage of TCP is that it provides guaranteed delivery of segments. The Figure 2.5.1 shows the TCP header structure.

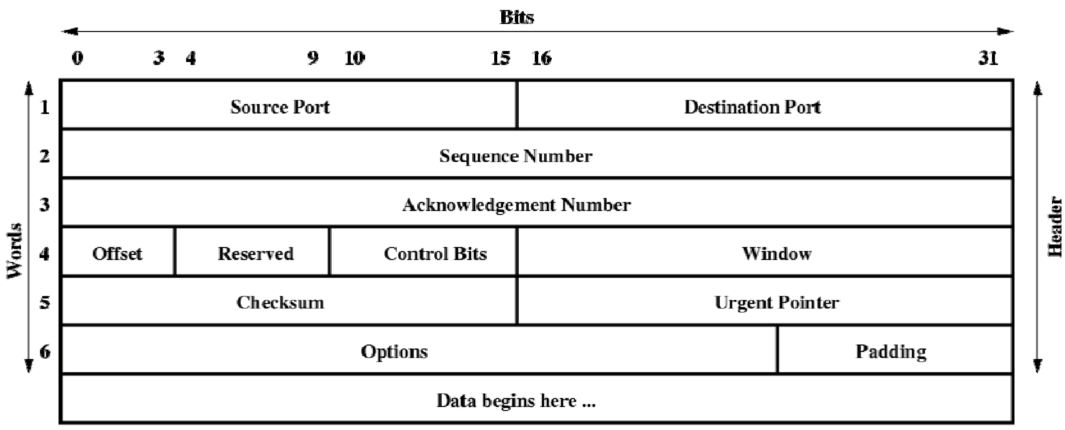

Figure 2.1. The data segment format of the TCP Protocol (CNAPC 2005)
(Source: The Cisco Certified Network Associate Curriculum)

- *UDP (*User *Datagram Protocol):* is a connectionless and unreliable transport protocol. Although UDP is responsible for transmitting messages, no software checking for segment delivery is provided at this layer. The advantage that UDP provides is speed. Since UDP provides no acknowledgments, less traffic is sent across the network, making the transfer faster. UDP is used for data streaming of audio and video where consistent speed is more important than reliable delivery. The Figure 2.5.2 shows the UDP datagram format. Protocols that use UDP include TFTP, SNMP, Network File System (NFS), and Domain Name System (DNS).

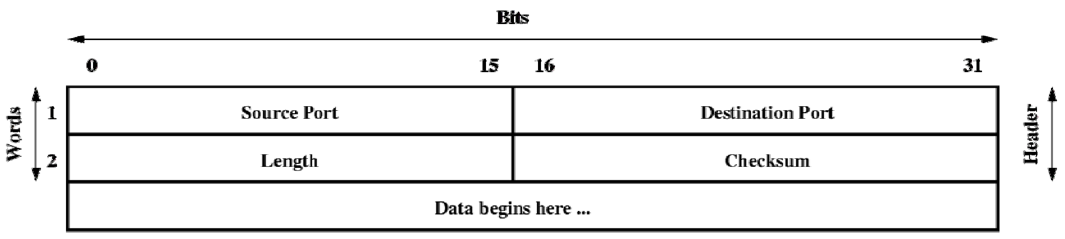

Figure 2.2. The UDP datagram format (CNAPC 2005)
(Source: The Cisco Certified Network Associate Curriculum)

Internet Layer: The Internet layer of the TCP/IP stack corresponds to the network layer of the OSI model. These layers are responsible for getting packets through a network using software addressing. Several protocols operate at the TCP/IP Internet layer that corresponds to the OSI network layer:

- *IP* -- provides connectionless, best-effort delivery routing of datagram; is not concerned with the content of the datagram; looks for a way to move the datagram to their destination
- *ICMP* -- provides control and messaging capabilities
- *ARP* -- determines the Data Link Layer (MAC) addresses for known IP addresses
- *RARP* -- determines network addresses when data link layer addresses are known

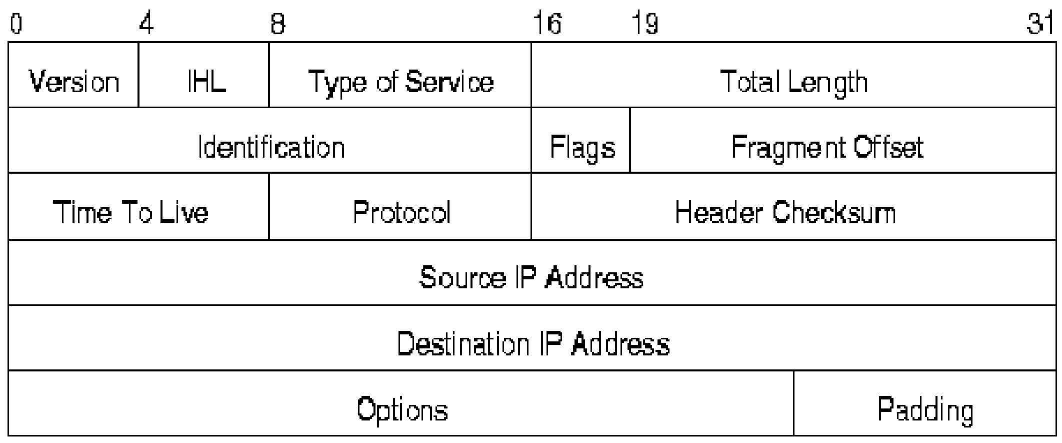

Figure 2.3. The IP header format (CNAPC 2005)
(Source: The Cisco Certified Network Associate Curriculum)

Network Access Layer: The combination of OSI datalink and physical layers deals with pure hardware (wires, satellite links, network interface cards, etc.) and access methods such as **CSMA/CD** (carrier sensed multiple access with collision detection). Ethernet exists at the network access layer - its hardware operates at the physical layer and its medium access control method (CSMA/CD) operates at the datalink layer. (WEB_3 2005).

There are number of standard Internet services that users want and that most sites try to support. HTTP for web services, SMTP for e-mail and FTP for file transfer.

HTTP (Hypertext Transfer Protocol) is the Internet standard that supports the exchange of information on the World Wide Web, as well as on internal networks. It supports many different file types including text, graphics, sound, and video. It defines the process by which Web browsers originate requests for information to send to Web servers.

SMTP (Simple Mail Transport Protocol) governs the transmission of e-mail over computer networks. It does not provide support for transmission of data other than plain text.

FTP (File Transfer Protocol) is a reliable connection-oriented service that uses TCP to transfer files between systems that support FTP. It supports bi-directional binary file and ASCII file transfers.

Every packet from the upper layer is the data for lower layer; it adds a header to packet and pass to lower layer. Firewalls are mostly dealing with the information the header contains. The content of the header includes:

- Source IP Address
- Destination IP Address
- Transport protocol
- Source and destination application port
- Eventually, specific protocol flags (e.g. TCP's ACK- and SYN-flag)
- The network interface a packet has been received on

2.6 Firewall Controls

There are four general techniques that firewalls use to control access and enforce the site's security policy. Originally, firewall focused primarily on service control, but they have since evolved to provide all four (Stalling 2002):

Service Control: Determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as a Web or mail service.

Direction Control: Determines the direction in which particular service requests may be initiated are allowed to flow through the firewall.

User Control: Controls access to a service according to which user is attempting to access it. This feature is typically applied to users inside the firewall perimeter (local users). It may also be applied to incoming traffic from external users; the latter requires some form of secure authentication technology.

Behavior Control: Controls how particular services are used. For example, the firewall may filter e-mail to eliminate spam, or it may enable external access to only a portion of the information on a local Web server.

2.7 Firewall Types

There are several types of firewall products. Firewalls are classified into three main types based on their mechanism; packet filter firewalls, stateful inspection firewalls and application proxy-gateway firewalls (NIST 2002).

2.7.1 Packet Filters

Packet filter firewalls are the most fundamental type of firewalls; they are the simplest firewalls to implement. Packet filters provide an efficient and general way to control network traffic. Packet filters provide transparent security since they work at lower layers. No changes are required to client and host applications. They filter the traffic based on packet's header content according to predefined specification criteria (NIST 2002):

- The *source address* of the packet, i.e., the Layer 3 address of the computer system or device the network packet originated from.
- The *destination address* of the packet, i.e., the Layer 3 address of the computer system or device the network packet is trying to reach.
- The *type of traffic*, that is, the specific network protocol being used to communicate between the source and destination systems or devices (often Ethernet at Layer 2 and IP at Layer 3).
- Possibly some *characteristics of the Layer 4 communications sessions*, such as the source and destination ports of the sessions (e.g., TCP:80 for the destination port be-longing to a web server, TCP:1320 for the source port belonging to a personal computer accessing the server).
- Sometimes, information pertaining *to which interface of the router* the packet came from and which interface of the router the packet is destined for; this is useful for routers with 3 or more network interfaces.

Packet filter firewalls has two strengths: speed and flexibility. Since they operate at lover layers and do not usually examine the data above Layer 3 (OSI model Network layer), they can operate very quickly. Since most networks can be accommodated using Layer 3 and below, they can be used to secure almost any type of network. Packet filter firewalls also possess several weaknesses: (NIST 2002)

- Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application-specific vulnerabilities or functions. For example, a packet filter firewall cannot block specific application commands; if a packet filter firewall allows a given application, all functions available within that application will be permitted.
- Because of the limited information available to the firewall, the logging functionality present in packet filter firewalls is limited. Packet filter logs normally contain the same information used to make access control decisions (source address, destination address, and traffic type).
- Most packet filter firewalls do not support advanced user authentication schemes. Once again, this limitation is mostly due to the lack of upper-layer functionality by the firewall.
- They are generally vulnerable to attacks and exploits that take advantage of problems within the TCP/IP specification and protocol stack, such as network layer address spoofing. Many packet filter firewalls cannot detect a network packet in which the OSI Layer 3 addressing information has been altered. Spoofing attacks are generally employed by intruders to bypass the security controls implemented in a firewall platform.
- Finally, due to the small number of variables used in access control decisions, packet filter firewalls are susceptible to security breaches caused by improper configurations. In other words, it is easy to accidentally configure a packet filter firewall to allow traffic types, sources, and destinations that should be denied based upon an organization's information security policy.

2.7.2 Stateful inspection firewalls

Stateful inspection firewalls are enhancement of the packet filter technology. Stateful inspection evolved from the need to accommodate certain features of the TCP/IP protocol suite. Besides inspecting individual packet content, the stateful inspection also inspects the attributes of the multipacket flows. They save additional information of network connections in dynamic tables called "state tables"; this "state table" is then used to validate traffic. Stateful inspection firewalls create a directory of active connections. A packet that does not belong to an active connection and is not a connection request is refused by the firewall. A packet that belongs to an active connection is allowed through, the firewall.

Stateful inspection firewalls share the strengths and weaknesses of packet filter firewalls, but due to the state table implementation, stateful inspection firewalls are generally considered to be more secure than packet filter firewalls. A stateful inspection firewall also differs from a packet filter firewall in that stateful inspection is useful or applicable only within TCP/IP network infrastructures. Stateful inspection firewalls can accommodate other network protocols in the same manner as packet filter firewalls, but the actual stateful inspection technology is relevant only to TCP/IP (NIST 2002).

2.7.3 Application Proxy-Gateways

Application Proxy-Gateways combine the lover layer access control with upper layer functionality. They are software made and operate on general purpose hardware. Each individual application-proxy, also referred to as a proxy agent, interfaces directly with the firewall access control ruleset to determine whether a given piece of network traffic should be permitted to transit the firewall (NIST 2002). In addition to the ruleset, each proxy agent has the ability to require authentication of each individual network user in different forms (User ID and Password Authentication, Hardware or Software Token Authentication, Source Address Authentication).

Application-proxy gateway firewalls have numerous advantages over packet filter firewalls and stateful inspection packet filter firewalls. They have more extensive logging capabilities due to the firewall being able to examine the entire network packet rather than just the network addresses and ports. Another advantage is that they allow security administrators to enforce whatever type of user authentication is deemed appropriate for a given enterprise infrastructure. Application-proxy gateways are capable of authenticating users directly, (packet filter firewalls and stateful inspection packet filter firewalls authenticate users based on the network layer address of the system they reside on). They are not simply Layer 3 devices; they can be made less vulnerable to address spoofing attacks.

Besides these advantages, the major disadvantage of application-proxy gateways is their high process time need. Because of the “full packet awareness” in application-proxy gateways, the firewall is forced to spend quite a bit of time reading and interpreting each packet. For this reason, they are not generally well suited to high-bandwidth or real-time applications.

2.8 Bastion Hosts

A bastion host is defined as a host that is more exposed to the hosts of an external network than the other hosts of the network it protects (WEB_2 2004)

By design, a bastion host is highly exposed, because its existence is known to the Internet. For this reason, firewall builders and managers need to concentrate security efforts on the bastion host. You should pay special attention to the host's security during initial construction and ongoing operation. Because the bastion host is the most exposed host, it also needs to be the most fortified host.

Bastion hosts are used with many different firewall approaches and architectures. Bastion hosts can be used with a firewall based on packet filtering, proxying, or a hybrid approach. The principles for building a bastion hosts are extensions of those for securing any mission critical host (WEB_2 2004):

- Keep it simple
- Prepare for the bastion host to be compromised:
- Internal hosts should not trust it any more than is absolutely required
- If possible, it should be connected in a way to the network so that it can not sniff on internal traffic
- Provide extensive logging for incident detection / analysis, if possible such that it can not be easily tampered with even when the host is compromised

2.9 Firewall Architectures

This section describes the three types of firewall architectures, variety of ways to put various firewalls components together (Chapman and Zwicky 1995).

2.9.1 Dual-Homed Host Architecture

Dual-homed *host architecture* is built around the dual-homed host computer, a computer that has at least two network interfaces. Such a host could act as a router between the networks these interfaces are attached to; it is capable of routing IP packets from one network to another. However, to implement a dual-homed host type of firewalls architecture, you disable this routing function (Chapman and Zwicky 1995). Thus, IP packets from one network (e.g., the Internet) are not directly routed to the other network (e.g., the internal, protected network). Systems inside the firewall can communicate with the dual-homed host, and systems outside the firewall can communicate with the dual-homed host, but these systems can't communicate directly with each other. IP traffic between them is completely blocked.

The network architecture for a dual-homed host firewall is simple: the dual homed host sits between, and is connected to, the Internet and the internal network. Figure 4.3 shows this architecture.

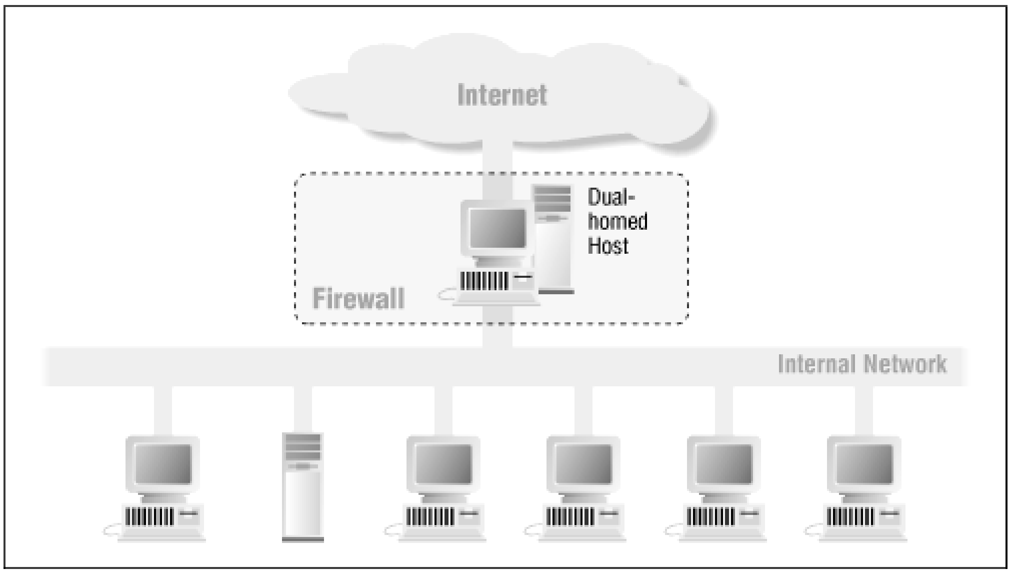

Figure 2.4. Dual-homed host architecture (Chapman and Zwicky 1995)
(Source: “Building Internet Firewalls”, (O’RIELLY), ISBN 1-56592-124-0)

Dual-homed hosts can provide a very high level of control. If you aren't allowing packets to go between external and internal networks at all, you can be sure that any packet on the internal network that has an external source is evidence of some kind of security problem. In some cases, a dual-homed host will allow you to reject connections that claim to be for a particular service but that do not actually contain the right kind of data. However, it takes considerable work to consistently take advantage of the potential advantages of dual-homed hosts.

A dual-homed host can only provide services by proxying them, or by having users log into the dual-homed host directly. *Bastion Hosts*, user accounts present significant security problems by themselves. They present special problems on dual-homed hosts, where they may unexpectedly enable services you consider insecure. Furthermore, most users find it inconvenient to use a dual-homed host by logging into it. Proxying is much less problematic, but may not be available for all services you're interested in.

2.9.2 Screened Host Architecture

Dual-homed host architecture provides services from a host that is attached to multiple networks (but has routing turned off), a screened host architecture provides services from a host that's attached to only the internal network, using a separate router (Chapman and Zwicky 1995). In this architecture, the primary security is provided by packet filtering. Figure 4.4 shows a simple version of a screened host architecture.

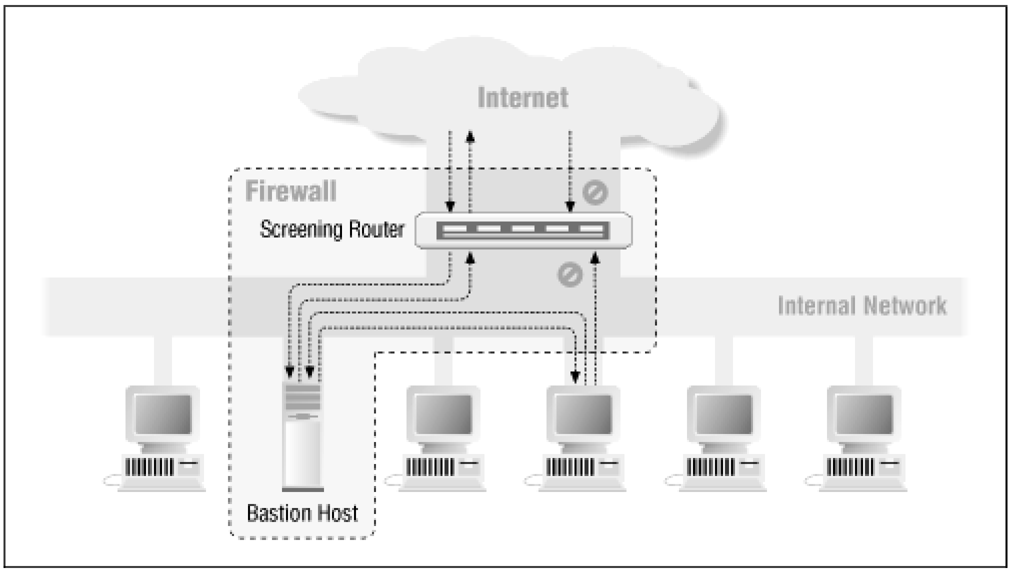

Figure 2.5. Screened host architecture (Chapman and Zwicky 1995)
(Source: “Building Internet Firewalls”, (O’RIELLY), ISBN 1-56592-124-0)

The bastion host sits on the internal network. The packet filtering on the screening router is set up in such a way that the bastion host is the only system on the internal network that hosts on the Internet can open connections to (for example, to deliver incoming email). Even then, only certain types of connections are allowed. Any external system trying to access internal systems or services will have to connect to this host. The bastion host thus needs to maintain a high level of host security.

The packet filtering also permits the bastion host to open allowable connections (what is "allowable" will be determined by your site's particular security policy) to the outside world.

The packet-filtering configuration in the screening router may do one of the following:

- Allow other internal hosts to open connections to hosts on the Internet for certain services (allowing those services via packet filtering)
- Disallow all connections from internal hosts (forcing those hosts to use proxy services via the bastion host)

These approaches can be mixed for different services; some may be allowed directly via packet filtering, while others may be allowed only indirectly via proxy. It all depends on the policy.

Because this architecture allows packets to move from the Internet to the internal networks, it may seem more risky than a dual-homed host architecture, which is designed so that no external packet can reach the internal network. In practice, however, the dual-homed host architecture is also prone to failures that let packets actually cross from the external network to the internal network. Furthermore, it's easier to defend a router, which provides a very limited set of services, than it is to defend a host. For most purposes, the screened host architecture provides both better security and better usability than the dual-homed host architecture.

Compared to other architectures, however, such as the screened subnet architecture discussed in the following section, there are some disadvantages to the screened host architecture. The major one is that if an attacker manages to break in to the bastion host, there is nothing left in the way of network security between the bastion host and the rest of the internal hosts. The router also presents a single point of failure; if the router is compromised, the entire network is available to an attacker.

2.9.3 Screened Subnet Architecture

The *screened subnet architecture* adds an extra layer of security to the screened host architecture by adding a perimeter network that further isolates the internal network from the Internet (Chapman and Zwicky 1995).

Bastion hosts are the most vulnerable machines on your network. Despite your best efforts to protect them, they are the machines most likely to be attacked, because they are the machines that *can be* attacked. If, as in a screened host architecture, your internal network is wide open to attack from your bastion host, then your bastion host is a very tempting target. There are no other defenses between it and your other internal machines. By isolating the bastion host on a perimeter network, you can reduce the impact of a break-in on the bastion host.

With the simplest type of screened subnet architecture, there are two screening routers, each connected to the perimeter net. One between the perimeter net and the internal network, the other between the perimeter net and the external network. To break into the internal network with this type of architecture, an attacker would have to get past *both* routers. Even if the attacker somehow broke in to the bastion host, he would still have to get past the interior router. There is no single vulnerable point that will compromise the internal network.

It is good idea to create a layered series of perimeter nets between the outside world and interior network. Less trusted and more vulnerable services are placed on the outer perimeter nets, farther from the interior network. The idea is that an attacker who breaks into a machine on an outer perimeter net will have a harder time successfully attacking internal machines because of the additional layers of security between the outer perimeter and the internal network. This is only true if there is actually some meaning to the different layers, however; if the filtering systems between each layer allow the same things between all layers, the additional layers do not provide any additional security.

Figure 4.5 shows a possible firewall configuration that uses the screened subnet architecture.

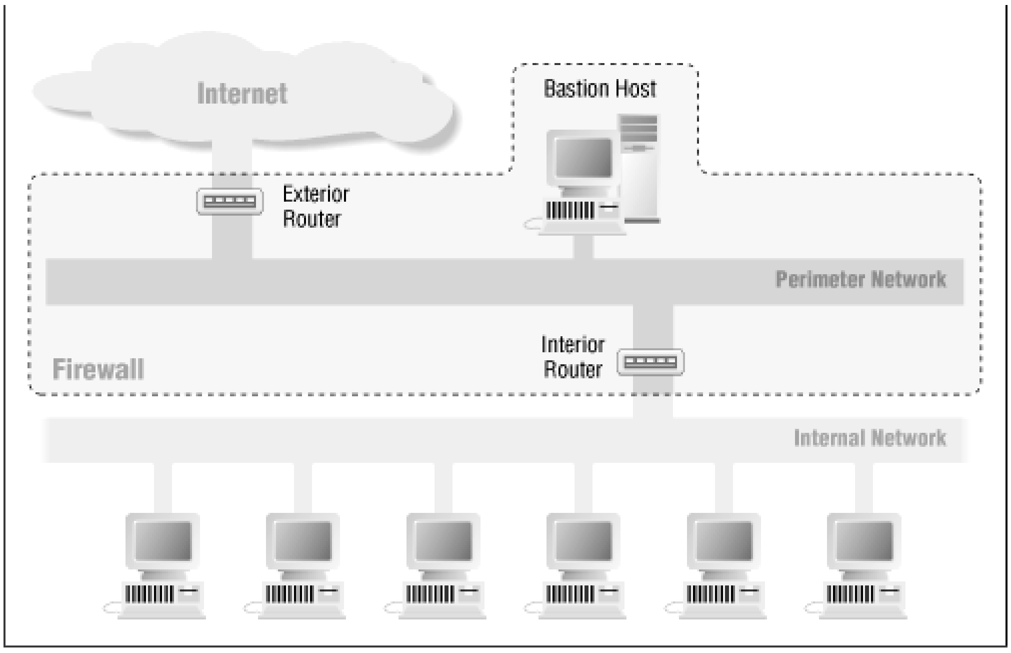

Figure 2.6. Screened subnet architecture (Chapman and Zwicky 1995)
(Source: “Building Internet Firewalls”, (O’RIELLY), ISBN 1-56592-124-0)

The components in this type of architecture include perimeter network, bastion host, interior router and exterior router (Chapman and Zwicky 1995).

Perimeter network: The perimeter network is another layer of security, an additional network between the external network and your protected internal network. If an attacker successfully breaks into the outer reaches of your firewall, the perimeter net offers an additional layer of protection between that attacker and your internal systems.

Bastion host: With the screened subnet architecture, you attach a bastion host or hosts to the perimeter net; this host is the main point of contact for incoming connections from the outside world; for example:

- For incoming email (SMTP) sessions to deliver electronic mail to the site
- For incoming FTP connections to the site's anonymous FTP server
- For incoming domain, name service (DNS) queries about the site and so on.

Outbound services (from internal clients to servers on the Internet) are handled in either of these ways:

- Set up packet filtering on both the exterior and interior routers to allow internal clients to access external servers directly.

- Set up proxy servers to run on the bastion host to allow internal clients to access external servers indirectly.

In either case, the packet filtering allows the bastion host to connect to, and accept connections from, hosts on the Internet; which hosts, and for what services, are dictated by the site's security policy.

Much of what the bastion host does is act as proxy server for various services, either by running specialized proxy server software for particular protocols (such as HTTP or FTP), or by running standard servers for self-proxying protocols (such as SMTP).

Interior router: The *interior router* (sometimes called the *choke router* in firewalls literature) protects the internal network both from the Internet *and* from the perimeter network (Chapman and Zwicky 1995). The interior router does most of the packet filtering for your firewall. It allows selected services outbound from the internal net to the Internet.

The services the interior router allows between the bastion host (on the perimeter net itself) and internal net are not necessarily the same services the interior router allows between the Internet and internal network. The reason for limiting the services between the bastion host and the internal network is to reduce the number of machines (and the number of services on those machines) that can be attacked from the bastion host, should it be compromised.

The services allowed between the bastion host and the internal network should be limited to just those that are actually needed, such as SMTP (so the bastion host can forward incoming email), DNS (so the bastion host can answer questions from internal machines, or ask them, depending on your configuration), and so on. The services should also be limited by allowing them only to or from particular internal hosts; for example, SMTP might be limited only to connections between the bastion host and internal mail server or servers.

Exterior router: In theory, the *exterior router* (sometimes called the *access router* in firewalls literature) protects both the perimeter net and the internal net from the Internet (Chapman and Zwicky 1995). In practice, exterior routers tend to allow almost anything outbound from the perimeter net, and they generally do very little packet filtering. The packet filtering rules to protect internal machines would need to be essentially the same on both the interior router and the exterior router; if there is an error in the rules that allows access to an attacker, the error will probably be present on both routers.

The only packet filtering rules that are special on the exterior router are those that protect the machines on the perimeter network (that is, the bastion hosts and the internal router). Generally, however, not much protection is necessary, because the hosts on the perimeter net are protected primarily through host security (Chapman and Zwicky 1995).

The rest of the rules that you could put on the exterior router are duplicates of the rules on the interior router. These are the rules that prevent insecure traffic from going between internal hosts and the Internet. To support proxy services, where the interior router will let the internal hosts send some protocols as long as they are talking *to* the bastion host, the exterior router could let those protocols through as long as they are coming *from* the bastion host. These rules are desirable for an extra level of security, but they are theoretically blocking only packets that cannot exist because they have already been blocked by the interior router. If they do exist, either the interior router has failed, or somebody has connected an unexpected host to the perimeter network.

One of the security tasks that the exterior router *can* usefully perform a task that usually can't easily be done anywhere else is the blocking of any incoming packets from the Internet that have forged source addresses. Such packets claim to have come from within the internal network, but actually are coming in from the Internet.

2.10 Firewall Technologies

Firewalls have additional capabilities one of them is Network Address Translation (NAT), Virtual Private Networks (VPN), Proxy.

2.10.1 Network Address Translation

NAT is the process of translating internal IP addresses to IP addresses that are visible to the external network. NAT is very often used with a special group of IP addresses (virtual IP, non Internet-routable IP address), although it works with any IP address scheme. What NAT does is basically a one-to-one or a many-to-one IP address translation. An inside (local) IP address is mapped to an outside (global) IP address, meaning that an inside IP address is replaced by the appropriate outside IP address, and vice versa (Enterasys 2001).

Network Address Translation includes the following steps: (Enterasys 2001)

- The IP address in the IP header is replaced with the new inside or outside IP address. The port numbers in the TCP/UDP header is replaced with the new port if port translation is enabled.
- The checksum for the IP packet is recalculated and checked for integrity.
- The TCP header checksum must also be recalculated since this checksum is calculated using the new inside or outside IP address, new port (if applicable) and the payload (if applicable).

NAT technology was developed in response to two major issues in network engineering and security. First, NAT is an effective tool for “hiding” the network-addressing schema present behind a firewall environment. In essence, NAT allows an organization to deploy an addressing schema of its choice behind a firewall, while still maintaining the ability to connect to external resources through the firewall. Second, the depletion of the IP address space has caused some organizations to use NAT for mapping non-routable IP addresses to a smaller set of legal addresses (NIST 2002).

NAT Pros

- Conserves the legally registered addressing scheme; enterprises and ISPs benefit since they can reduce the number of legally registered IP addresses
- Network design is simplified by now-limitless availability of addressing schemes
- Merging and changing of networks is simplified; for an enterprise it is possible to change the ISP vendor without having to completely renumber the network

NAT Cons

- Loss of end-to-end IP trace ability; the command “traceroute” will not be of great help anymore
- Applications that send IP addressing information within their data require special handling, e.g., FTP.

Overall, NAT is a great solution to the problems of limited IP addresses and having private (local) address schemes being connected to the outside (global) Internet.

2.10.2 Virtual Private Networks

Another valuable use for firewalls is the construction of Virtual Private Networks (VPNs). VPN is a connection that allows private data to be sent securely over a shared or public network, such as the Internet (Netgear 2004). A VPN is constructed on top of existing network media and protocols by using additional protocols and usually encryption. If the VPN is encrypted, it can be used as an extension of the inner, protected network. Thus, an organization or agency can send unencrypted network traffic from systems behind the firewall to other remote systems behind a cooperating VPN gateway; the firewall encrypts the traffic and forwards it to the remote VPN gateway, which decrypts it and passes it on to the destination systems. Most of the popular firewalls nowadays incorporate this type of functionality (NIST 2002).

In most cases, VPN is used to provide secure network links across networks that are not trusted. With VPN, communication links between users and sites can be achieved quickly, inexpensively, and safely across the world. In this way, VPN empower organizations to extend their network service to branch offices and remote users - such as traveling employees, telecommuters, and strategic partners - by creating a private WAN via the Internet (Netgear 2004). By using VPN technology, an organization purchases a single connection to the Internet, and that connection is used to allow remote users access into private networks and resources. This single Internet connection can also be used to provide many other types of services. As a result, this mechanism is considered to be cost-effective (NIST 2002).

2.10.3 Proxy Systems

Proxying provides Internet access to a single host, or a very small number of hosts, while appearing to provide access to all of your hosts. The hosts that have access act as proxies for the machines that do not doing what these machines want done. The proxy server evaluates requests from the client and decides which to pass on and which to disregard. If a request is approved, the proxy server talks to the real server on behalf of the client, and proceeds to relay requests from the client to the real server, and to relay the real server's answers back to the client.

There are a number of advantages to using proxy services:

Proxy services allow users to access Internet services `directly': With the dual-homed host approach, a user needs to log into the host before using any Internet services. This is often inconvenient, and some users become so frustrated that they look for ways around the firewall. With proxy services, users think they are interacting directly with Internet services.

Of course, there is more going on behind the scenes but it is usually transparent to users. While proxy services allow users to access Internet services from their own systems, they do so without allowing packets to pass directly between the user's system and the Internet. The path is indirect, either through a dual-homed host, or through a bastion host and screening router combination.

Proxy services are good at logging: Because proxy servers understand the underlying protocol, they allow logging to be performed in a particularly effective way. For example, instead of logging all of the data transferred, an FTP proxy server logs only the commands issued and the server responses received; this results in a much smaller and more useful log.

There are also some disadvantages to using proxy services.

Proxy services lag behind nonproxied services: Although proxy software is widely available for the older and simpler services like FTP and Telnet, proven software for newer or less widely used services is harder to find. There is usually a distinct lag between the introduction of a service and the availability of proxying servers for it; the length of the lag depends primarily on how well the service is designed for proxying. This makes it difficult for a site to offer new services immediately as they become available. Until suitable proxy software is available, a system that needs new services may have to be placed outside the firewall, opening up potential security holes.

Proxy services may require different servers for each service: You may need a different proxy server for each protocol, because the proxy server has to understand the protocol in order to determine what to allow and disallow, and in order to masquerade as a client to the real server and as the real server to the proxy client. Collecting, installing, and configuring all these various servers can be a lot of work.

Products and packages differ greatly in the ease with which they can be configured, but making things easier in one place can make it harder in others. For example, servers that are particularly easy to configure are usually limited in flexibility; they're easy to configure because they make certain assumptions about how they're going to be used, which may or may not be correct or appropriate for your site.

Proxy services usually require modifications to clients, procedures, or both: Except for a few services designed for proxying, proxy servers require modifications to clients and/or procedures. Either kind of modification has drawbacks; people cannot always use the readily available tools with their normal instructions.

Proxy services are not workable for some services: Proxying relies on the ability to insert the proxy server between the client and the real server; that requires relatively straightforward interaction between the two. A service like *talk* that has complicated and messy interactions may never be possible to proxy.

Proxy services do not protect you from all protocol weaknesses: As a security solution, proxying relies on the ability to determine which operations in a protocol are safe. Not all protocols provide easy ways to do this. The X Window System protocol, for example, provides a large number of unsafe operations, and it is difficult to make it work while removing the unsafe operations. HTTP is designed to operate effectively with proxy servers, but it is also designed to be readily extensible, and it achieves that goal by passing data that is going to be executed. It is impossible for a proxy server to protect you from the data; it would have to understand the data being passed and determine whether it was dangerous or not.

2.11 Selecting a Firewall

In order to pick the best architecture and packet screening method for a firewall solution, the following questions should be considered (WEB_4 2005):

- What does the firewall need to do?
- What additional services would be desirable?
- How will it fit in the existing network?
- How will it affect existing services and users?

2.11.1 Do You Need a Firewall?

The decision to implement a firewall solution should not be made without doing some research and analysis. The decision to implement a firewall solution should be based on requirements that have been identified and documented, not because implementation of a firewall has been identified as a solution at other organizations (WEB_4 2005). Firewall implementations based on little or no thought can create more security holes and cause more network problems than no firewall implementation at all.

2.11.1.1 What does the firewall need to control or protect?

In order to make a sound decision, first identify what functions the firewall would need to perform. Will it control access to and from the network, or will it protect services and users?

What would the firewall control?

- Access into the network
- Access out of the network
- Access between internal networks, departments, or buildings
- Access for specific groups, users or addresses
- Access to specific resources or services

What would it need to protect?

- Specific machines or networks
- Specific services
- Information - private or public
- Users

After identifying what the firewall needs to control or protect, determine what might happen without this control and protection. What would happen if users had access to things they should not? What would happen if the unprotected services or information were compromised? Is the risk of not having control or protection great enough to warrant taking the next step in the assessing the need for a firewall solution?

2.11.1.2 What impact will a firewall have on your organization, network and users?

- What resources will be required to implement and maintain a firewall solution?

- Who will do the work? Are experienced technical personnel available for the job or will someone need to be hired from outside your organization?
- Is hardware available that meets the requirements to support a firewall solution?
- Will existing services be able to function through a firewall?
- What will the financial impact be on the organization? (Financial impact should include initial implementation costs, ongoing maintenance and upgrades, hardware and software costs, and technical support costs, whether the support is provided in-house or from an outside source.)

Examining what the firewall needs to do, how it will fit in the existing network, and how it will effect existing services and users, should help the organization make an informed decision on whether or not a firewall is needed. The next step in the selection process is to review or develop the organization's security policy.

2.12 Iptables

The iptables tool inserts and deletes rules from the kernel's packet filtering table. Each rule is a line that the kernel looks at to find out what to do with a packet. If all the criteria - or matches - are met, the target instruction is performed. Normally the rules are written in a syntax that looks like this (Andreasson 2001):

iptables (-t table) command (match) (target/jump)

To use another table than the standard table, the table specification at the point at which (table) is specified should be inserted. However, it is not necessary to state explicitly what table to use, since by default iptables uses the filter table on which to implement all commands. Neither do you have to specify the table at just this point in the rule. It could be set anywhere along the line. However, it is more or less standard to put the table specification at the beginning.

The command should always come first, or alternatively directly after the table specification. The "command" is used to tell the program what to do, for example to insert a rule or to add a rule to the end of the chain, or to delete a rule.

The "match" is the part of the rule that is sent to the kernel that details the specific character of the packet, what makes it different from all other packets. Here we could specify what IP address the packet comes from, from which network interface, the intended IP address, port, protocol or whatever.

Finally, the packet has the target. If all the matches are met for a packet, “target/jump” part of the command tells the kernel what to do with it. We could, for example, tell the kernel to send the packet to another chain that we have created ourselves, and which is part of this particular table. We could tell the kernel to drop the packet dead and do no further processing, or we could tell the kernel to send a specified reply to the sender.

The **-t** option specifies which table to use. Per default, the filter table is used. We may specify one of the following tables with the -t option.

The **nat** table is used mainly for Network Address Translation. "NAT"ed packets get their IP addresses altered, according to rules. Packets in a stream only traverse this table once. It is assumed that the first packet of a stream is allowed. The rest of the packets in the same stream are automatically "NAT"ed or Masqueraded etc, and will be subject to the same actions as the first packet. These will not go through this table again, but will nevertheless be treated like the first packet in the stream. This is the main reason why you should not do any filtering in this table. The PREROUTING chain is used to alter packets as soon as they get in to the firewall. The OUTPUT chain is used for altering locally generated packets (i.e., on the firewall) before they get to the routing decision. Finally the POSTROUTING chain is used to alter packets just as they are about to leave the firewall.

The **mangle** table is used mainly for mangling packets. Among other things, we can change the contents of different packets and that of their headers. Examples of this would be to change the **TTL**, **TOS** or **MARK**. The **MARK** is not really a change to the packet, but a mark value for the packet is set in kernel space. The table consists of five built in chains, the PREROUTING, POSTROUTING, OUTPUT, INPUT and FORWARD chains (Andreasson 2001). PREROUTING is used for altering packets just as they enter the firewall and before they hit the routing decision. POSTROUTING is used to mangle packets just after all routing decisions have been made. OUTPUT is used for altering locally generated packets before they enter the routing decision. INPUT is used to alter packets after they have been routed to the local computer itself, but before the user space application actually sees the data. FORWARD is used to mangle packets after they have hit the first routing decision, but before they actually hit the last routing decision. Note that mangle cannot be used for any kind of Network

Address Translation or Masquerading; the nat table was made for these kinds of operations.

The **filter** table should be used exclusively for filtering packets. For example, we could **DROP**, **LOG**, **ACCEPT** or **REJECT** packets without problems, as we can in the other tables. There are three chains built in to this table. The first one is named FORWARD and is used on all non-locally generated packets that are not destined for local host (the firewall) (Andreasson 2001). INPUT is used on all packets that are destined for our local host (the firewall) and OUTPUT is finally used for all locally generated packets.

When a packet first enters the firewall, it hits the hardware and then get has passed on to the proper device driver in the kernel. Then the packet starts to go through a series of steps in the kernel before it is either sent to the correct application (locally), or forwarded to another host or whatever happens to it. The packet goes through the different steps Figure 2.12.1 explains how packets traverse the different chains and in which order (Andreasson 2001).

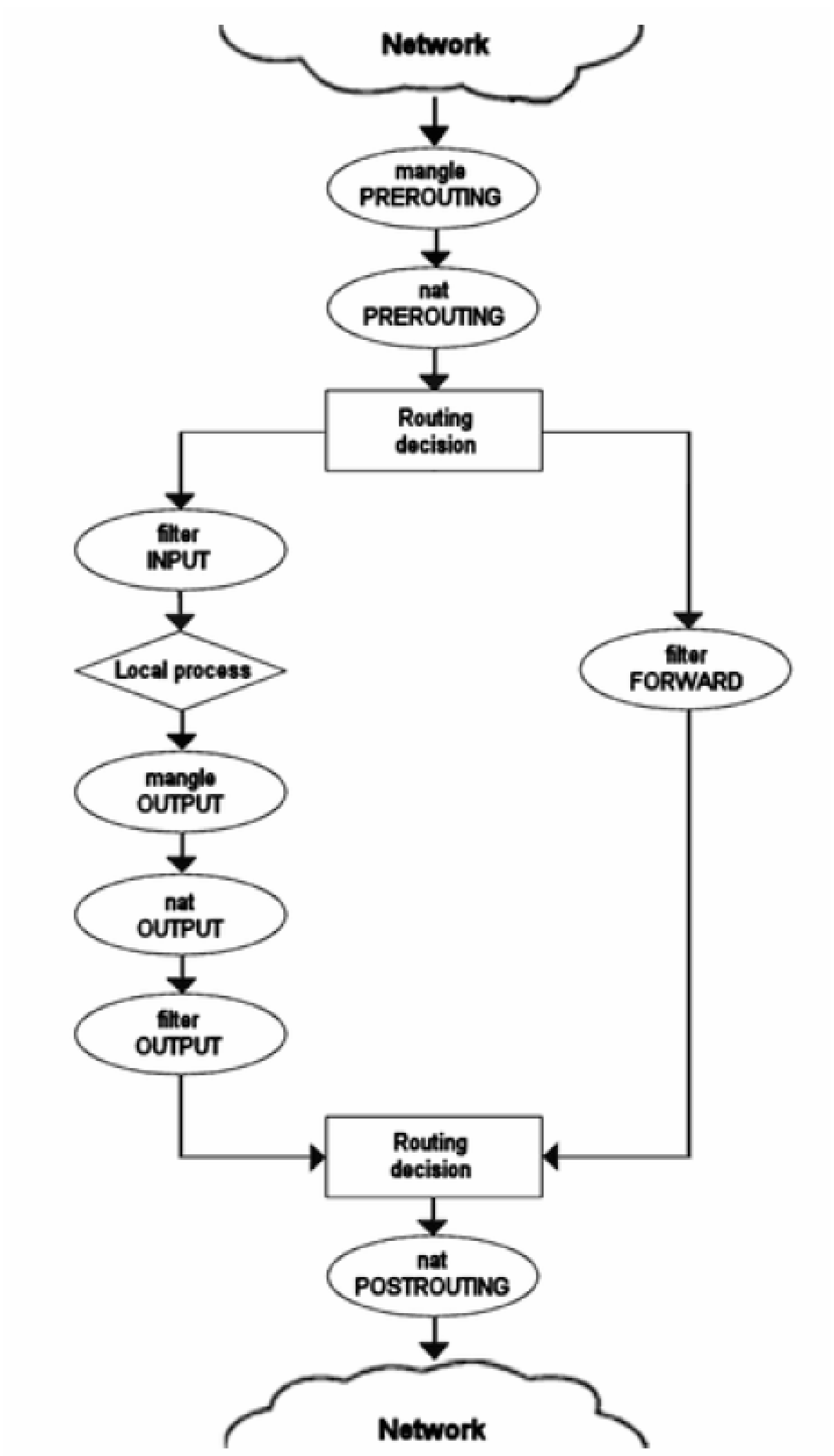

Figure 2.7. Picture of how the packets traverse the built in chains now.

2.12.1.1. Writing Rules

Iptables uses a set of chain rules. The three default chains are named INPUT, OUTPUT, and FORWARD. A chain is just a simple checklist of rules and specifies what to do with each of the packets. The chain rules will either ACCEPT a packet or DROP a packet. If the packet does not have any more rules left in the chain, the system will consult the chain policy to decide what to do. Most systems are setup with a policy of deny. Therefore, if the packet does not match any rules that "allow" it through, then it will "drop" it.

The first decision the kernel has to make upon receipt of a packet is "Where is the destination?" If the destination is for the box itself, it will consult the rules for the INPUT chain. If the destination is for another network interface (and IP Forwarding is enabled), the packet is compared against the FORWARD chain. As long as the packet gets an "ACCEPT" by one of the chain rules the packet will be forwarded on. If the Linux box itself needs to send network packets, it will consult the OUTPUT chain and if the packet is ACCEPTed by one of the rules, it will be sent out to the appropriate interface. One of the key concepts is that the INPUT and OUTPUT chains actually refer to the local machine rather than to all incoming and outgoing packets.

We can start with three built-in chains INPUT, OUTPUT and FORWARD which can not be deleted. The followings are operations to manipulate the chains (WEB_6 2004):

- Create a new chain (-N).
- Delete an empty chain (-X).
- Change the policy for a built-in chain. (-P).
- List the rules in a chain (-L).
- Flush the rules out of a chain (-F).
- Zero the packet and byte counters on all rules in a chain (-Z).

There are several ways to manipulate rules inside a chain:

- Append a new rule to a chain (-A).
- Insert a new rule at some position in a chain (-I).
- Replace a rule at some position in a chain (-R).
- Delete a rule at some position in a chain, or the first that matches (-D).

Some other options:

-j Specify the target (--jump)

-i Specify the input interface (--in-interface)

-o Specify the output interface (--out-interface)

-p Specify the protocol (--proto)

-s Specify the source (--source)

-d Specify the destination (--destination)

! Specifies an inversion (match addresses NOT equal to)

The followings are some examples, how to write rules to iptables and their explanations (what the rule does):

- /sbin/iptables -A INPUT -p tcp -j ACCEPT

This rule would accept all tcp traffic. This is a little too broad isn't it? Let's take a look at how we can specify some other options.

- /sbin/iptables -A FORWARD -s 192.168.1.0/24 -p tcp -j ACCEPT

This rule will allow traffic to be forwarded, as long as the protocol was tcp, and the source was a machine on the 192.168.1.0 subnet

- /sbin/iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

This rule, coupled with the one above will allow for MASQUERADE(ing) your internal network traffic, via NAT (Network Address Translation), so that you can share your internet connection with the rest of your network.

Some useful tcp options (these also work for udp):

--sport Filters on the source port

--dport Filters on the destination port

- /sbin/iptables -A FORWARD -p tcp --dport 80 -j ACCEPT

This rule would allow traffic going to the www port (80) to be forwarded on.

- /sbin/iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0 -j DNAT --to 5.6.7.8:8080

Change destination addresses of web traffic to 5.6.7.8, port 8080.

--tcp-flags allows you to filter on specific TCP flags. The first option after "--tcp-flags" specifies which flags are to be examined, and the second option specifies which flags are to be set. The list of possible flags is as follows SYN, ACK, FIN, RST, URG, PSH. Here is an example of tcp flags.

- /sbin/iptables -A INPUT -p tcp --tcp-flags ALL SYN -j DENY

CHAPTER 3

INTRUSION DETECTION SYSTEMS (IDS)

When a user of an information system takes an action that user was not legally allowed to take, it is called *intrusion* (Jones and Sielken 2000). The intruder may come from outside, or the intruder may be an insider, who exceeds his limited authority to take action. Whether or not the action is detrimental, it is of concern because it might be detrimental to the health of the system, or to the service provided by the system.

As information systems have come to be more comprehensive and a higher value asset of organizations, complex, *intrusion detection* subsystems have been incorporated as elements of operating systems, although not typically applications (Jones and Sielken 2000). Most intrusion detection systems attempt to detect suspected intrusion, and then they alert a system administrator.

Intrusion *detection* involves determining that some entity, an *intruder*, has attempted to gain, or worse, has gained unauthorized access to the system. None of the automated detection approaches of which we are aware seeks to identify an intruder before that intruder initiates interaction with the system. Of course, system administrators routinely take actions to prevent intrusion. These can include requiring passwords to be submitted before a user can gain any access to the system, fixing known vulnerabilities that an intruder might try to exploit in order to gain unauthorized access, blocking some or all network access, as well as restricting physical access. Intrusion detection systems are used in addition to such preventative measures.

Intruders are classified in two groups. *External intruders* do not have any authorized access to the system they attack. *Internal intruders* have some authority, but seek to gain additional ability to take action without legitimate authorization. J. P. Anderson divided internal intruders into three subgroups: masqueraders, clandestine, and legitimate (Jones and Sielken 2000). In later related work, Brownell Combs has divided internal intruders into two categories. He separates internal users who have accounts on the system from pseudo-internal intruders who are, or can be thought of as being, physically in space of legitimate users, but have no accounts (Jones and Sielken

2000). They do however have physical access to the same equipment used by those who have accounts. He shows how distinguishing the two categories can be distinguished enables better defense against the pseudo-internal intruders.

To limit the scope of the problem of detecting intrusions, system designers make a set of assumptions. Total physical destruction of the system, which is the ultimate denial of service, is not considered. Intrusion detection systems are usually based on the premise that the operating system, as well as the intrusion detection software, continues to function for at least some period of time so that it can alert administrators and support subsequent remedial action.

Second layer in the perimeter defense is intrusion detection systems (IDSs). The audits of security existed before the intrusion detection. Audit is the process of generating, storing and revising events of a system chronologically. IDS is the evolved version of the traditional audits (Gomez 2003).

Intrusion detection is the process of monitoring and searching networks of computers and systems for security policy violations (Bace 2000). Intrusion Detection Systems (IDSs) are software or hardware products that automate this monitoring and analysis process. An IDS inspects all inbound and outbound network activity, system logs and events, and identifies suspicious patterns or events that may indicate a network or system attack from someone attempting to break into or compromise a system (WEB_7 2002).

Theoretically, IDSs work like a burglar alarm, alerting security managers that an attack may be taking place so that they can respond accordingly. IDSs trigger these alerts by detecting anomalous traffic patterns or “signatures” that are characteristic of an attack. As in the physical world, our logical burglar alarm provides valuable notification that someone has managed to breach perimeter security measures, and should allow security managers to determine exactly what happened during the attack, and hopefully provide indications of how the security weakness might be addressed.

It is also assumed that intrusion detection is not a problem that can be solved once; continual vigilance is required. Complete physical isolation of a system from all possible, would-be external intruders is a simple and effective way of denying external intruders, but it may be unacceptable because physical isolation may render the system unable to perform its intended function. Some possible solution approaches cannot be used because they are in conflict with the service to be delivered.

In addition, potential internal intruders legitimately have access to the system for some purposes. Therefore, it is assumed that at least internal intruders, and possibly external intruders, have some access and therefore have some tools with which to attempt to penetrate the system. It is typically assumed that the system, usually meaning the operating system, does have flaws that can be exploited. Today, software is too complicated to assume otherwise. New flaws may be introduced in each software upgrade. Patching the system could eliminate known vulnerabilities. However, some vulnerabilities are too expensive to fix, or their elimination would also prevent desired functionality (Jones and Sielken 2000).

Vulnerabilities are usually assumed to be independent. Even if a known vulnerability is removed, a system administrator may run intrusion detection software in order to detect *attempts* at penetration, even though they are guaranteed to fail. Most intrusion detection systems do not depend on whether specific vulnerabilities have been eliminated or not. This use of intrusion detection tools can identify a would-be intruder so that his or her other activities may be monitored. New vulnerabilities may, of course, be discovered in the future.

IDSs have gained acceptance as a necessary addition to every organization's security infrastructure. Since they are first put on the security market, those organizations have several compelling reasons to acquire and use IDSs. Some of them are listed below (Bace and Mell 2001):

- To prevent problematic behaviors by increasing the perceived risk of discovery and punishment for those who would attack or otherwise abuse the system,
- To detect attacks and other security violations that are not prevented by other security measures,
- To detect and deal with the preambles to attacks (commonly experienced as network probes and other reconnaissance activities),
- To document the existing threat to an organization,
- To act as quality control tool for security design and administration, especially for large and complex enterprises,
- To provide useful information about intrusions that take place, allowing detailed analysis, recovery, and correction of causative factors.

Intrusion detection systems have the same critical job. It is critical that the system's detection mechanisms are accurate enough to differentiate between the good and bad traffic allowed by the firewall. The following are all possible results of intrusion detection:

- Undetected bad traffic (false negative)
- Detected bad traffic (true negative)
- Good traffic that the system thinks is bad (false positive – false alarm)
- Good traffic that the system identifies as good (true positive)

Undetected bad traffic: Failure to identify malicious traffic as an attack. This is the worst thing that can happen, because it means the IDS failed to do its job. Failing to detect an attack can occur when an IDS does not have adequate or comprehensive intrusion detection mechanisms in place. It also occurs when new attacks are created and then missed by poorly implemented detection mechanisms (OneSecure 2001). While it is virtually impossible to detect every attack, the goal of any system should be to minimize the number of undetected attacks.

Detected bad traffic: Identifying "real attacks" as an attack. This is the ideal result of an IDS. The ability to detect bad traffic with speed and reliability is referred to as intrusion detection accuracy. All other functions of the system hinge on this capability (OneSecure 2001). The more accurate the system, the more you can trust its abilities. A system must have proven accuracy before enabling it to take the necessary actions (such as dropping the connection) to secure the network.

Identifying good traffic as an attack: False alarm or false positive. This is the most troublesome and time-consuming aspect of IDS solutions. It occurs when the IDS sees something in legitimate and benign traffic that makes it believe there is an attack (OneSecure 2001). It is detrimental because each and every alarm needs to be investigated in order to determine whether an attack was successful and assess any resulting damage. Every moment spent investigating a false positive reduces the time available to investigate real threats. The result is that false positives can erode trust in the product; sometimes causing real attack alarms to be overlooked (the "crying wolf" effect). Most IDSs can be tuned to try to reduce the occurrence of false positives, however, the tuning process is often long and involved, sometimes taking weeks to accomplish (OneSecure 2001). In addition, because of the management design of current IDSs, tuning is often an all or nothing approach. This means that security managers must choose whether or not to look for a certain attack. If, in the interest of reducing false positives, the detection of certain attacks is turned completely “off,” those attacks will be able to go by the IDS completely undetected.

Nevertheless, false positives are not the result of poor software design by IDS vendors. As Stefan Axelsson demonstrated in his 1999 ACM presentation, (Axelsson 1999) there are some fundamental mathematical constraints that make false positives endemic to the whole paradigm of real-time signature (pattern) recognition. Deviations from baseline norms can be caused by a variety of factors, many of them innocuous. So, false positives are inherently part of signature-based intrusion detection schemes or any other type of anomaly detection system.

Identifying good traffic as good traffic: An ideal result of intrusion detection mechanisms, identifying good traffic for what it is good traffic (OneSecure 2001).

Users should be aware that most of the existing IDSs are not difficult to by-pass if the attacker is knowledgeable. In addition, users should be aware that IDSs generate voluminous logs that must be examined carefully if the IDS is to be effective (NIST 2002). The advantages and disadvantages of IDSs are summarized below:

Advantages:

- The deployment of network-based IDSs has little performance impact upon an existing network,
- Host-based IDSs are able to monitor events local to a host,
- IDSs can allow security managers, regardless of their level of security expertise, to track security problems on their systems, initiating incident handling procedures (Bace and Mell 2001).

Disadvantages:

- IDSs have a tendency to generate “false positives”. That is, they frequently generate alerts about an attack when none is taking place,
- The only way to eliminate false positives would be to tune the system down to the point where it would also ignore real attacks – yielding “false negatives” – an obviously unacceptable approach,
- IDSs are extremely administration-intensive. Highly skilled security professionals must constantly tune the system, update signatures, analyze alerts

to determine if they are real or false, and then respond with appropriate remedial action,

- An IDS must be closely monitored and continually fine-tuned to the usage patterns and vulnerabilities discovered in its deployed environment. Such maintenance typically consumes a fair amount of administrative resources and effort,
- A substantial amount of time may pass between the attack and the remediation, allowing the attacker to do irreversible damage in the meantime,
- Any IDS system that relies exclusively on documented attack profiles will be vulnerable to new, as-yet-undocumented attacks.

3.1 IDSs Types

There exist various IDS products in the market today. These products are categorized in several ways according to their different characteristics (Debar, Dacier and Wespi 1999):

Detection method:

- Behavior Based (Misuse detection)
- Knowledge Based (Anomaly detection)

Audit source location

- Host log files (Host based)
- Network Packets (Network based)

Behavior on detection

- Passive
- Active

3.1.1 Detection Method

The *detection method* describes the characteristics of the analyzer (Debar, Dacier and Wespi 1999). When the intrusion-detection system uses information about the normal behavior of the system it monitors, we qualify it as misuse detection. When the intrusion-detection system uses information about the attacks, we qualify it as anomaly detection.

3.1.1.2. Misuse Detection IDS

Knowledge-based intrusion-detection techniques apply the knowledge accumulated about specific attacks and system vulnerabilities. The intrusion-detection system contains information about these vulnerabilities and looks for attempts to exploit them. When such an attempt is detected, an alarm is triggered. In other words, any action that is not explicitly recognized as an attack is considered acceptable. Essentially, the IDS looks for a specific attack that has already been documented. Like a virus detection system, misuse detection software is only as good as the database of attack signatures that it uses to compare packets against. Therefore, the accuracy of misuse intrusion detection systems is considered good. However, their completeness requires that their knowledge of attacks be updated regularly.

Misuse detection provides various benefits. One of the first benefits is that the signature definitions are modeled on known intrusive activity. Furthermore, the user can examine the signature database, and quickly determine which intrusive activity the misuse detection system is programmed to alert on. Another benefit is that the misuse detection system begins protecting your network immediately upon installation. One final benefit is that the system is easy to understand. When an alarm fires, the user can relate this directly to a specific type of activity occurring on the network.

Along with the numerous benefits, misuse detection systems also have their share of drawbacks. One of the biggest problems is maintaining state information for signatures in which the intrusive activity encompasses multiple discrete events (that is, the complete attack signature occurs in multiple packets on the network) (Carter 2002). Another drawback is that your misuse detection system must have a signature defined for all of the possible attacks that an attacker may launch against your network. This leads to the necessity for frequent signature updates to keep the signature database of your misuse detection system up-to-date. One final problem with misuse detection systems is that someone may set up the misuse detection system in their lab and intentionally try to find ways to launch attacks that bypass detection by the misuse detection system.

3.1.1.3. Anomaly Detection IDS

Anomaly detection techniques assume that an intrusion can be detected by observing a deviation from normal or expected behavior of the system or the users (Debar, Dacier and Wespi 1999). The model of normal or valid behavior is extracted from reference information collected by various means. The security manager defines the baseline, or normal, state of the network's traffic load, breakdown, protocol, and typical packet size. The intrusion-detection system later compares this model with the current activity. If a deviation is observed, an alarm is generated. In other words, anything that does not correspond to a previously learned behavior is considered intrusive. Therefore, the intrusion-detection system might be complete, but its accuracy is a difficult issue. The anomaly detection technique is as good as its normal model definition.

Anomaly detection systems offer several benefits. First, they can detect insider attacks or account theft very easily. If a real user or someone using a stolen account starts performing actions that are outside the normal user profile, it generates an alarm. Second, because the system is based on customized profiles, it is very difficult for an attacker to know with certainty what activity he can do without setting off an alarm. Probably the largest benefit, however, is that intrusive activity is not based on specific traffic that represents known intrusive activity (as in a misuse IDS). An anomaly detection system can potentially detect an attack the first time it is used (Debar, Dacier and Wespi 1999). The intrusive activity generates an alarm because it deviates from normal activity, not because someone configured the system to look for a specific stream of traffic.

Like every IDS, anomaly detection systems also suffer from several drawbacks. The first obvious drawback is that the system must be trained to create the appropriate user profiles. During the training period to define what normal traffic looks like on your network, the network is not protected from attack. Just defining "normal" is a challenge in itself (Debar, Dacier and Wespi 1999). Maintenance of the profiles can also become time-consuming. Nevertheless, the biggest drawback to anomaly detection is probably the complexity of the system and the difficulty of associating an alarm with the specific event that triggered the alarm. Furthermore, you have no guarantee that a specific attack will even generate an alarm. If the intrusive activity is too close to normal user activity, then the attack will go unnoticed. It is also difficult to know which attacks will set off alarms unless actually test the attacks against the network using various user profiles (Carter 2002).

3.1.2 Audit Source Location

The audit source location distinguishes among intrusion detection systems based on the kind of input information they analyze. This input information can be audit trails, system logs or network packets.

3.1.2.1. Host Based IDS (HIDS)

In a host-based system or HIDS, activities on each individual computer or host are examined. Host audit sources are the only way to gather information about the activities of the users of a given machine. On the other hand, they are also vulnerable to alterations in the case of a successful attack. This creates an important real-time constraint on host-based intrusion-detection systems, which have to process the audit trail and generate alarms before an attacker taking over the machine can subvert either the audit trail or the intrusion-detection system itself. It is advisable to place HIDSs on all mission-critical systems, even those that should not, in theory, allow external access.

The major benefit of a host-based monitoring system is that the success or failure of an attack can be readily determined. A network-based system alarms on the presence of intrusive activity, but cannot always ascertain the success or failure of such an attack. Host-based detection systems also do not have to worry about fragmentation attacks or variable time-to-live attacks because the host's own stack takes care of these issues. One advantage of a host-based monitoring system is that if the network traffic stream is encrypted, the host-based monitoring system has access to the traffic in unencrypted form. Although host-based intrusion detection does not offer true real-time response, it can come extremely close if implemented correctly (ISS 1998). Many current host-based systems receive an interrupt from the operating system when there is a new log file entry. This new entry can be processed immediately, significantly reducing the time between attack recognition and response. There remains a delay between when the operating system records the event and the host-based system recognizes it, but in many cases an intruder can be detected and stopped before damage is done.

Two of the major drawbacks to a host-based monitoring system are as follows: incomplete network picture and necessity to support multiple operating systems. By only examining information at the local host level, a host-base monitoring system has difficulty constructing an accurate network picture or coordinating the events happening across your entire network. The other difficulty lies in the fact that a host-based monitoring system needs to run on every system in your network. This requires verifying support for all of the different operating systems that you use on your network.

3.1.2.2. Network Based IDS (NIDS)

Network-based IDSs, or NIDS, analyze the individual packets flowing through the network. Network packets are usually “sniffed” off the network, although they can be derived from the output of routers and switches. Network-based IDSs often consist of a set of single-purpose sensors placed at various points in a network. These units monitor network traffic, performing local analysis of that traffic and reporting attacks to a central management console. NIDSs analyze traffic moving across the network in much greater detail than a firewall. Therefore, NIDSs can detect malicious packets that are designed to be overlooked by a firewall’s simplistic filtering rules. NIDSs also watch for attacks that originate from within a network. That is why, they are complement for firewalls.

Network-based intrusion detection is normally more effective than host-based intrusion detection due to the fact that a single system can monitor multiple systems and resources.

A network-based monitoring system has the benefit of seeing and coordinating attacks that are occurring across your entire network very easily. Seeing the attacks against your entire network gives you a clear indication of the extent to which your network is being attacked. Furthermore, because the monitoring system is only examining traffic from your network, it does not have to support every type of operating system that you use on your network. Network-based IDS use live network traffic for real-time attack detection. Therefore, an attacker cannot remove the evidence (ISS

1998). Captured data includes not only the method of attack, but information that may help lead to identification and prosecution. Since many hackers understand audit logs, they know how to manipulate these files to cover their tracks, frustrating host-based systems that need this information to detect an intrusion. Network-based IDS add valuable data for determining malicious intent. A network-based IDS placed outside of a firewall can detect attacks intended for resources behind the firewall, even though the firewall may be rejecting these attempts. Host-based systems do not see rejected attacks that never hit a host inside the firewall. This lost information can be critical in evaluating and refining security policies.

One disadvantage of NIDS is that encryption of the network traffic stream can essentially blind your network based IDS. Reconstructing fragmented traffic can also be a difficult problem to solve. Probably the biggest drawback to network-based monitoring, however, is that as networks become increasingly larger (with respect to bandwidth), it becomes more difficult to place a network based IDS at a single location on your network and capture all of the traffic successfully (Carter 2002). This then requires the utilization of more sensors throughout your network, which increases the costs of your IDS.

3.1.3 Behavior on Detection

Behavior on detection describes the response of the intrusion-detection system to attacks. When it actively reacts to the attack by taking either corrective closing holes or proactive logging out possible attackers, closing down services actions, then the intrusion detection system is said to be active (Debar, Dacier and Wespi 1999). If the intrusion-detection system merely generates alarms including paging, etc., it is said to be passive.

3.1.3.1. Passive System

Many intrusion detection systems merely log the intrusion and security administrator, by email or pager for example. This is known as passive-response intrusion detection, as it does not actively attempts to stop the intrusion. Instead, a system administrator or someone else will have to respond to the alarm, take appropriate action to halt the attack, and possibly identify the intruder. Modern IDSs offer a wide range of options to send notifications of intrusions, including pager, cell phone, email, SNMP trap messages, or simply a message box on the administrator's PC. It is important to make sure that the notifications are send in a secure manner to prevent the attacker from intercepting or altering them.

3.1.3.2. Active System

Active-response IDSs automatically take action in response to a detected intrusion. The exact action differs per product and depends on the severity and type of attack. A common active response is increasing the sensitivity level of the IDS to collect additional information about the attack and the attacker. Another possible active response is making changes to the configuration of systems or network devices such as routers and firewalls to stop the intrusion and block the attacker. This could involve blocking the source address of the attacker, restarting a server or service, closing connections or ports, and resetting TCP sessions.

3.2 Detection Techniques

The ideal performance on an intrusion detection system is to identify as many attacks as possible and limit the number of false alarms (OneSecure 2001). Unfortunately, there is no single detection mechanism available today that an IDS can deploy to detect every type of attack. As a result, only a system that combines a variety of technologies to detect different types of attacks can get the job done. The most common way to implement a combination of detection methods was to purchase two or more products and run them together. The two most commonly available network intrusion detection mechanisms on the market today are signature detection and protocol anomaly detection (OneSecure 2001). Signature-based systems look for known attack patterns (signatures) in traffic. Signature detection finds attacks for which a signature is written, but is unable to detect new attacks or many very complicated attacks. Protocol anomaly detection-based systems, on the other hand, do a good job of detecting some of the unknown attacks, but are unable to identify attacks that operate without violating any protocols. These detection mechanisms include:

- Intrusion detection using signature,
- Intrusion detection using protocol anomalies,
- Intrusion detection using stateful signatures.

3.2.1 Intrusion Detection Using Signature

The evolution of NIDSs started with the implementation of a non-intrusive packet monitor, called a sniffer because of its ability to "sniff" the packets on the network. Intrusion detection vendors applied the packet-monitoring concept to build systems that performed packet signature detection (OneSecure 2001). Signature-based detectors analyze system activity, looking for events or sets of events that match a predefined pattern of events that describe a known attack. They compare events and packets with signatures stored in their database and find out the matching ones. The most common form of signature-based detection used in commercial products specifies each pattern of events corresponding to an attack as a separate signature (Bace and Mell 2001). Signature detection finds attacks for which a signature is written and it is very effective at detecting attacks without generating an overwhelming number of false positives.

However, there are many drawbacks to signature-based approach to intrusion detection, especially if effort is placed entirely on building up a large repository of attack signatures, without regard to how the traffic is reassembled, decoded, normalized and analyzed. It is a problem when information is transmitted over the network; the information is split into numbered TCP (Transmission Control Protocol) segments that are sent as packets. In an ideal world, the packets would be transmitted in sequence and without loss. But, unfortunately, that's not the case. When a message is actually transmitted, the network will deliver the packets randomly (out of sequence) or as even smaller pieces of data (called fragments), which are broken down by networking devices, such as routers, to facilitate ease of transmission. Even worse, for whatever reason, packets can get "lost" or can be duplicated (OneSecure 2001).

Another disadvantage of this mechanism is that it is unable to detect new attacks. Therefore, it must be constantly updated with signatures of new attacks. However, signature updates may sometimes result in inability of detecting previously detected attacks. Lastly, signature-based detectors can not detect many very complicated attacks.

3.2.2 Intrusion Detection Using Protocol Anomalies

Protocol anomaly detection, which is sometimes called protocol analysis, is the ability to analyze packet flows (the uni-directional communication between two systems) to identify irregularities in the generally accepted Internet rules of communication (OneSecure 2001). Those rules are defined by open-protocols and published standards (RFC-Request For Comment), as well as vendor-defined specifications for communication between networked devices.

Protocol anomaly detection attempts to save time by first identifying the protocol, and then looking specifically for anomalous activity or attack patterns relevant to that protocol. By doing so, it can do a much more targeted, and thus more effective search (TopLayerNetworks 2002). In other words, protocol anomaly detection identifies traffic that doesn't meet specifications or violates the relevant standards. Once an irregularity is identified, it can be used to make network security decisions. This is very effective in detecting suspicious activity, such as a buffer-overflow attack.

The advantages of protocol anomaly detection are that it can detect (OneSecure 2001):

- Unknown and new attacks, based on the fact that these attacks deviate from protocol standards,
- Attacks that bypass systems that implement other detection methods,
- Slightly modified attacks that change the format of known attack patterns, with no affect on the strength of the attack, to evade signature-based systems.

An example of detecting the FTP bounce attack using protocol anomaly is described below (OneSecure 2001):

The FTP bounce attack exploits a design flaw in the specifications of FTP (File Transfer Protocol). To download or upload files, a user (FTP client) must first connect to an FTP server. When this happens, the server requires the client to send the IP address and port number to which the file should be sent to or taken from. This is done via a mechanism called a "Port Command." However, the "Port" command specification does not limit the IP address to the user's address. Because of this, an attacker can tell the FTP server to open a connection to an IP address that is different from the user's address and then use the open port to transfer files containing a Trojan through the FTP server onto the victim. Since protocol anomaly detection is designed to look at network relationships and determine whether both sides are acting within the normal specifications, IDS can parse the requests in a "Port" command whenever seen and compare it to the IP address from which the "Port" command arrived. If they do not match, the IDS needs to send an alarm.

Protocol anomaly detection-based systems do a good job of detecting some of the unknown attacks like the one described. But their main drawback is that they are unable to identify attacks that operate without violating any protocols, such as Trojan or Worm. These attacks install and open up a backdoor on a network resource. This backdoor lays inactive until the attacker activates it and takes control over the resource. Besides that, protocol anomaly detection-based systems usually produce a large number of false alarms due to the unpredictable behaviors of users and networks.

3.2.3 Intrusion detection using stateful signatures:

Another alternative approach that overcomes the accuracy deficiencies of packet signature detection is stateful signature detection. This advanced detection mechanism identifies attack patterns by utilizing both stateful inspection and protocol analysis, which is performed as part of protocol anomaly detection (OneSecure 2001). As a result, stateful signature detection systems understand the context of each data byte and the state of the client and server at the time of transmission. This means that stateful signatures can be compared to only relevant data bytes, according to the communication state to which each signature is relevant.

Stateful signature detection mechanism looks at the context and the placement of signature to make smarter decisions about whether it represents an attack. The IDS keeps track of the state of the connection with the outside entity, and considers the broader context of all the transactions initiated during the connection (TopLayerNetworks 2002). In other words, stateful signatures only look for an attack in the state of the communication where that attack can cause damage, thus significantly improving performance and reducing false positives.

The drawback of this system is the same as the signature-based detectors; it catches only known perpetrators and only if the signature database is constantly updated. Unfortunately, the people out there trying to exploit networks are neither lazy nor stupid; they constantly unleash new variations and new attacks. With each new attack, new signatures have to be "taught" to the IDS. Over time, this has led to a need for IDS products to hold literally thousands of attack signatures, and constantly scan for them all (TopLayerNetworks 2002). In addition, while the signature database being constantly updated, attackers know exactly how the IDS products work, so they continuously make slight alterations to elude detection. Thus, a more intelligent signature mechanism is needed to bring about more accurate results.

3.3 Response Options for IDSs

Once IDSs have obtained event information and analyzed it to find symptoms of attacks, they generate responses. Some of these responses involve reporting results and findings to a pre-specified location. Others involve more active automated responses (Bace and Mell 2001). Though researchers are tempted to underrate the importance of good response functions in IDSs, they are actually very important. Today, IDS products support a wide range of response options, often categorized as active responses, passive responses, or some mixture of the two.

3.3.1 Active Responses

Active IDS responses are automated actions taken when certain types of intrusions are detected. There are three categories of active responses:

Collect additional information: The most innocuous, but at times most productive, active response is to collect additional information about a suspected attack. This might involve increasing the level of sensitivity of information sources (for instance, turning up the number of events logged by an operating system audit trail, or increasing the sensitivity of a network monitor to capture all packets, not just those targeting a particular port or target system). Collecting additional information is helpful for several reasons. The additional information collected can help resolve the detection of the attack (assisting the system in diagnosing whether an attack did or did not take place). This option also allows the organization to gather information that can be used to support investigation and apprehension of the attacker, and to support criminal and civil legal remedies (Bace and Mell 2001).

Change the environment: Another active response is to halt an attack in progress and then block subsequent access by the attacker. Typically, IDSs do not have the ability to block a specific person's access, but instead block IP addresses from which the attacker appears to be coming. It is very difficult to block a determined and knowledgeable attacker, but IDSs can often deter expert attackers or stop novice attackers by taking the following actions (Bace and Mell 2001):

- Injecting TCP reset packets into the attacker's connection to the victim system, thereby terminating the connection,
- Reconfiguring routers and firewalls to block packets from the attacker's apparent location (IP address or site),
- Reconfiguring routers and firewalls to block the network ports, protocols, or services being used by an attacker, and
- In extreme situations, reconfiguring routers and firewalls to break all connections that use certain network interfaces.

Take action against the intruder: The most aggressive form of this response involves launching attacks against or attempting to actively gain information about the attacker's host or site. Due to legal ambiguities about civil liability, this option can represent a greater risk than the attack it is intended to block. The primary reason for approaching this option with a great deal of caution is that it may be illegal. Furthermore, since many attackers use false network addresses when attacking systems, this action has a high risk of causing damage to innocent Internet sites and users. Finally, strike back can escalate the attack, provoking an attacker who originally intended only to browse a site to take more aggressive action (Bace and Mell 2001).

3.3.2 Passive Responses

Passive IDS responses provide information to system users, relying on them to take necessary action based on that information. Many IDS products rely solely on passive responses.

Alarms and notifications: Alarms and notifications are generated by IDSs to inform users when attacks are detected. The most common form of alarm is an onscreen alert or popup window. This is displayed on the IDS console or on other systems as specified by the user during the configuration of the IDS. The information provided in the alarm message varies widely, ranging from a notification that an intrusion has taken place to extremely detailed messages outlining the IP addresses of the source and target of the attack, the specific attack tool used to gain access, and the outcome of the attack (Bace and Mell 2001). Another set of options is to configure the IDSs so that they send alert messages to cellular phones and pagers carried by incident response teams or security managers.

SNMP traps and plug-ins: Some commercial IDSs use SNMP traps and messages to send alarms to central network management consoles. This provides the ability to adapt the entire network infrastructure to respond to a detected attack, the ability to shift the processing load, associated with an active response, to a system other than the one being targeted by the attack, and the ability to use common communication channels (Bace and Mell 2001).

3.4 Snort

Snort is a lightweight network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. (Roesch 2003). Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes modular plugin architecture.

3.4.1 Snort Modes

Snort operates in two basic modes: packet sniffer mode and NIDS mode. It can be used as a packet sniffer, like tcpdump or snoop (Rafeeq 2003). When sniffing packets, Snort can also log these packets to a log file. The file can be viewed later on using Snort or tcpdump. No intrusion detection activity is done by Snort in this mode of operation. Using Snort for this purpose is not very useful as there are many other tools available for packet logging.

When Snort is used in network intrusion detection (NIDS) mode, it uses its rules to find out if there is any network intrusion detection activity.

The rules contain the information that defines the who, where, and what of a packet, as well as what to do in the event that a packet with all the attributes indicated in the rule should show up. The first item in a rule is the rule action. The rule action tells Snort what to do when it finds a packet that matches the rule criteria. There are five available default actions in Snort: alert, log, pass, activate, and dynamic.

- **alert** - generate an alert using the selected alert method, and then log the packet,
- **log** - log the packet,
- **pass** - ignore the packet,
- **activate** - alert and then turn on another dynamic rule,
- **dynamic** - remain idle until activated by an activate rule , then act as a log rule.

One can also define his own rule types and associate one or more output plug-ins with them. He can then use the rule types as actions in Snort (Snort 2004) rules.

3.4.1.1. Network Sniffer Mode

In the network sniffer mode, Snort acts like the commonly used program tcpdump. It can capture and display packets from the network with different levels of detail on the console. You don’t need a configuration file to run Snort in the packet sniffing mode. The following command displays information about each packet flowing on the network segment:

```
./snort -v
```

Snort will continue to display captured packets on the screen until you break using Ctrl-C. At the time Snort terminates, it will display statistical information. The following is a typical output for a TCP packet (Rafeeq 2003):

```
11/20-15:56:14.633891 192.168.1.2:22 -> 192.168.1.100:2474 TCP TTL:64
TOS:0x10 ID:57044 IpLen:20 DgmLen:184 DF ***AP*** Seq: 0xF5683D7A Ack:
0x9DAEEE9C Win: 0x6330 TcpLen: 20
```

If you analyze the output, you can see the following information about the packet:

- Date and time the packet was captured.
- Source IP address is 192.168.1.2 .
- Source port number is 22.
- Destination IP address is 192.168.1.100.
- Destination port is 2474.
- Transport layer protocol used in this packet is TCP.
- Time To Live or TTL value in the IP header part is 64.
- Type of Service or TOS value is 0x10.
- Packet ID is 57044.
- Length of IP header is 20.
- IP payload is 184 bytes long.
- Don't Fragment or DF bit is set in IP header.
- Two TCP flags A and P are on.
- TCP sequence number is 0xF5683D7A.
- Acknowledgement number in TCP header is 0xDAEEE9C.
- TCP Window field is 0x6330.
- TCP header length is 20.

3.4.1.2. Network Intrusion Detection Mode

In intrusion detection mode, Snort does not log each captured packet as it does in the network sniffer mode. Instead, it applies rules on all captured packets. If a packet matches a rule, only then is it logged or an alert is generated. If a packet does not match any rule, the packet is dropped silently and no log entry is created. When you use Snort in intrusion detection mode, typically you provide a configuration file on the command line. This configuration file contains Snort rules or reference to other files that contain Snort rules. In addition to rules, the configuration file also contains information about input and output plug-ins. The typical name of the Snort configuration file is snort.conf. The following command starts Snort in the Network Intrusion Detection (NID) mode:

```
snort -c /opt/snort/etc/snort.conf
```

When start this command, Snort will read the configuration file snort.conf and all other files included in this file. Typically these files contain Snort rules and configuration data. After reading these files, Snort will build its internal data structures and rule chains. All captured packets will then be matched against these rules and appropriate action will be taken, if configured to do so.

If snort.conf file any other file included in this file is modified, Snort have to be restarted for the changes to take effect. Other command line options and switches can be used when Snort is working in IDS mode. For example, you can log data into files as well as display data on the command line. However if Snort is being used for long-term monitoring, the more data you log, the more disk space you need. Logging data to the console also requires some processing power and the processing power of the host where Snort is running becomes a consideration. The following command will log data to /var/log/snort directory and will display it on the console screen in addition to acting as NIDS:

```
snort -dev -l /var/log/snort -c /etc/snort/snort.conf
```

However in most real-life situations, you will use -D command line switch with Snort so that it does not log on the console but runs as a daemon.

3.4.2 Components of Snort

Snort is logically divided into multiple components. These components work together to detect particular attacks and to generate output in a required format from the detection system. Snort consists of the following major components (Rafeeq 2003):

- Packet Decoder
- Preprocessors
- Detection Engine
- Logging and Alerting System
- Output Modules

Figure 5-2 shows the arrangement of these components. Any data packet coming from the network enters the packet decoder. And on its way towards the output modules, it is either dropped, logged or an alert is generated.

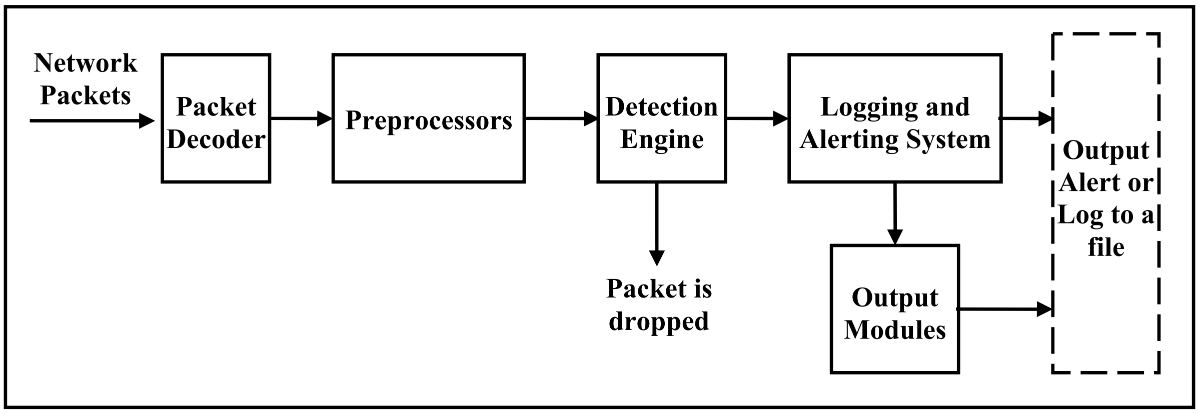

Figure 3.1 Components of Snort (Rafeeq 2003)
(Source: “Intrusion Detection Systems with Snort”. Prentice Hall PTR. ISBN 0-13-140733-3)

3.4.2.1. Packet Decoder

The packet decoder takes packets from different types of network interfaces and prepares the packets to be preprocessed or to be sent to the detection engine. The interfaces may be Ethernet, SLIP, PPP, WiFi and so on (Rafeeq 2003). It parses the packet and decodes the string of bytes into a packet structure that is formed of protocol fields and flags. Each subroutine in the decoder imposes order on the packet data by overlaying data structures on the raw network traffic. These decoding routines are called in order through the protocol stack, from the data link layer up through the transport layer, finally ending at the application layer. During this decoding process, it validates the length and checksum fields. It then forwards the valid packets to the preprocessors.

3.4.2.2. Preprocessors

When a packet is received by Snort (Snort 2004), it may not be ready for processing by the main Snort (Snort 2004) detection engine and application of Snort (Snort 2004) rules. For example, a packet may be fragmented. Before searching a string within the packet or determine its exact size, defragmentation is required by assembling all fragments of the data packet. On IDS, before applying any rules or try to find a signature, the packets have to be reassembled (Rafeeq 2003). The job of a preprocessor is to make a packet suitable for the detection engine to apply different rules to it. In addition, some preprocessors are used for other tasks such as detection of anomalies and obvious errors in data packets, decoding of HTTP URI. All enabled preprocessors operate on each packet. There is no way to bypass some of the preprocessors based upon some criteria.

3.4.2.3. The Detection Engine

The detection engine is the most important part of Snort (Snort 2004). Its responsibility is to detect if any intrusion activity exists in a packet. The detection engine employs Snort (Snort 2004) rules for this purpose. The rules are read into internal data structures or chains where they are matched against all packets. Snort (Snort 2004) organizes parts of packets to make the job of matching rules against them faster. It maintains detection rules in a two dimensional linked list of what are termed Chain Headers and Chain Options. The commonalities are condensed into a single Chain Header and individual detection signatures are kept in Chain Option structures. If a packet matches any rule, appropriate action is taken; otherwise the packet is dropped. Appropriate actions may be logging the packet or generating alerts.

3.4.2.4. Logging and Alerting System

This system is responsible from the generation of alerts and logging of packets and messages. Depending upon what the detection engine finds inside a packet, the packet may be used to log the activity or generate an alert. All of the log files are stored under a preconfigured location by default. This location can be configured using command line options. There are many command line options to modify the type and detail of information that is logged by the logging and alerting system.

3.4.2.5. Output Modules

Basically, these modules control the type of output generated by the logging and alerting system. Depending on the configuration, output modules can send output messages to a number of other destinations. Commonly used output modules are:

- The database module is used to store Snort (Snort 2004) output data in databases, such as MySQL, MSSQL or Oracle,
- The SNMP module can be used to send Snort (Snort 2004) alerts in the form of traps to a management server,
- The Sending Server Message Block (SMB) alerts module can send alerts to Microsoft Windows machines in the form of pop-up SMB alert windows,
- The syslog module logs messages to the syslog utility (using this module you can log messages to a centralized logging server.),
- XML or CSV modules can be used to save data in XML or comma separated files. The CSV files can then be imported into databases or spreadsheet software for further processing or analysis.

CHAPTER 4

FIREWALL MONITORING

Many people say that a firewall is only as good as its implementation. In other words, if a firewall is not implemented correctly, it may be ineffective.

In complex network environments it can be easy to make mistakes during the implementation process. There are many crucial components associated with a firewall's implementation. Each of these components must be identified, configured, and tested. The one component that has the most impact on the network and on the firewall's ability to provide the appropriate access control is the configuration of the rules and filters.

The ideal implementation plan would be to install the firewall solution in a test environment, test the rules and filters and then implement the solution in the production environment. In many situations this is just not possible (Schultz 1996). Many organizations configure rules and filters and implement them without testing them. Regardless of whether a test environment is available, time should be taken to test the rules and make sure they are allowing and denying access in the appropriate ways.

The point is that effective firewall testing is much more than simply unleashing a series of attacks, then determining whether or not they defeated the firewall's security mechanisms. Too much is at stake, and the value of firewall testing can be greatly diminished if it is not conducted properly. To be done properly, firewall testing should involve at least as much planning as the testing activity itself. Planning is the most basic step in effective firewall testing.

4.1 Basic Rules for Firewall Testing

Probably the most important point overall is planning. Included in the planning phase is the first step in the procedure to get the approval of the management! This should be written down and must be obtained in advance. Don’t do a test on a firewall or network just because someone is calling you that can be a serious trouble (Haeni 1997).

A part of this management approval will be an authorization list. This list establishes what we are allowed to do during testing and what we are not. This can mean for example to scan ports on a firewall but not perform *syn flood* attacks to disable the firewall. Then it is important that we don't do more than is authorized by this list. Generally this means that we look for (potential) weaknesses but we do not exploit them. In the same spirit we are not disrupting or changing any systems except for what is approved by the client.

4.2 Questions to Answer with Firewall Testing

In the case of firewall testing, we want to know first if the firewall policy (part of the information technology (IT) policy) is correctly implemented.

An internet firewall policy should contain the following points (Haeni 1997).

- Security Requirements
  - access control
  - assurance
  - logging
  - alarming
  - availability
- Required Functionality
  - outgoing services
  - incoming services
  - services provided to the internet

Another important point that should be verified is that no leakages in the perimeter exist. The firewall should be the only way to communicate with the outside world. Unfortunately, the topology of a network is often not entirely known and there exist other connections to (mostly) the internet. These can be dial-up connections, ISDN connections that are existent without knowledge of the network administrators. An additional danger exists with dial-in lines where the user can access the computer at work from home. The appearance of additional connections can increase when a firewall is set up as suddenly the access to services like WWW or FTP are restricted. A clear policy is needed that can enforce punishes for such behavior or will never get this problem under control. The strongest and best maintained firewall is of no use if we simply can bypass it.

We also want to determine if the logging of the events is adequate. We want not only to be able to control who is getting access to our network but also to track down an intruder and to find security holes when/after an incident occurs. In the same spirit, we want to test if the alarming is adequate. When a probe or attack is launched against our network or gateway, the firewall has to detect this and alarm the network and/or security administrators that they can take adequate actions. This alarming is typically an e-mail message and a message to the administrator's pager.

We can also determine if the reaction from the administrators is according to the intrusion detection policy. First of all, we must have such a policy. Then, the management has to decide if we really want to test this policy, as you have to launch the probing without announcing it to the involved network and security administrators. You can probably provoke a negative "corporate feeling" if you wake them up in the middle of the night when the alarming of the firewall is going off, they race to the office and spend the next 10 hours trying to keep the (so believed) hacker out of the system and trying to reconstruct what happened. They will probably not appreciate it if it was only a test. On the other hand if you announce a firewall test 24 hours before it happens, it is quite probable that they run around, applying patches and stuffing holes that they know to exist. The decision which way to go (or finding something in between) has to be made by the management.

After a firewall is set up, configured and necessary rules are written, it is expected to operate as it is configured. Experiences have shown that that is not always true. The firewall may not operate correctly due to malfunction or hacking, internal network may be open to attacks and this situation should be recovered as fast as possible. To be able to detect such problems security administrator may detect the firewall or traffic periodically. But in the time interval between the attacks succeeded (or malfunction of firewall) and security administrator detect it anything can happen to you network. Therefore, the firewall should be monitored for 24 hour period. It is not possible for people to do this task. So, there must be an automated monitoring tool. IDS tools may perform such a monitoring function, by taking care of the security of IT systems and detecting potential intrusions (Dworakowski 2004).

CHAPTER 5

PROPOSED ARCHITECTURE & EXPERIMENTAL ENVIRONMENT

In this section, an explanation of the proposed architecture is given. The proposed architecture (Figure 5.1) contains a firewall and two Intrusion Detection Systems (IDSs). One IDS monitors the traffic between the external network and firewall, while the other is monitoring the traffic between firewall and internal network. Intrusion detection systems also communicate to compare their monitoring results.

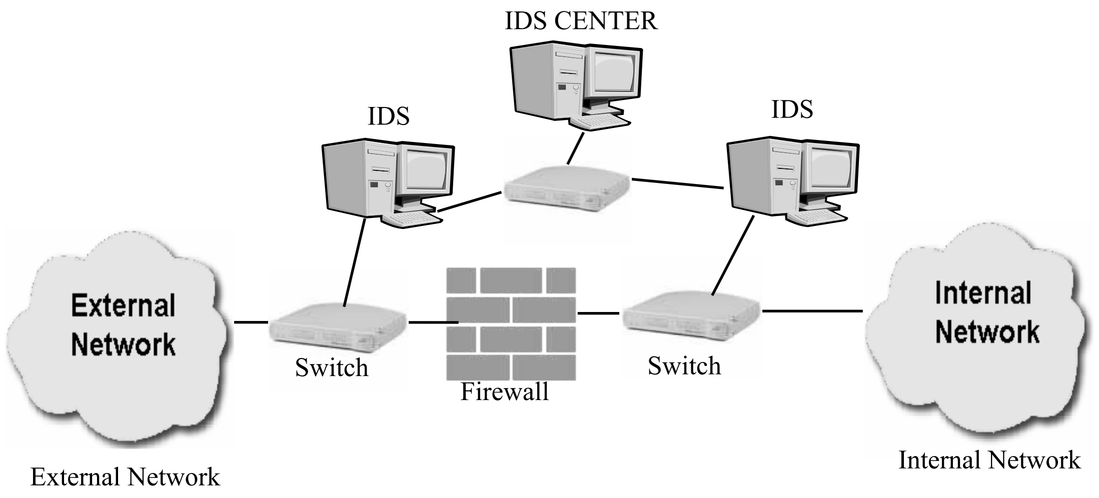

Figure 5.1. Proposed architecture.

Traffic from external network to internal network, first comes to external interface, and then firewall filters the traffic according to the previously defined rules. Traffic passed by the firewall is forwarded to internal interface, and then reaches its destination in internal network. It is the same for the traffic from internal network to the external network in reverse.

Intrusion Detection Systems are placed before and after the firewall. While the traffic flows from internal network to the external network or from external network to internal network, it is monitored by the IDSs. The decision can be made by comparing the traffic before and after it is filtered by the firewall. To get information about the traffic before it is filtered and after it is, IDSs are used. The IDS between the external network and firewall collects information about the incoming traffic before it is filtered. The same IDS collects the information about the outgoing traffic after it is filtered. In the same way the IDS between the firewall and internal network collects information about the outgoing traffic before it is filtered and the incoming traffic after it is filtered.

The firewall filters the traffic according to predefined rules. By using IDSs the traffic before and after the firewall is monitored to check whether the firewall enforces the rules or not. IDSs on the two sides of the firewall give opportunity to get information about the incoming and outgoing traffic (traffic in two ways). This information is used to monitor the operation of the firewall.

The firewall and IDSs (sensors) do not have any connection. Those two IDSs that are monitoring network traffic have two interfaces sniffing interface (stealth interface) and management interface. Stealth interfaces allow an IDS to be invisible to other hosts on the network while still being able to detect attacks. They have no TCP/IP stack. Where a packet needs to be transmitted through the interface the packet would need to be crafted (WEB_10 2005). IDSs also have a management interface through which it can be managed. This interface belongs to monitoring network composed of IDSs and IDS Center. The rules of IDSs are sent by IDS Center, when configuration files changes (it means the policy of firewall changed) it is sent to the IDSs. The sensors send their detection results to the IDS Center. This detection results are stored in a database and used to monitor firewall operation.

As it is sated both incoming and outgoing traffic can be detected. It means firewall policies for incoming and outgoing traffic are monitored. The rules defined on firewall are written on both IDSs as well. IDSs monitor the traffic before it reaches the firewall and after it is passed by the firewall.

For example if the incoming FTP traffic is blocked by the firewall. The expected situation is that, if there is incoming FTP traffic between the external network and the firewall, and it is detected by the first IDS (IDS between the external network and firewall), it should be blocked by the firewall and the second IDS (IDS between the firewall and internal network) should not detect any incoming FTP traffic. The following algorithm explains how the detection results of sensors are used to monitor the firewall. Monitoring algorithm is explained inTable1.

IDS1: IDS between the Internet and firewall

IDS2: IDS between the firewall and local area

		IF		THEN
		IDS1	IDS2	Firewall
Incoming Traffic	DROP	alert	no alert	operate properly
		alert	alert	**problem**
	ACCEPT	alert	alert	operate properly
		alert	no alert	**problem**
Outgoing Traffic	DROP	no alert	alert	operate properly
		alert	alert	**problem**
	ACCEPT	alert	alert	operate properly
		no alert	alert	**problem**

Table 5.1. Monitoring algorithm.

While making comparison there are three important properties to take into consideration. These are **alert time**, **package** and **rule**.

t1: alert time of IDS1

t2: alert time of IDS2

- For incoming traffic t1 should be smaller than t2, for outgoing traffic t1 should be greater than t2.
- Both for incoming and outgoing traffic the alert should be raised by the same rule and for the same package.
- To identify the rule **sid** keyword is used. It is used to add a "Snort ID" to rules. All numbers above 1,000,000 can be used for local rules (Rafeeq 2003).

  alert ip any any -> any any (sid: 1000001;)

- To identify the package the IP header information of the package is used.

Locating sensors as they can monitor the traffic before it is filtered and after passed by the firewall accomplishes these goals:

- Allow you to verify that the firewall enforces its rules for incoming and outgoing traffic.
- Knowing of any attempts that are being made before any packet filtering is done (Pre-firewall – External)
- Knowing that an attempt was successful or blocked by the firewall (Post-Firewall – Internal)
- Detecting attacks originating from internal network, and verifying the configuration of firewall.

The proposed architecture is built in IYTE Department of Computer Engineering IS3 Laboratory. The firewall and IDS work on Linux Red Hat 9.0 operating system. There are three networks on this experimental environment; one simulates internal network, one external network and the other monitoring network composed of IDS center and IDS agents. The architecture in the laboratory contains six PCs, two Cisco Catalyst 2950 switches and one 3com switch. These PCs and switches are connected with UTP cables.

One PC for firewall, one for IDS between external network and firewall, one for IDS between firewall and internal network, one for IDS center that collects the detection results of sensors, one for external network and one for internal network. Operating system of five computers is Linux Red Hat 9.0, only the PC for internal network has Microsoft Windows 2000 operating system.

Figure 5.2 shows the lifecycle of a network packet. The network packet coming from external network to the internal network is copied by the switch between the external network and firewall. The first copy of the network packet (CP1) goes to the IDS between the external network and firewall and detected according to the rules written on the IDS, detection results (DR) are sent to the IDS Center.

The network packet (P) reaches the firewall. There is two probabilities, the firewall may pass or drop the network packet. If it is dropped the lifecycle ends. If it is passed, switch between the firewall and internal network creates second copy of the network packet (CP2). The copy of the packet goes to the IDS between the firewall and internal network and detected according to the rules written on the IDS, detection results (DR) are sent to the IDS Center. at the end the network packet reaches its destination and completes its lifecycle.

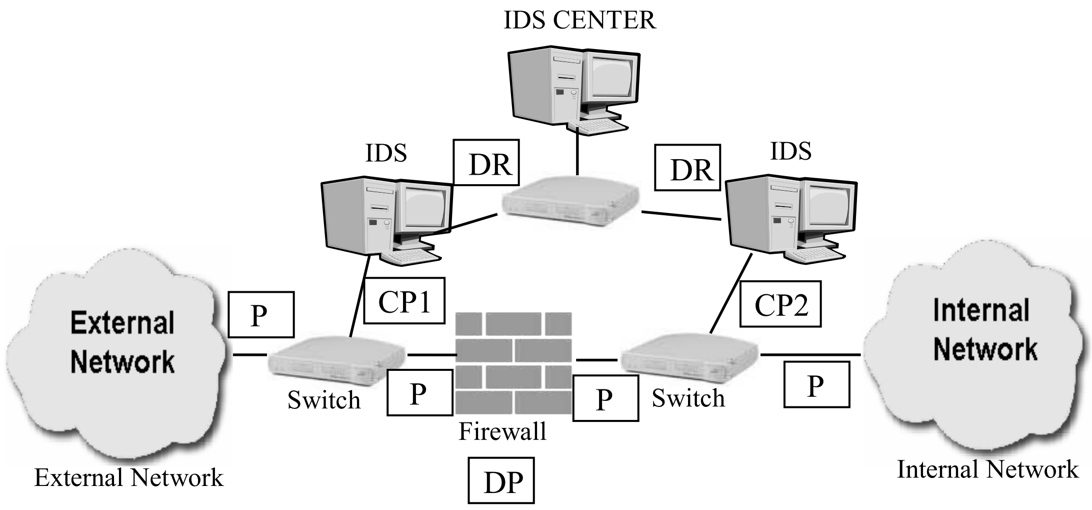

Figure 5.2. Packet flow.

Network IDSs have great difficulty in promiscuously monitoring switched networks. This was overcome by configuring a switch to replicate the data from all ports or onto a single port. This function has a multitude of names including; Port Mirroring, Monitoring Port, Spanning Port, SPAN port and Link Mode port. Generally Port Mirroring usually indicates the ability to copy the traffic from a single port to a mirror port but disallows any type of bidirectional traffic on the port (WEB_9 2005).

In the architecture one interface of the switch is source and the other is destination for Monitoring Port. The source interface is the interface of the firewall and the destination is the port to which the IDS agent (sniffing interface of IDS) is attached. To copy all the traffic coming to the firewall interface onto another interface the following configuration should be made.

Switch(config)# monitor session 1 source interface **interface number** both

Switch(config)# monitor session 1 destination interface **interface number**

Sniffing interfaces of the sensors is configured in stealth mode to make them invisible to external and internal networks. IP address is not assigned to the interfaces and the following command is used.

ifconfig eth1 up

The firewall used in architecture is IpTables. Computer on which the firewall operates has two network cards; one is belonging to external network and the other is belonging to internal network. Ip addresses of external, internal network and monitoring network and configuration of these network cards is as follows:

external network: 192.168.1.0/24

internal network: 192.168.0.0/24

monitoring network: 192.168.5.0/24

Eth0 (internal network)

ip: 192.168.0.254

netmask : 255.255.255.0

default gateway: 192.168.1.254

Eth1 (external network)

ip: 192.168.1.254

netmask : 255.255.255.0

default gateway: 192.168.0.254

To communicate external and internal network the following changes should be made on the /sbin/iptabels file.

1. Public and private network definitions:

   External network definition

   INET_IP="192.168.1.254"

   INET_IFACE="eth1"

   INET_BROADCAST="192.168.1.255"

   Internal network definition

   INET_IP="192.168.0.254"

   INET_IFACE="eth0"

   INET_BROADCAST="192.168.0.255"

2. Enable ip forwarding.

   echo "1" > /proc/sys/net/ipv4/ip_forward

3. Network address translation (NAT).

$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP

The IDS used in architecture is Snort. There are two Snort sensors and an Snort center where sensors send their logs. Following phase explains the installation of monitoring network. The IP addresses of Snort sensors and Snort Center are:

sensor1: 192.168.5.1/255.255.255.0

sensor1: 192.168.5.2/255.255.255.0

center: 192.168.5.5/255.255.255.0

Phase I – MySQL/SnortCenter/ACID server

- Apache Installation
- MySQL Database Installation
- ACID Console Installation
- SnortCenter Console Installation

Phase II - Snort sensor(s) installation

- Snort Sensor Installation
- SnortCenter Agent Installation

Phase III - SnortCenter completion

- Add sensors to the SnortCenter Console

5.1 Snort Center Installation

The operating system of IDS center is Linux Red Hat 9.0. Apache Web Server, MySQL Database, ACID Console and SnortCenter Console need to be installed.

5.1.1 Apache Web server installation and required configurations.

Install "httpd" package from Red Hat 9.0 installation CD, there may be dependency problems with different versions.

The next step is to setup Apache so that the console websites use SSL.

```
# rpm –ivh mod_ssl-2.*.*-*.i386.rpm
# chkconfig --level 2345 httpd on
# /etc/rc.d/init.d/httpd start
```

Remove the fake key and certificate that were generated during the installation with the following commands:

```
# cd /etc/httpd/conf
# rm ssl.key/server.key
# rm ssl.crt/server.crt
```

Next, you need to create your own random key. Type in the following command:

```
# make genkey
```

Then the system expects a password. This password will be asked every time web servers starts.
Now make testcert

```
# make testcert
```

The password will be asked, after entering password correctly additional information that will be requested.

After providing the correct information, a self-signed certificate will be created and placed in /etc/httpd/conf/ssl.crt/server.crt.

Edit the httpd.conf file and make the following changes:

```
# vi /etc/httpd/conf/httpd.conf
```

Look for the below info in the httpd.conf file (NOTE: there are multiple HAVE_SSL entries. Find the exact one that is listed):

```
Listen 80
```

Change to look like this:

```
#Listen 80
```

You will need to restart your Apache server.

```
# service httpd restart
```

Again the password will be requested.

5.1.2 MySql Database installation and required configurations.

Install MySql server from Red Hat 9.0 installation CD in case of dependency problems.

```
# rpm -ivh MySQL-client-4.0.X-X.i386.rpm
# rpm -ivh MySQL-shared-3.23.X-X.i386.rpm
# mysql -u root
mysql> set password for ‘root’@’localhost’ = password(‘yourpassword’);
mysql> create database snort;
mysql>exit
```

Note that after setting the root password above you need to login using a password to access the database with root. E.g. # mysql –u root –p

MySql doesn't start the mysql service on run level 3. Do the following to correct the problem.

# chkconfig --level 3 mysql on

The database tables need to be set up. This can be accomplished by running the create_mysql script. This can be is included in the snort-2.0 archive. If the file is not located in the directory from which the mysql program was run from, add the path to the source statement. E.g. **mysql> source /home/john/create_mysql**

# mysql -u root -p

mysql> connect snort

mysql> source create_mysql

mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort;

So you can connect locally with this account

mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort@localhost;

Creates a user that cannot delete alerts from database: may only need the local account

mysql> grant CREATE, INSERT, SELECT, UPDATE on snort.* to acidviewer;

So you can connect locally with this account

mysql> grant CREATE, INSERT, SELECT, UPDATE on snort.* to acidviewer@localhost;

Set the passwords for the MySQL accounts.

mysql> connect mysql

mysql> set password for 'snort'@'localhost' = password('yourpassword');

mysql> set password for 'snort'@'%' = password('yourpassword');

mysql> set password for 'acidviewer'@'localhost' = password('yourpassword');

mysql> set password for 'acidviewer'@'%' = password('yourpassword');

mysql> flush privileges;

mysql> exit

ACID requires the installation of PHP and the supporting Mysql module. Install them from Red Hat 9.0 installation CD.

5.1.3 ACID Console Installation

To install Acid Console ACID 0.9.6B23, ADODB v3.40, JPgraph v1.11, GD v2.0.12 packages should be installed.

Once the files have been downloaded untar the following files to /var/www/html.

# tar -zxvf acid* -C /var/www/html

# tar -zxvf adodb* -C /var/www/html

# tar -zxvf gd* -C /var/www/html

# tar -zxvf jpgraph* -C /var/www/html

# cd /var/www/html

# mv gd-2.0.11 gd

# mv jpgraph-1.11 jpgraph

Configure the ACID configuration file:

# cd /var/www/html/acid

# vi acid_conf.php

Once you're in the acid_conf.php file modify the following variables. Change the xxxx to reflect the password you have chosen for the snort account.

```
$DBlib_path=”../adodb”;
$alert_dbname=”snort”;
$alert_user=”snort”;
$alert_password=”xxxx”;
$ChartLib_path=”../jpgraph/src”;
```

Next setup the view only ACID portal (NO deleting of events). This is good for people who only need to view alerts. Copy the /var/www/html/acid to /var/www/html/acidviewer (view only acid).

# cp –R /var/www/html/acid /var/www/html/acidviewer

# cd /var/www/html/acidviewer

# vi acid_conf.php

Change the following variables in /var/html/www/acidviewer/acid_conf.php. Again, Change the xxxx to reflect the password you've chosen for the acidviewer account.

```
$alert_user=”acidviewer”;
$alert_password=”xxxx”;
```

Now both of the ACID websites are secured with Apache. Setup the two accounts for accessing the ACID website. When prompted enter your password for that web account. Be careful not to include the –c option in the third line!

# mkdir /usr/lib/apache

# htpasswd -c /usr/lib/apache/passwords admin

# htpasswd /usr/lib/apache/passwords acidviewer

# cd /usr/lib/apache

# chown apache passwords

# chmod 400 passwords

Add the following lines to /etc/httpd/conf/httpd.conf in the DIRECTORY section. Section means the general area when you see the other Directory formats.

```
<Directory “/var/www/html/acid”>
        AuthType Basic
        AuthName “yourcompany”
        AuthUserFile /usr/lib/apache/passwords
        Require user admin
        AllowOverride None
</Directory>
<Directory “/var/www/html/acidviewer”>
        AuthType Basic
        AuthName “yourcompany”
        AuthUserFile /usr/lib/apache/passwords
        Require user acidviewer
        AllowOverride None
</Directory>
```

Restart the httpd service.

# service httpd restart

5.1.4 SnortCenter Console Installation

First create the SnortCenter database and a database user:

# mysql –u root –p

mysql> create database snortcenter;

mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snortcenter.* to snortcenter@localhost;

mysql> set password for 'snortcenter'@'localhost' = password('yourpassword');

mysql> flush privileges;

mysql> exit

Download and install the SnortCenter console.

# tar -zxvf snortcenter* -C /var/www/html

# cd /var/www/html

# mv snortcenter-beta snortcenter

# cd snortcenter

# vi config.php

Edit the following lines in config.php. The $DB_password should be the root password on the database and the $hidden_key_num should just be a random number (its used in the authentication system to encrypt a value in the cookie).

```
$DBlib_path = “../adodb”
$DB_user = “snortcenter”
$DB_password=”XXXX”
$hidden_key_num = “XXXXXXX”
```

5.2 Snort Sensor Installation

The first thing we need to do is install the MySQL dependences for snort. Install MySQL-client from Red Hat 9.0 installation cd, then install the libraries.

```
# rpm –ivh MySQL-devel-*.**.**-*.rpm
```

Next download compile and install snort.

```
# cp snort-2.0.*.tar.gz /usr/src/redhat/SOURCES
# cd /usr/src/redhat/SOURCES
# tar –zxvf snort-2.0.*.tar.gz
# cd /usr/src/redhat/SOURCES/snort-2.0.*
# ./configure --with-mysql
# make
# make install
```

Create a directory for your snort configuration files:

```
# mkdir /etc/snort
```

Create the logging directory for snort. Port scan information is put here. Also, if you're doing packet logging or are not populating a database, then that information is placed here as well.

```
# mkdir /var/log/snort
```

Snort Center Agent Installation

```
# mkdir /opt/snortagent/
# cp snortcenter-agent-v0.1.6*.tar.gz /opt/snortagent
# cd /opt/snortagent
# tar -zxvf snortcenter-agent-v0.1.6*.tar.gz
# cd sensor
# ./setup.sh
```

You will then be prompted with a series of questions:

Config File Directory	= /etc/snort
Log File Directory	= /var/log/snort
Perl	= <ENTER>
Snort	= <ENTER>
Snort Rule Config File	= /etc/snort
Port	= <ENTER>
IP Address	= (Your sensors management IP (eth0))
Login Name	= <ENTER>
Password	= (Password that the consoles uses to access the sensor)
Confirm Password	= (Same as above)
Host	= <ENTER>
SSL	= Y
Allow IP	= (This is the IP address of you SnortCenter Server)
Start on Boot	= Y

5.3 Snort Center Completion

Local users are member of internal network they are behind the firewall. They connect external network through the firewall. External users are not member of our (internal) network. All the users connected to internal network through the firewall according to defined rules are external users. The following servers are in the internal network: Web Server, FTP Server, DNS Server, Mail Server.

The firewall has the following rules to enforce:

- WWW, DNS and relay Mail CAN be accessed by local and external users.
- Local users CAN use the Internet email and Internet browsing.
- The FTP server is used by local users ONLY.
- Local users SHOULD NOT use news, telnet, or rlogin services to external hosts.
- Local users CAN use Internet FTP to download files.

The rules are defined on a chain called "policy".

- WWW, DNS and relay Mail can be accessed by local and external users.
```
$IPTABLES -A policy -d WebServer -p tcp -i eth1 --dport 80 -j ACCEPT
$IPTABLES -A policy -d WebServer -p udp -i eth1 --dport 80 -j ACCEPT
$IPTABLES -A policy -d DNSServer -p udp -i eth1 --dport 53 -j ACCEPT
$IPTABLES -A policy -d MailServer -p tcp -i eth1 --dport 25 -j ACCEPT
```
- Local users CAN use the Internet email and Internet browsing.
```
$IPTABLES -A policy -p tcp -i eth0 --dport 80 -j ACCEPT
$IPTABLES -A policy -p udp -i eth0 --dport 80 -j ACCEPT
```
- The FTP server is used by local users only.
```
$IPTABLES -A policy -d FTPServer-p udp -i eth1 --dport 21 -j DROP
```
- Local users SHOULD NOT use news, telnet, or rlogin services to external hosts.
```
$IPTABLES -A policy -p udp -i eth0 --dport 119 -j DROP
$IPTABLES -A policy -p tcp -i eth0 --dport 23 -j DROP
$IPTABLES -A policy -p udp -i eth0 --dport 111 -j DROP
```
- Local users CAN use Internet FTP to download files only.
```
$IPTABLES -A policy -p tcp -i eth0 --dport 21 -j ACCEPT
```

The rules defined on firewall are written on both IDSs as well. IDSs monitor the traffic between external network and firewall before it reaches the firewall, and between the firewall and internal network after it is passed by the firewall. Corresponding rules defined on Snort are as follows:

- WWW, DNS and relay Mail can be accessed by local and external users.
```
pass tcp any any-> WebServer 80
pass udp any any-> WebServer 80
pass udp any any-> DNSServer 53
pass tcp any any-> MailServer 25
```
- Local users CAN use the Internet email and Internet browsing.
```
pass tcp InternalNetwork any-> any 25
pass tcp InternalNetwork any-> any 80
pass udp InternalNetwork any-> any 80
```

- The FTP server is used by local users only.

```
alert udp any any-> FTPServer 21
```

- Local users SHOULD NOT use news, telnet, or rlogin services to external hosts.

```
alert udp InternalNetwork any-> any 119
alert tcp InternalNetwork any-> any 23
alert udp InternalNetwork any-> any 111
```

- Local users CAN use Internet FTP to download files.

```
pass udp InternalNetwork any-> any 21
```

CHAPTER 6

CONCLUSION

The objective of this thesis is to monitor firewall using the detection capability of intrusion detection systems. An architecture is proposed to monitor firewall. To accomplish this objective, first, firewalls are examined to determine when, how and why they are used and to state their importance to protect private networks from untrusted networks. Then, intrusion detection systems are examined to determine their abilities. Then, the importance of testing the firewalls, a methodology for testing and need for monitoring the firewall and an architecture to do this is presented. Therefore, it is clear, we must be sure that the firewall operates properly and enforces the previously defined rules. If firewall not operates correctly, that would be disastrous for the systems, supposed to be protected by the firewall should be detected as fast as possible. To be able to do that, it is a good idea to integrate the isolation function of the firewall with the detection function of the IDS. Two IDSs, one between the private network and firewall and the other between the firewall and Internet, gives us the opportunity to be able to monitor firewall rules for both incoming and outgoing traffic.

REFERENCES

Andreasson, O., 2001. “Iptables Tutorial 1.1.9”.

Axelsson, S. 1999. “The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection”. In Proceedings of the 6th ACM Conference on Computer and Communications Security. pp. 1-7. November 2-4, 1999.

Bace, R. and Mell, P., 2001. “Intrusion Detection Systems”. NIST Special Publication.

Bace, R., 2000. “Intrusion Detection”. Macmillan Technical Publishing.

Carter, E., 15.02.2002, “Intrusion Detection Systems”, http://www.informit.com/.

Chapman, D. Brent, Zwicky, Elizabeth D. 1995, *Building Internet Firewalls*, (O’RIELLY), ISBN 1-56592-124-0, First Edition, November 1995.

CNAPC 2005, “The Cisco Certified Network Associate Curriculum”, http://std.cnap.ege.edu.tr

Debar, H., Dacier, M., Wespi, A., 1999. “Towards a taxonomy of intrusion-detection systems”, *Computer Networks*. 31 1999 805–822

Dworakowski W., 2004. “Why is a firewall alone not enough? What are IDSes and why are they worth having?”, http://www.windowsecurity.com/articles/Why_is_a_firewall_alone_not_enough_What_are_IDSes_and_why_are_they_worth_having.html, 23 Jul 2004.

Enterasys 2001, “Network Address Translation”, Enterasys Networks, http://www.enteresys.com

Gómez, D. G., 2003. “Sistemas de Detección de Intrusiones: Capítulo 4”. July 2003. http://www.dggomez.arrakis.es/secinf/ids/html/cap01.htm

Haeni , Reto E., 1997. “Firewall Penetration Testing”, The George Washington University Cyberspace Policy Institute

ISS 1998, Internet Security Systems, October 21998 “Network- vs. Host-based Intrusion Detection A Guide to Intrusion Detection Technology”

Jones Anita K. and Sielken Robert S., 2000 “Computer System Intrusion Detection: A Survey”, Department of Computer Science University of Virginia

NA 2003. Network Associates, “Intrusion Prevention: Myths, Challenges, and Requirements”, April 2003. (http://www.networkassociates.com/)

Netgear 2004, “Security & Savings with Virtual Private Networks”, NETGEAR

NIST 2002. National Institute of Standards and Technology (NIST). “Guidelines on Firewalls and Firewall Policy”. January 2002

OneSecure 2001. OneSecure “Intrusion Detection and Prevention Protecting Your Network From Attacks Allowed By The Firewall”

Rafeeq, R. Ur., 2003. *Intrusion Detection Systems with Snor*t. Prentice Hall PTR. ISBN 0-13-140733-3. 2003

Schultz , E. Eugene., 1996. “*How to Perform Effective Firewall Testing*”, Computer Security Journal, Vol. XII, No. 1, Spring 1996

Shay, W. A., 2000. “Firewall”, University of Wisconsin-Green Bay. 2000

Snort 2004. Snort.org. January 2004. (http://www.snort.org/)

Stalling, W., 2002. *Network Security Essentials Applications and Standards*, (Prentice Hall), pp. 345

TopLayerNetworks 2002. Top Layer Networks. “Beyond IDS: Essentials of Network Intrusion Prevention”. November 2002

Wack, J. P., Carnahan, L. J. 1995, “*Keeping Your Site Comfortably Secure: An Introduction to Internet Firewalls*”, NIST Special Publication 800-100 U.S. Department of Commerce

WEB_1 2005. Check Point Software Technologies Ltd. “Stateful Firewall Technology Products and Solutions”, 08/11/2005. (http://www.checkpoint.com/products/technology)

WEB_2 2004. “Network Security Chapter 13 Internet Firewalls”, 12/10/2004. http://www.tkn.tu-berlin.de/curricula/NetworkSecurity/Handouts/13_Firewalls.pdf

WEB_3 2005. “TCP/IP Protocol Suite”, 05/04/2005. http://burks.bton.ac.uk/burks/pcinfo/hardware/ethernet/tcpip.htm

WEB_4 2005. “An Introduction to Network Firewalls and the Firewall Selection Process”, 20/11/2005. http://www.more.net/technical/netserv/tcpip/firewalls/

WEB_5 2004. “How to use IP Tables”, 15/08/2004. http://krnlpanic.com/

WEB_6 2004. Russell Rusty, 24.01.2002, “Linux 2.4 Packet Filtering HOWTO”, 01/10/2004. http://www.iptables.org/documentation/HOWTO//packet-filtering-HOWTO-7.html

WEB_7 2002. “Intrusion Detection System”. 29.10.2002. (http://www.webopedia.com/tem/I/intrusion_detection_system.html)

WEB_8 2005. P. Rushing, “iptables fontend?”, 30/03/2005. “http://www.sblug.org/pipermail/sblug-list/2002-March/006157.html”, March 2002

WEB_9 2005. “Catalyst 2950 Desktop Switch Command Reference”, 20/06/2005. http://www.cisco.com/

WEB_10 2005. “Intrusion Detection Terminology (Part Two),20/08/2005. http://www.infosecurity.org.cn/content/ids/intru_detec_ter_2.htm