Parallel & Secure Embedded System Approach for Complex Topological & Large Load Operations in Telecommunication Networks (PSECTLOT)

Developed by

Qaisar Javaid

Ahmed Salman Mirza

Supervised by

Prof. Dr. Khalid Rashid

Dr. S. Tauseef ur- Rehman

Faculty of Applied Sciences
International Islamic University, Islamabad
(2005)

International Islamic University, Islamabad
Faculty of Applied Sciences
Department of Computer Science

Dated: 18-07-2005

Final Approval

It is certified that we have read the project report submitted by Ahmed Salman Mirza 69-CS/MS/02 and Qaisar Javaid 77-CS/MS/02, and it is our judgment that this project is of sufficient standard to warrant its acceptance by The International Islamic University, Islamabad for MS degree in Computer Science.

COMMITTEE

External Examiner

Dr. Abdus Sattar
House No.143, Street No.60
Sector I-8/3,
Islamabad.

Internal Examiner

Dr. Sikandar Hayat Khiyal
Head, Department of Computer Sciences,
International Islamic University,
Islamabad

Supervisor(s)

Prof. Dr. Khalid Rashid
Dean, Faculty of Applied Sciences
International Islamic University,
Islamabad.

Dr. S. Tauseef Ur-Rehman
Head, Department of Telecommunication,
International Islamic University,
Islamabad.

Dedication

The thesis "Parallel & Secure Embedded System Approach for Complex Topological & Large Load Operations in Telecommunication Networks" is dedicated to our Creator, without His will, nothing is possible. Secondly, we dedicate our project to the Prophet Muhammad (S.A.W) and then to the whole Muslim Ummah. Next, we dedicate this project to our beloved Parents, who stood by us and helped us in the time of need. Lastly, we dedicate the project to Prof. Dr. Khalid Rashid & Dr. S. Tauseef ur-Rehman, who helped us in all possible ways and guided us through the whole process as a Beacon House.

A dissertation submitted to the

DEPARTMENT OF COMPUTER SCIENCE,

INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD

as a partial fulfillment of the requirements for the award of the

Degree of MS in Computer Science (Networking)

Declaration

We hereby declare that this software, neither as a whole nor as a part thereof has been copied out from any source. If found to be copied from any other source, we shall bear the consequences. It is further declared that we have developed this software entirely on the basis of our personal efforts made under the sincere guidance of our teachers and the supervisor. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute.

Ahmed Salman Mirza
Reg No. 69-CS/MS/02
Qaisar Javaid
Reg No. 77-CS/MS/02

Acknowledgement

All praise to the Allah Almighty, the most merciful, the most gracious, with His Help and blessing, we were able to complete the project. Thanks to our project supervisors Prof. Dr. Khalid Rashid and Dr. S. Tauseef ur-Rehman whose sincere efforts helped us to complete our project successfully. Our immense gratitude goes to Prof. Dr. Khalid Rashid, Dean, Faculty of Applied Sciences, International Islamic University, Islamabad for his valuable and expert guidance, scholarly criticism and sympathetic attitude towards the completion of our studies.

Finally our deepest sense of acknowledgement to our parents who helped us during our most difficult times and it is due to their untiring efforts that we are at this position.

Ahmed Salman Mirza
Reg. No. 69-CS/MS/02

Qaisar Javaid
Reg. No. 77-CS/MS/02

Project in Brief

Project Title:	Parallel & Secure Embedded System Approach for Complex Topological & Large Load Operations in Telecommunication Networks
Objective:	To build a Network based Firewall
Undertaken By:	Ahmed Salman Mirza Qaisar Javaid
Supervised By:	Prof. Dr. Khalid Rashid Dean, Faculty of Applied Scinces International Islamic University, Islamabad Dr. S. Tauseef Ur-Rehman Head, Department of Telecommunication, International Islamic University, Islamabad
Date Started:	1st February, 2004
Date Completed	May 2005
Technologies Used:	VC 6 MS Access
System Used:	Pentium 4

Abstract

This thesis presents the design architecture of distributed security cluster of computers to be saved from the malicious attacks of intruders. Intrusion detection has traditionally been performed at the operating system (OS) level by comparing expected and observed system resource usage. OS intrusion detection systems (OS IDS) can only detect intruders, internal or external, who perform specific system actions in a specific sequence or those intruders whose behavior pattern statistically varies from a norm. It describes the new way of representing the *Sniffer Firewall* on the network hosts, which would enable the safety of all the nodes, which are present on the network along with the safety of the whole cluster. This approach of distributed security enables the network to be safe from all sorts of attacks may it be from inside or outside of the network.

Abbreviations

DES (Data Encryption Standard)

DLL (Dynamic Link Library)

DNS (Domain Name System)

FTP (File Transfer Protocol)

HTML (HyperText Markup Language)

HTTP (HyperText Transfer Protocol)

HTTPS (Secure HTTP)

ICMP (Internet Control Message Protocol)

IDS (Intrusion Detection System)

LAN (local area network)

MAC (Machine Authentication Code)

PPP (Point-to-Point Protocol)

RFC (Request for Comments)

SMTP (Simple Mail Transfer Protocol)

TCP (Transmission Control Protocol)

TCP (Transmission Control Protocol)

UDP (User Datagram Protocol)

VPN (Virtual Private Network)

WAN (Wide Area Network)

Table of Contents

Table of Contents

2 Design

3 Implementation

Chapter No.1
Introduction

1. Introduction

Security is a critical issue in reliable network computing. The security of a network is at stake when the network is exposed to people through intranet. Such networks can easily be affected by or destroyed by the unauthorized intruders. Private networks use enclosed network to protect them selves from the intruders. Single gateway firewalls are use to repel malicious attacks. In such gateways all the network nodes are considered as trusted where as it distrusts all the external hosts. Computer security has become a head of the news topic. While computer systems are invading every corner of our lives and getting complex and hard to secure, the knowledge of how to exploit their vulnerabilities is simultaneously reaching more and more users. After about 15 years of development, IDSs are now starting to be available on a large scale. Yet, there remain a few obstacles that current IDS technologies are still unable to address. Future of IDSs will mainly depend on their ability to find solutions to these challenges.

The purpose of this thesis is to analyze these challenges and define some guidelines for building the next generation of IDS. We will then suggest architecture for such IDS thereon we give the implementation and finally we present results and future work. The first part consists in an overview of practical computer security, in order to understand the context in which IDSs are used. We will then study the technologies currently used in common IDSs, while discussing their relative efficiency. Once the inner-working of IDSs explained, we will focus on the challenges that IDSs are facing, in order to identify some requirements for a long term competitive IDS and make suggestions on how to implement it. As the networks increase over the time period, their exposure to external threats also increases. The intruders would always have a new way to attack the network; therefore new means for protection and prevention from external threats would be implemented.

1.1 Need for Network Security

With the passage of time, need for communication is increasing greatly. Data needs to be transferred from one place to another, but in a secure fashion. It is important

in any environment to know what type of threats we might be facing. Be aware of any potential security holes in the system, and take care to prevent attacks against these. Many technologies and theories are implemented to provide protection to the data flowing from intruders, hackers and unauthorized personnel's. In today's booming e-commerce economy age, virtually every business, including the 'brick and mortal', is connected to compete for market share in the cyberspace. Enterprise's networked systems are inevitably exposed to the increasing threats from external hackers as well as from internal hackers. The consequences can be loss or modification of critical business data, disruption of services (availability), compromise of proprietary business plans or processes (confidentiality and integrity).

To counter these threats, many methods, tools and technologies have been deployed. Methods like

- Implementing policies and procedure
- User awareness
- Deploying firewall and intrusion detection and prevention systems
- Control systems access
- Computer incident handling teams

Battle is going on between intruders and defenders. To obtain maximum security one has to deploy all security layers, to cover maximum variety of threats.

1.2 Practical Network Security

Computers tend nowadays to be organized in large networks connected to Internet, and are used in an increasing number of sensitive applications, such as online banking and shopping, control of hardware installations and public infrastructures. Meanwhile, as more and more critical tasks are delegated to computers, computer security becomes a matter of first importance; hence the interest of understanding what

everyday practical computer security consists of, which is the purpose of this chapter. But let us first look at a more theoretical point of view.

1.3 Literature Review

Computer networks, including the world wide Internet, have grown in both size and complexity. The services they offer make them the main source to exchange data and an optimal environment for e-businesses. Unfortunately, they have also become the means to attack hosts and legitimate users. The growing importance of network security is shifting security concerns towards the network itself rather than being host-based. Security systems will soon evolve into network-based and distributed approaches to deal with heterogeneous platform technologies and support scalable solutions. Among all security issues, intrusion is one of the most critical and widespread. Intrusion can be defined as an attempt to compromise, or otherwise cause harm, to a network. Intrusion detection involves the act of detecting unauthorized and malicious access one or more computers. In addition to identifying attacks, the IDS can be used to identify security vulnerabilities and weaknesses, enforce security policies, and provide further system auditing by exploiting the logs/alerts from the output component of the IDS.

The first work performed on intrusion detection is commonly considered to be the one reported by Anderson, which introduced, among others, the idea of doing anomaly detection by creating profiles of normal use and detecting deviations from those profiles [1]. This idea was later formally presented by Denning in what is considered to be the seminal paper for modern intrusion detection [2].

In the area of host-based intrusion detection there has been substantial work using different methods for analyzing data generated by the host. One of the first host-based intrusion detection systems implemented was IDES [3,4,5,6], which used both a statistical detection engine based on Denning's model and a rule based expert system for detecting known intrusions by their signatures [2,7]. Kumar used pattern-matching techniques to detect and classify attacks [8]. More recently, Forrest et al. have applied

classification techniques to sequences of Unix system calls to identify anomalous behavior in its processes [9]. Also, Lane and Brodley have used classification of command sequences to perform automatic user identification [10]. Lunt has surveyed the most common host-based intrusion detection and audit trail analysis techniques [11,12].

Network-based intrusion detection includes a good amount of work. One of the first implemented network-based intrusion detection systems was the Network Security Monitor (NSM), which applies both statistical models and rule-based detection to network connections, using the traffic source, destination and service as relevant features for classification [13]. NADIR is an interesting system because it has been in use in a production environment for several years is, deployed at the Los Alamos National Laboratory, and which also uses both statistical and rule-based methods to analyze data collected at a number of different places within LANL's network [14]. Mukherjee, et al. made a survey on, some existing network-based intrusion detection systems were done [15].

In distributed intrusion detection data is both collected and analyzed in a distributed fashion. One of its earliest exponents is DIDS, which uses some NSM components and has the ability to do both local and global analysis of the data [16,17]. At the local level it uses both statistical and rule-based detection, and at the global level it uses a rule-based expert system. In this sense, DIDS can be described as a number of host-based and network-based intrusion detection systems that can communicate and share results with one another. This is the form of most intrusion detection systems that call themselves distributed: the same techniques used in host based and network-based intrusion detection systems are used, but the results are shared and can be analyzed at different levels. Other examples of distributed intrusion detection systems are GrIDS, ASAX, EMERALD, AAFID and a system proposed by Barnett and Vu [18-24]. All of these systems use different sources for data and different mechanisms for analyzing it. However, they share a similar general structure: a hierarchical arrangement where host-local components perform some part of the work and relay their results to components higher in the hierarchy. This continues until the partial results reach the top-level

components, which have a network-wide view of the systems. Because all these systems ultimately depend on a centralized component, we could argue that they are not truly distributed [25]. Furthermore, even these distributed intrusion detection systems operate based on heavy" host components, usually implemented as separate processes. These components, as those in all the intrusion detection systems previously mentioned, are subject to tampering and disabling by an intruder, and impose a continuous extra load on the system being monitored.

Not much work was done on signature based intrusion detection. So we opted to work on signature based intrusion detection in which there is a list of signatures that the IDS uses to compare against activity on the network or host. When a match is found, the IDS take some action, such as logging the event or sending an alarm to a management console.

1.4 Theoretical Computer Security

Security is a critical issue in reliable network computing. The security of a network is at stake when the network is exposed to people through intranet. Such networks can easily be affected by or destroyed by the unauthorized intruders. Private networks use enclosed network to protect them selves from the intruders. Single gateway firewalls are used to repel malicious attacks. In such gateways all the network nodes are considered as trusted where as it distrusts all the external hosts.

Computer security threats are usually classified in three categories referred to as *Confidentiality*, *Integrity* and *Availability* (CIA). *Confidentiality* concerns the protection of data from unauthorized disclosure, *Integrity* deals with unauthorized modification of data, and *Availability* ensures a reliable access to data. Confidentiality, Integrity and Availability are implemented in a computer system1 through a so-called 'security model'. From a theoretical point of view, every operation in a computer system, can be seen as a 'subject' accessing an 'object'. Both subject and object can be any kind of computer entity such as a process, a file or a hardware device, depending on the context.

The part of the computer system, managing these accesses, is called a Reference Monitor. The Reference Monitor allows access based on a sequence of access control rules. For these rules to be effective, the computer system should also provide the Reference Monitor with a reliable method to identify and authenticate subjects and objects. The way all these tasks are implemented defines a security model. Figure 1.1 shows a simplified Reference Monitor at work.

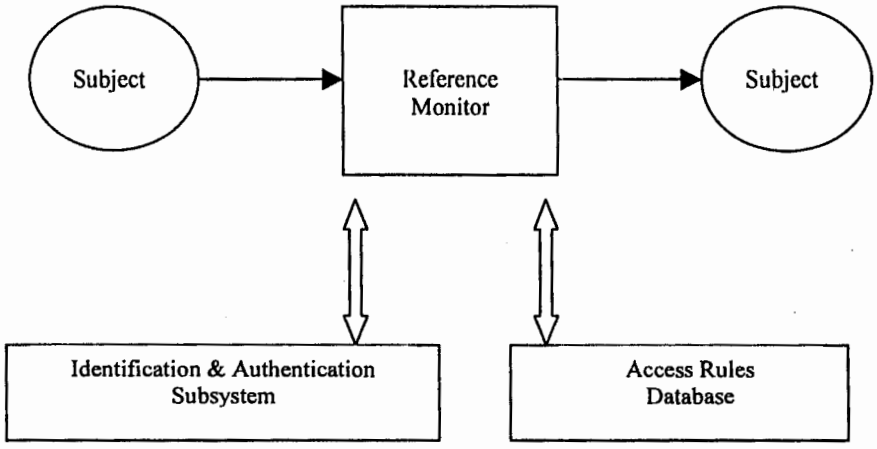

Figure 1.1 A Reference Monitor Regulates Subjects Accessing Objects

Theoretically attack is any activity aiming at breaking a security model. Interconnecting computers in networks opens the door to a wide panel of attacks against computer systems. With the advanced techniques used by the hackers, external attacks can easily immerse into the gateway and become a threat from inside.

1.5 Protocol

Once a reliable physical link is established between computers, data may be exchanged. Yet, since the link in itself is nothing more than an electrical cable, protocols have to be defined to enable reliable data exchange between computers, and a special architecture needs to be created to route data between computers. These protocols are often organized in layers, going from the physical link to the program application layer. Each of these layers contains protocols in charge of a particular aspect of the data

exchange: transferring data on a local segment, identifying each computer in a large network, enabling a reliable data exchange despite data losses, etc.

The most widespread network protocol stack is the TCP/IP protocol suite, which is used for Internet. It is made of 4 layers: link, network, and transport and application layer. The main protocols used in the TCP/IP stack are shown in figure 1.2.

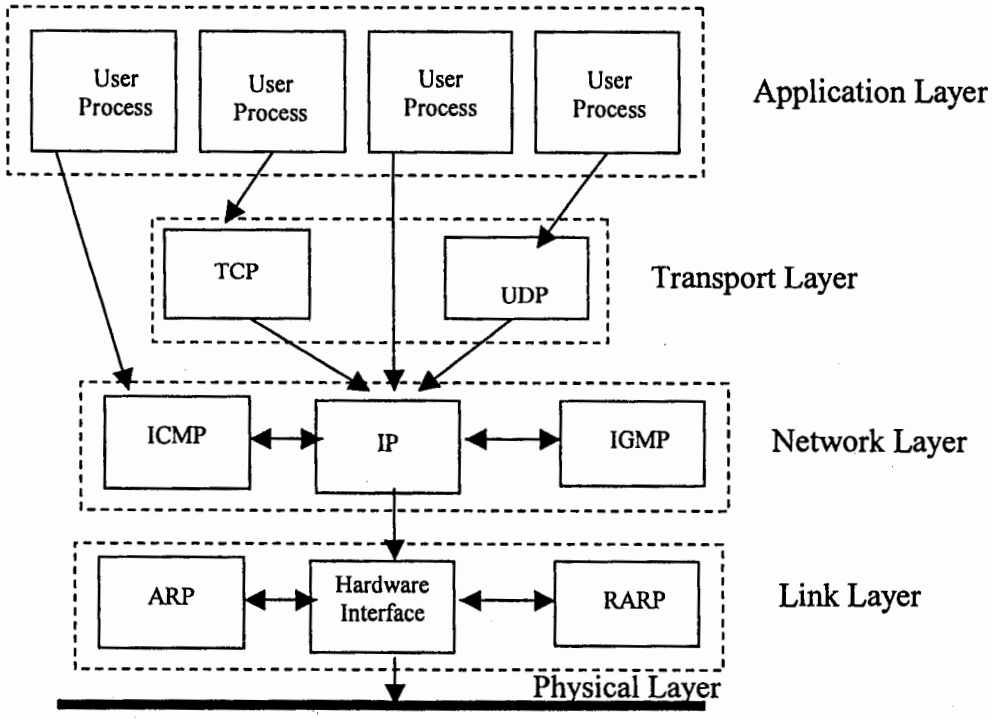

Figure 1.2 The TCP/IP Protocol Suite

1.6 Network Applications

As soon as some reliable mean of exchanging binary information is available to interconnected computers, doors are opening to an incredibly rich panel of applications. An endless list of services is now available on computer networks: World Wide Web (www), email, files sharing, online databases, streamed music and video, interfaces to other systems (telephone, hardware...), etc. Any program running on a computer connected to a Network has the possibility to send and receive data from other

computers. Most network applications are designed according to the client-server model in which a program called 'server' is running on a computer (also called server, by extension) and listening for incoming data sent by a 'client'. Figure 1.3 shows a typical state full client-server connection.

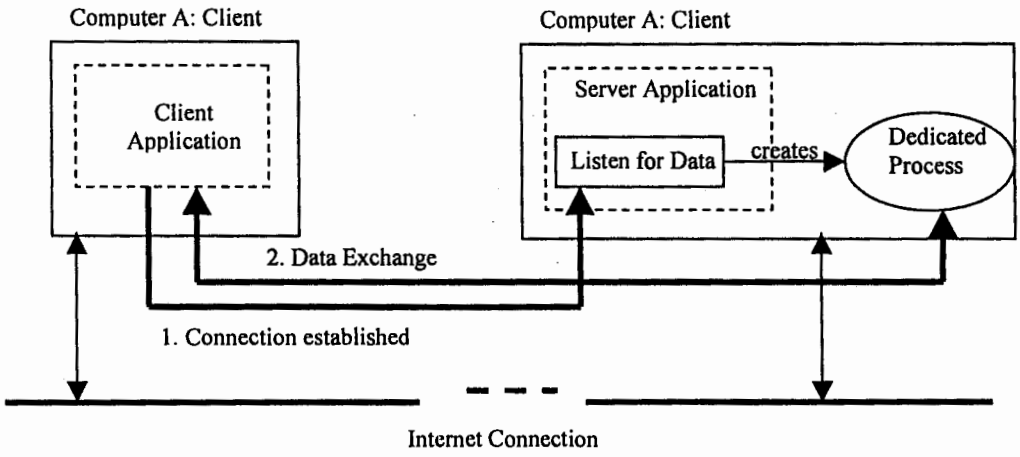

Figure 1.3 a State Full Client-Server Connection

1.7 Theoretical Analysis of Network Weaknesses

Computer networks have inherent weaknesses that expose them to security threats. These weaknesses can be listed

Sharing: Network resources (file system, devices, application servers...) are designed to be accessible from any point in a network, and are thus inherently exposed.

System complexity: A large network may interconnect different operating systems, use different network protocols and offer different services. Moreover, different people may administrate it, with different needs for security, different physical location or different languages. All this put together makes managing and securing a large network a challenging task.

Unknown perimeter: Considering the complexity of large networks, it gets sometimes hard to identify clear network boundaries as well as responsibilities for their administration.

Many points of attacks: Data traveling through a network can be accessed from many different locations. As well, networks may have multiple entry points, with in equal security levels.

Anonymity: Networks in themselves do not provide any mean of authentication. These have to be implemented in the protocols used. Intruders may thus benefit of partial or complete anonymity, or be able to masquerade themselves.

Unknown path: Data can often take many different paths to join two end points in a network. Network components, moreover, cannot keep track of all data passing through them during more than a short time. Tracing some network activity back to its sources may thus be technically challenging, not to mention problems related to multiple geographical locations, laws and languages.

1.8 Practical Causes for Network Vulnerabilities

There are many more possible attacks than those listed, and a real attack will usually combine many of them. The main practical causes for these vulnerabilities can be identified as:

- **Weak security model**. Protocols and applications are often designed without considering security as a main issue. Some examples resulting of weak security models are clear text protocols that can be sniffed, or bad password policy enabling password guessing, or trusted relations between computers, between which no authentication is required.
- **Inherent theoretical breach**. When a protocol or an algorithm has defects in its design itself. Examples of this are hijacking for the TCP protocol, or flawed cryptographic algorithms, such as WEP (Wireless Encryption Protocol).
- **Implementation breach**. An algorithm or protocol has been incorrectly implemented, thus presenting security vulnerabilities. Buffer overflows result from such programming mistakes, and are usually due to a lack of control on received arguments.

- **Configuration breach**. When an application, device or a system, is improperly configured, thus giving its user some higher privilege than required.
- **Lack of security awareness**. Computer users are usually unaware of what the security risks are. They will use weak passwords, install unsafe applications, and be vulnerable to social engineering.

1.9 Firewalls

Firewall is responsible for delimiting areas in computer networks and applying rules on how network packets circulate from one area to another. The firewall stands at the cross point between these areas. On a network layer level, it controls packet forwarding between network areas, based on configurable access rules. On a higher application level, it may control user sessions, do content filtering, etc.

1.9.1 Virus Scanners

Virus scanners aim at detecting 3 categories of dangerous program files circulating on computer networks: viruses, worms and Trojan horses. A virus is a self-reproducing program that is encapsulated in a host application, which serves as a mean of transport, thus propagating itself from computers to computers, as the host application is itself replicated. A worm is like a virus, but does not need a host application to provide transport. It uses instead some breaches in system applications to forward itself to another system and run there. Both viruses and worms, beside their replicating functionalities, may also attempt to violate the security model of a system by deleting files, spying on information, etc. Trojan horses, finally, are legitimate programs that have been modified to include some dangerous functionality, like those listed previously. Virus scanners will usually be located on workstations or network gateways to filter the content of received files.

1.10 Standard IDS Architecture

An IDS, whether it is host or network based, usually consists in 4 distinct successive layers: a *sensor*, running some *filters* that are generating an *alert-flow* directed to a *monitoring central*, as illustrated on figure 1.4. This general model can be used to describe host based IDSs, as well as network based IDSs or even Distributed IDSs (DIDSs). The sensor is responsible for gathering relevant data from a monitored system. The filters are small-specialized programs parsing the data provided by a sensor in order to detect possible attack patterns.

Suspect patterns trigger the filters into sending alerts. The alerts produced by all the filters are gathered into an alert-flow. This alert-flow is then directed to a monitoring central that it will reach after being filtered and preprocessed. On the monitoring central, an end-user monitors and interprets the alert- flow. This layered model is of interest to us since it successfully describes the most common IDS architectures. Some IDS are using more simple models, and some others are based on completely different architectures, inspired for example by the human immune system or by multi-agent technologies.

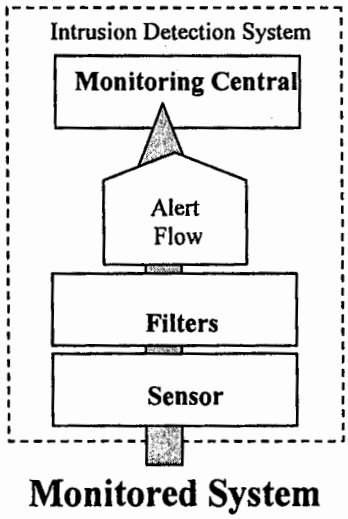

Figure 1.4 A 4 Layer Model For Standard IDS Architecture.

1.11 Distributed IDS Architecture

The previous model can be extended to Distributed IDSs, in which sensors, filters; alert-flow processing components and monitoring centrals can be located on different systems. A common alert-flow is then created by gathering alerts from multiple groups of sensors and filters. The previous 4-layer model can still describe such distributed architectures, as shown on figure 1.5.

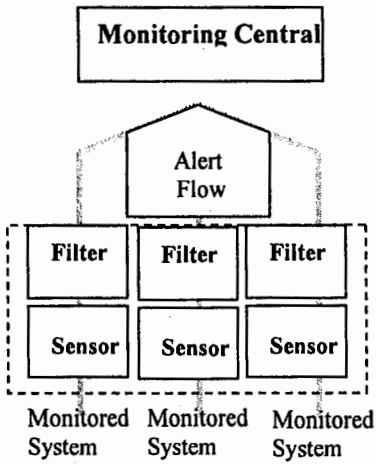

Figure 1.5 Distributed IDS architecture.

1.11.1 Signature Filter

A signature filter basically looks for one specific signature in the data flow provided by a sensor. An attack usually possesses some kind of signature that identifies it. This signature often consists in one or more specific binary pattern found in a given file or network packet. The signature can be described as a Boolean relation called rule. Although simple to implement, signature filters have a number of limitations. They are able to recognize an attack only when they 'know' a signature for this attack, and thus require continuous updates of their signature database as well as continuous research work to analyze new attacks and find their signatures.

Another limitation is that the process of discovering signatures is complex and not reliable. It relies on an individual having both detailed access to attack logs and the technical skills to analyze them, or who gets to learn how a new attack works. This person should then inform competent authorities (such as security focus) of his findings. These authorities then propagate the information among security professionals including those in charge of updating IDSs filters and signatures. Hence, signature filters will not detect attacks until they have been widely discovered, and probably widely used.

1.11.2 Alert Generation

Another issue in filter design is the generation of alert messages to acknowledge the anomaly or attack detected. This can be divided into two areas: defining a standard for an alert format, and increasing alert accuracy through internal filter programming techniques.

1.11.2.1 Alert Format

An efficient format for alerts is required in order to give accurate information to the end-user of the IDS as well as to enable alert manipulation by intermediate programs managing the alert-flow. There is currently no standard format for describing alerts, despite the many suggested candidates. Typical solutions involve using general description languages such as XML, or defining proprietary format based for example on key-value pairs. A difficulty is to define a standard flexible enough to be able to describe new attacks, but structured enough to enable automated analysis of its messages.

1.11.2.2 False-positives

With a theoretically perfect filter, each alert sent would exactly correspond to a real attack. In practice, a filter can often miss an attack or on the other hand send an alert while no attack is perpetrated. An alert of this second category is called false positive and

occurs typically when a filter interprets some legitimate activity as revealing an attack. Reducing the amount of false-positives is one of the main issues in IDS design.

1.11.2.3 Filter Design for Alert-flow Control

There are two criteria that a filter should match to provide efficient alert generation:

1. Avoid false-positives
2. One attack should not trigger a flood of alerts

These two measures aim at reducing the alert-flow and increasing alert relevancy. These requirements can be met by using filter components processing the alert-flow itself. Such a component can be added inside the filter, and works as follow: when the attack detection component of the filter generates an alert, it stores it temporarily in an alert log which is parsed and emptied periodically by an alert filter component, as shown on figure 1.6.

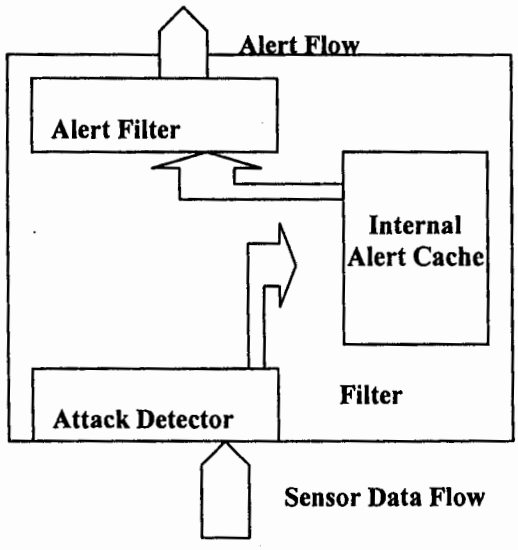

Figure 1.6 Filter's Internal Alert Processing Component

This component analyses the alert log and can:

- Forward relevant alerts
- Build a macro-alert summarizing multiple related alerts
- Apply alert level filtering rules to implement a higher level of attack analysis

1.12 Alert-flow

IDS filters generate an alert-flow, which is ultimately directed to an end-user, whether it is an individual or an automated response tool. For this end-user, monitoring and interpreting the alert-flow is a complex task that can be made simpler by inserting alert-flow preprocessing components between the filters and the end-user. Most actual IDSs offer no or very simple alert-flow preprocessing, and it is therefore a promising research field.

1.13 User Interface

In the simplest case of IDS, alerts are just stored in a log file and it is up to the end-user to analyze it properly. To facilitate this task for the end-user, most modern IDSs provide a (Graphical) User Interface to the alert-flow. The design of such an interface is an important issue, since its purpose is to help the end-user to analyze the raw alert-flow generated by the IDS. An in-depth analysis of IDS GUIs' requirements is beyond the scope of this document, but we can list a few common features that they should possess:

- They should be designed in order to clearly show important alert information, such as IP addresses and alert type and severity.
- They should help in sorting alert by severity.
- They should provide information on what the alert means.
- They should possess some customizable filtering capabilities, to filter irrelevant alerts.

1.14 Alert Storage

All alerts generated by the IDS are stored permanently for later inspection in a dedicated database, in order to facilitate post-attack analysis and forensics. A database is preferable to a raw log file, since it is more adapted to automated report generation and query work. According to the first security requirement for IDSs, this database should be secured, preferably located on a dedicated computer, isolated from the monitored network and connected to the IDS through a separate channel. Alerts could be stored encrypted.

1.15 Alert-flow Processing

Dynamically increasing the relevancy of the alert-flow between the filters and the end-user is one of the key issues on the way to designing more efficient IDS. This is a challenging problem, closely related to artificial intelligence. Ultimately, a goal is to include an expert system inside IDSs in order to assist the end-user in identifying attacks. A second layer of alert-flow processing can be done beyond the point where the alert-flows generated by each of the filters get merged. Different kind of processing is then possible to reduce the alert-flow and increase its relevancy.

Dynamic Processing Dynamic processing is obtained by modifying dynamically the alert-flow before it reaches the end-user. The dynamic processing engine should not introduce too much delay in forwarding the alert-flow and the quality of its analysis is thus limited by time and delivery requirements.

Passive Processing Passive processing is a form of data mining obtained by analyzing the alert database, in order to identify attack patterns some time after the related alerts were generated. It is not limited by time as for dynamic processing, and can thus provide a deeper and more accurate analysis of alert logs. It could theoretically be able to detect attacks spread over a longer time scale or distributed attacks. On the other hand, due to

the delay between an attack and its recognition, this technique would not be of use for stopping ongoing attacks.

1.16 IDS Administration

An IDS's administration is a heavy task, especially with distributed or remotely managed IDS. Specific administration frameworks are required to support at least the following 4 activities:

IDS Monitoring IDS should provide self-monitoring capabilities, in order to detect the failure of some IDS components and allow to automatically or manually fix it. This can be enhanced through a framework for automatically probing the IDS status.

Configuration IDS configuration is a complex and time-consuming task. For filters, an appropriate configuration framework is strongly advised, in particular for distributed IDS.

Updating As for configuration, having an updating framework to automatically or remotely upgrade various components of an IDS is also an issue, especially in distributed or remotely managed IDS where having a person doing upgrades manually on site can be hard or expensive.

Alert Investigation Being able to efficiently query the alert log is another issue when starting to investigate past sequence of events, hence the importance of having a proper database to store alerts.

1.17 Report Generation

Storing the alerts in a database offers us an interesting option of a periodical report generation. Such a report can be of use for long term surveys, for providing the monitored system administration staff with regular snapshots of the security related activities on their system. Such information will help in administrative decisions concerning system evolution.

1.18 Security Architecture

As seen on figure 1.7 the security process is a never-ending process of new threats/attacks, which may change the policy, which in turn affects the security mechanism, and design of the finishing product.

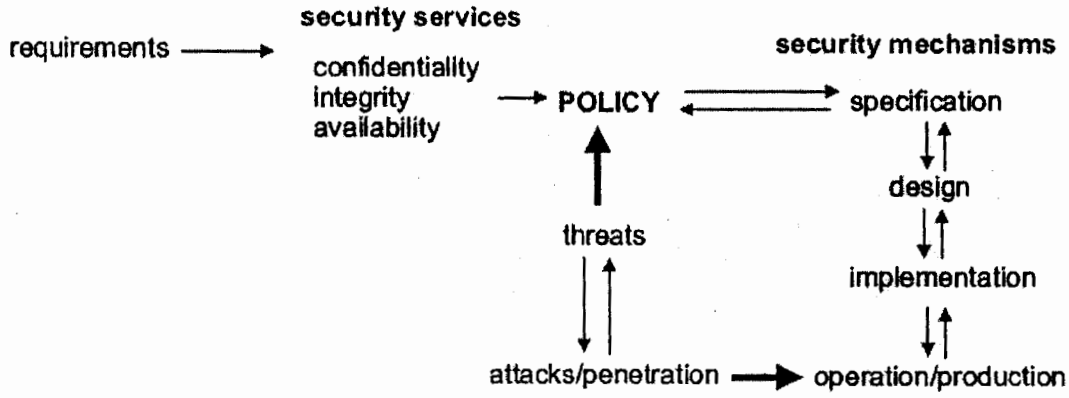

Figure 1.7 the Ever Evolving Process of Security

Chapter No.2
Design

2. Design

The continued discovery of programming errors in network-attached software has driven the introduction of increasingly powerful and devastating attacks. Attacks can cause destruction of data, clogging of network links, and future breaches in security. In order to prevent, or at least mitigate, these attacks, a network administrator can place a firewall or Intrusion Detection System at a network choke point such as a company's connection to a trunk line. A firewall's function is to filter at the header level. If a connection is attempted to a disallowed port, such as FTP, the connection is refused. This catches many obvious attacks, but in order to detect more subtle attacks, an Intrusion Detection System (IDS) is utilized.

The IDS differs from a firewall in that it goes beyond the header, actually searching the packet contents for various patterns imply that an attack is taking place, or that some disallowed content is being transferred across the network. Current IDS pattern databases reach into the thousands of patterns, providing for a difficult computational task. Methods commonly used to protect against security breaches include firewalls with filtering mechanisms to screen out obviously dangerous packets, and intrusion detection systems, which use much, more sophisticated rules and pattern matching to sense potential malicious packets. These techniques require significant computational resources. However, using automated design strategies for highly parallel adaptive soft processors, there is potential for dramatic performance improvements.

2.1 Fundamental Differences in Systems' Behaviors

Each computer system connected to a network possesses its own state and its own set of rules defining the way it passes from one state to another. When monitoring and interpreting some activity, IDS should know exactly the state and the behavioral rules of the monitored system(s), in order to accurately predict the system's reaction. In practice, the IDS is generally forced to make assumption on the state, and to assume that the

behavioral rules of the end-system are implementing standard protocols in the same way as the IDS itself.

These two assumptions are fundamentally flawed. This is especially true with network based IDS monitoring the data traffic between two end-systems and making assumptions on how they generate and handle this data exchange. Presently, each operating system has its own implementation of the TCP/IP protocol stack. The official description (RFCs) of most of these protocols, TCP and IP in particular, allow some aspects of the protocol implementation up to the programmer. This, in addition to the inherent complexity of these protocols, makes that two different implementations of the TCP/IP stack usually presents differences in the way they handle unusual cases of protocol usage (such as with IP fragment reassembly). Some specific network traffic might be interpreted differently depending on the stack implementation of the system receiving the traffic. Typically, some network packets may be accepted by a system and discarded by another.

2.1.1 Insertion and Evasion

Differences in TCP/IP stacks implementation of network based IDS and monitored end-systems are opening the door to two attacks called insertion and evasion described in '*Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection*' (T.H. Ptacek & T.N. Newsham, 1998). *'An IDS can accept a packet that an end-system rejects. An IDS that does this makes the mistake of believing that the end-system has accepted and processed the packet when actually it hasn't. An attacker can exploit this condition by sending packets to an end-system that it will reject, but that the IDS will think as valid. In doing this, the attacker is "inserting" data into the IDS. No other system on the network cares about the bad packets.'*

On the other hand:

'An end-system can accept a packet that an IDS rejects. An IDS that mistakenly rejects such a packet misses its contents entirely. This condition can also be exploited, this time

by slipping crucial information past the IDS in packets that the IDS is too strict about processing. These packets are "evading" the scrutiny of the IDS.' Network based IDS can be fooled using these two techniques. In practice, such techniques are called for IDS evasion techniques, and are most often based on IP fragmentation, packet dis-ordering or invalid packet forging.

2.1.2 Traffic Normalizer

Traffic normalizers are an interesting solution against most insertion and evasion techniques. They can be seen as filter dynamically altering network traffic in order to force it to follow a safe subset of protocol usage. They act as a proxy for traffic forwarding: they take in entry normal traffic, reassemble it, then regenerate an equivalent but safer traffic and forward it. Traffic normalizers would protect a monitored network from being vulnerable to IDS evasion techniques based on differences in TCP/IP stack implementation.

Unfortunately, traffic normalizers are and may stay purely theoretical. It is indeed difficult to define a complete subset of protocol usage that would be free of insertion evasion possibilities.

2.2 Alert-flow Control Challenges

The IDSs ability to efficiently manage the alert-flow is going to become the most important issue in IDS design in the next few years.

2.2.1 False-positive Alerts

False-positives are caused by the filters interpreting wrongly the sensors data, due to a lack of understanding or information about the monitored system. To avoid false-positives, a filter would need to know continuously the state and behavioral rules of the end-systems that generated these data. Such a level of knowledge cannot be reached in

practice. It is therefore impossible to eradicate false-positives. As time is passing, IDS have to monitor a constantly increasing flow of information, thus generating an increasing volume of alerts. Network based sensors will be required to monitor high volume of network traffic, and host based sensors may have to monitor larger systems.

The progressive integration inside distributed IDS of many different sensors will multiply the volume of alerts to treat at a central monitoring point. Although the quantity of attacks may increase, due to the spreading of attack tools and the decreasing of the competence level required to run attacks, it can be expected that normal activities will increase faster. Beside, normal activities generate false positives at a constant rate. As a consequence, the ratio of relevant alerts per false positives may decrease with time. In other words, IDS will probably have to deal with a statistical rise of the percentage of false-positives inside the alert-flow.

2.3 Sniffing on Promiscuous Mode

A sniffer works by looking at every packet sent in the network, including packets not intended for it. Are capturing each and every packet on the network. Sniff the packets and use the information obtained from them for detection and prevention. Sniffing allows collecting of all data (or better a copy of it) on the wire where the network interface is connected to, although the data might not be addressed to this network interface. Setting the network interface into "Promiscuous Mode" does this. This mode indicates that the MAC-Filter on the network interface is turned off, so accepting all data on the wire. Within one network segment, computers find their counterparts by using MAC-addresses. Also all packets that arrive via the router of your network segment, find there recipient by his MAC-address.

The software consists of the following main modules.

- Frame Operation Module
  - Frame Reading
  - Frame Capturing
- Source/destination Info Retrieval Engine

- MAC Reader
- Source Port
- Destination Port
- Source IP
- Destination IP

- Marker
- Signature Extractor Module
  - Data Portion Reader
  - Signature Extraction
- Activation Engine
- Support System

The basic architecture is extended in various ways. For better area performance, we present a partial tree architecture that allows for significant reduction in redundant comparisons by independently matching prefixes that are shared across a range of patterns. To provide increased throughput performance, we provide a design that replicates a fraction of the hardware to allow for exact matching for k bytes per cycle. For high throughput with exceptional area efficiency, we provide an architecture that sacrifices exactness and allows for an increased false positive rate. To achieve better utilization of these architectures, system-level preprocessing steps are required, serving various functions including partitioning, grouping, and code generation.

Frame Capturing Module: This module of the soft ware is divided into two sub parts namely, Frame capturing and frame reading. The first step in frame capturing is to get a pointer or handle the Ethernet adapter. After getting the handle of the adapter we can access it. The reason for getting the handle is to capture the data (frames) arriving on the network. Normal Ethernet adapters reject all incoming traffic that isn't sent to that adapter. In order to sniff on the wire, the adapter must be reconfigured to accept all traffic on the wire. i.e. we have to convert the adopter into promiscuous mode. For that purpose a driver must be written that puts both the adapter into promiscuous mode and buffers the incoming frames.

Source/Destination Info Retrieval Engine: This engine comprises of four sub modules, which are

- Source Port Retrieval
- Destination Port Retrieval
- Source IP retriever
- Destination IP Retriever

The purpose of this engine is to get the information of both source and destination ports along with its IP addresses, so that if any miss use is observed that IP can be identified and be stopped.

Marker: This module of the software assigns different color marks to the packets e.g. SYN/ACK frame has been exchanged between them the color assigned to these packets is "Blue". Indicating, no need to be alarmed.

Signature Extraction Module: This module comprises of two parts, one that gets the data portion of each packet and the other extracts the signature. The signatures present in the database are used to detect the known attacks. But to make the product more secure we are managing unknown attacks as well. As we know when any harmful executable file runs it may cause damage. Such files are prohibited to execute on the basis of their signature.

Activation Engine: The mechanism of activation for most of the attacks is very common. After entering into the system they run their setup portion that places them at the required path and associates them with some operating system's file or event and when that event occurs actual payload of the attack becomes active. Most common way is to use registry for that purpose. In **RUN** or **RUN ONCE** section of the registry the attack adds its entry, so when the system reboots activation of attack takes place.

Stopping Activation: The activation of malicious files is stopped, by diverting their path on their arrival, because they need specific path for their activation normally is **Windows/System** folder. When our application starts we scan the hard disk and search all exe files and keep their record. Now whenever we get an alert for an executable entry we just divert the file to a self-made folder, which stops its activation. It may be possible that an executable file entering the system is useful rather than harmful, no problem

because we are not destroying the files, we are just diverting them so when ever required administrator can delete the harmful ones, after which useful files can be reused.

Support System: For completely removing the effects of attack, we have provided the facility of online help. It means that manual solution about each attack is available that involves how to delete not required registry entries and how to update the modified files etc. By following these steps, complete recovery is possible.

2.4 Optimized Incremental Design

A problem with recent designs utilizing hard-wired comparator modules is in the requirement for a full place-and-route to make any change, no matter how small it is. Because of the exceptional area and time efficiency possible with this customized design paradigm, this issue has been largely ignored.

The partition least modified by the addition of the new rule is determined by comparing the pre-decoded bits already within the partition, as well as the potential for using previously mapped prefixes. The tool then modifies this VHDL code describing this partition. If the new pattern shares a prefix with some other pattern in the partition, the partial result of the previous pattern is mapped to the new pattern, reducing new wiring. The removal of rules is far easier, only the connections to the final result tree are removed. The new partition code is sent to the incremental synthesis and place-and-route functions of Xilinx ISE 6.2. The tool only re-synthesizes the modified modules. Because of the previously defined area constraints, each pipeline module is independent of the others. Thus, only the routing in the modified module requires place and route.

2.5 Learning From the Antivirus World

An interesting parallel can be drawn between the virus and anti-virus along with attacks and IDS. The anti-viruses are updated only when a new virus is detected, similarly the IDS are upgraded when a new attack is found. This can give us an insight on the future of IDS

2.5.1 Similarities Between Antivirus and IDS

Virus detection presents many similarities with attack detection. A virus can often be identified through a signature, like most of the attacks. Antiviruses have therefore been using signature detection from the beginning, as IDSs did. A virus acts unconventionally as a program. For example, it replicates its own code and modifies other executables, or modifies master boot records, which usual programs don't tend to do. A virus's activity can thus be detected with anomaly detection techniques. Antivirus indeed uses anomaly detection, although they call it 'heuristic analysis'. Besides, most of the techniques used in viruses and antivirus design have an equivalent in the IDS world. Viruses and antivirus have been studied and produced for more than 10 years, and are now technically mature, while IDSs are still in their youth. Antivirus systems have been available commercially many years earlier than IDSs, and it can be assumed that IDSs will follow the same evolution at least on a few aspects.

2.5.2 The Race for Polymorphism

The first antivirus programs were signature based. In the early days of viruses, all the copies of a virus had indeed the same body, and it was therefore possible to find all the copies of a virus spread inside an infected system just by searching systematically for an occurrence of a specific part of the virus's body. To escape antivirus detection, many techniques were developed by virus writers to reduce and finally remove any similarity between one virus and its copies. The idea was at first to encrypt the body of the virus and allow only the decrypting procedure. Then were developed 'mutation engines' that were able to generate a functionally identical decryption procedure, but encoded differently in terms of assembly instructions. Mutation engines are exploiting the fact that same operation can be written in many different ways at the assembly level.

Viruses using mutation engines are called polymorphic viruses, and are impossible to detect with raw signature detection since there is no constant similarity between two copies of the same virus. In reaction to this evolution, antivirus started using

heuristic analysis, which is the antivirus equivalent of IDS's anomaly detection. Such anti-viruses monitor the activity of programs on systems in an attempt to detect behaviors, which are typical of viruses. The interesting point is that attacks are following the example of viruses with a few years delay, and are currently just starting to explore the possibilities of polymorphism.

There are nowadays attempts to write polymorphic attacks2, or tools for generating polymorphic buffer-overflows3. These techniques are still very new, but can be expected to grow quickly. IDS that are exclusively using signatures will then become inefficient against a wide range of attacks.

2.5.3 Evasion Tricks

While seeking polymorphism, viruses were adapting techniques to evade antivirus detection using system breaches and tricks to become stealthier. Some viruses are even trying to detect an antivirus presence and consequently hide themselves or attack the antivirus. Most of the changes between virus generations are focused on developing and exploiting new tricks to evade detection. As a result, viruses and antivirus are engaged in a constant race between evasion and detection. In much the same way, attacks are now trying to exploit system or IDS breaches in order to evade detection. Considering the importance of such techniques in the virus world, we can expect them to become equally important in the IDS world. In particular, attackers have not been focusing so much yet on trying to identify IDSs and exploit their specificities to evade them. Moreover, a race between evasion and detection implies a constant need for research and reaction from the IDS development side, which should be taken into account when planning long term resources for IDS development.

2.5.4 Superiority of the Heuristic Approach

The antivirus equivalent of anomaly detection is called heuristic analysis. It covers a range of system monitoring techniques aiming at detecting signs of a virus's

activity. In this approach, a model is built describing viruses' typical activities, and the system is monitored in order to detect programs behaving according to this model. From a certain point of view, anomaly detection works in the opposite way, by detecting activities that are not belonging to a model of 'normal usage'. Yet, the implementations of these two techniques are very similar, and it can be assumed that they have an equivalent efficiency.

The heuristic analysis has been proven technically efficient for detecting viruses, especially viruses that have not been classified yet and for which signature based antivirus still do not possess any signature. Yet, heuristic analysis has two drawbacks that justify their lack of success as commercial products:

1. They do not provide clear alert information. Where a signature based antivirus can provide an alert specifying clearly which virus has been detected and how to remove it, heuristic engines would send alerts describing an anomaly in program behavior, which can be hard to interpret for non-educated users. This does not fit with the aim of most antivirus products to reach a large market, and thus users without a deep computer experience.
2. They are commercially less profitable than signature-based products, since the user doesn't have to subscribe to periodical signature database updates.

Antivirus soft wares are now seldom heuristic based only. The tendency is rather to rely on signature detection, while simultaneously running a heuristic engine in order to automatically detect new viruses. Suspicious programs are then automatically sent to the antivirus company for a further analysis, which will eventually lead to a signature release. If we perform a raw transposition of the lessons of heuristic analysis to anomaly detection, we can project the following results:

- Anomaly detection works, but requires an educated monitoring. Signature based detection should be preferred when the monitoring personnel is lacking computer experience.

- Anomaly detection could be used in parallel with signature detection, thus allow developing an automated process for detecting, analyzing and generating signatures of new attacks. This would be a first answer to the polymorphism and evasion problems.

Most of the attacks found could be handled by searching for some signature in the telnet packets. This kind of approach has known drawbacks: it is indeed easier to implement, but hard to update, and unable to 'understand' the protocol, thus inefficient in complex detection situations and susceptible to generate false-positives. A better approach is to integrate both an anomaly and a signature engine to the filter. The filter should be able to disassemble the telnet stream and divide it into two streams: a telnet option stream and a user-server data stream.

Anomaly detection would be performed on the telnet option stream to check for compliance with the telnet protocol, and on the user-server stream to monitor the login stage. The user-server data stream would pass through the command-checking engine. The filter should furthermore be able to differentiate the login stage of a telnet session from the 'user logged in' stage, and maintain internally a list of failed login attempts with their originating IP addresses, in order to enable brute force detection.

Yet, this signature matching can be performed much more accurately thanks to the anomaly engine in charge of analyzing the option stream. More accurate signature matching also reduces false positives. We could for example check for ENVIRON related attacks only in the ENVIRON option header when it occurs, and not systematically through all telnet traffic, as most of the filters currently do. A *Data Parser* separates telnet options from user data. An Option Validity Checker and a Sub Negotiation Validity Checker check Telnet options for their compliance with RFCs. Depending on whether the user is authenticated or not, data is passed through a *Login Checker* (checking for long or non printable usernames and passwords) or a *Good Usage Monitor* monitoring the user activity. A *Brute Force Monitor* runs in parallel, in cooperation with the *Login Checker*.

2.6 System Model

The system model is based upon the ever-evolving process of security. TDS will work as a security guard to protect the system from all sorts of attacks and viruses. IDS will capture all the frames passing through the network and display the results. Further more it will scan all the executable files and deactivate their execution. It will generate an alert if any intrusion is detected.

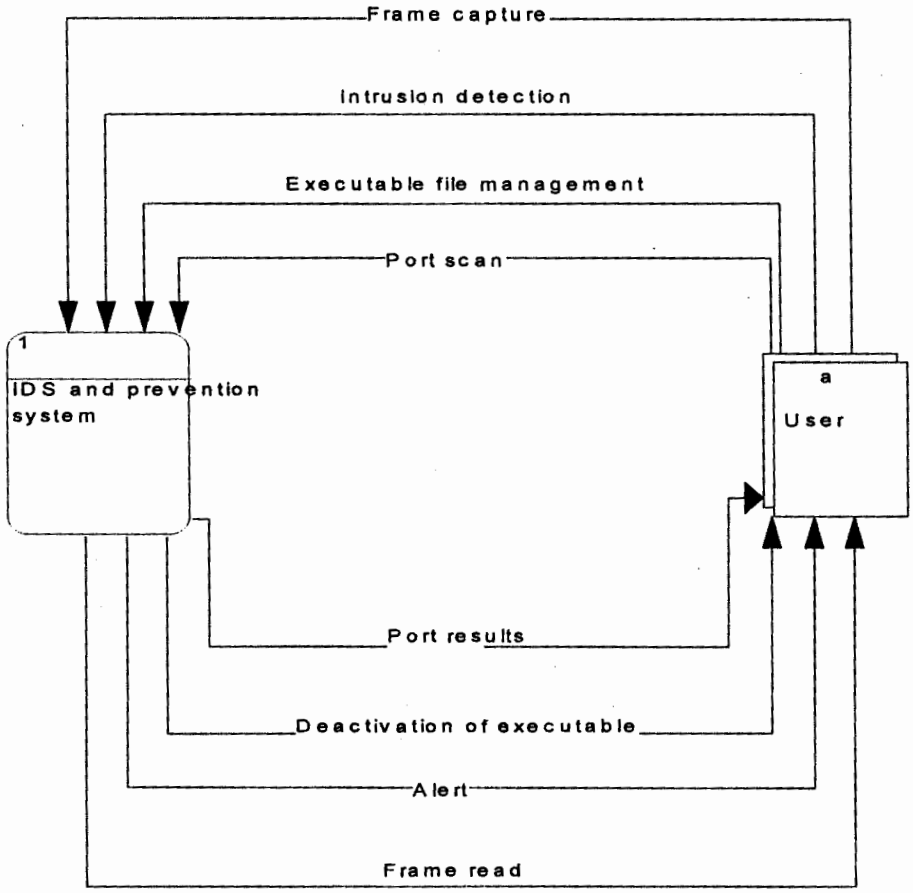

Figure 2.2 System Model

2.7 Work Flow Diagram

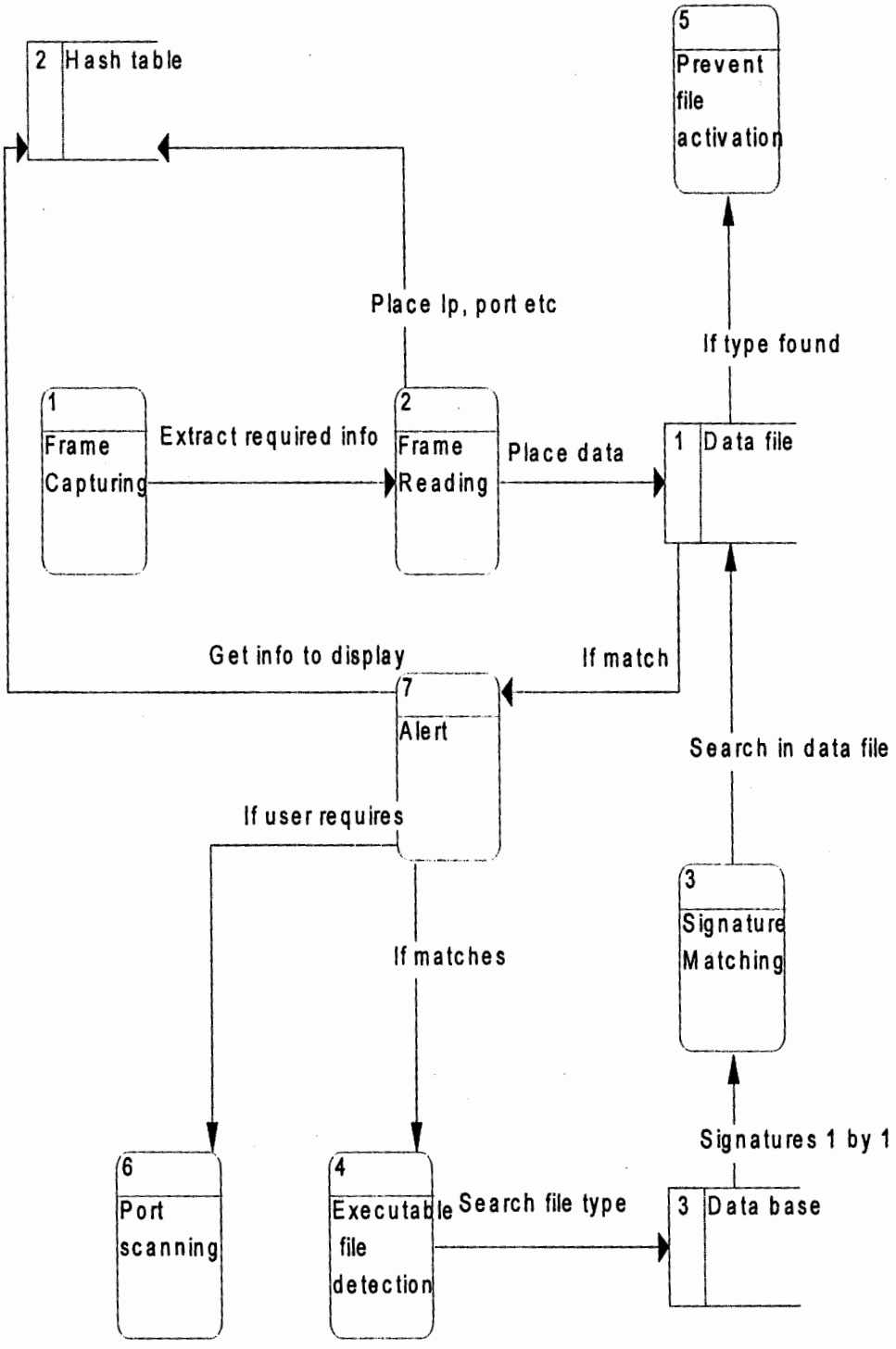

Figure 2.2 Work Flow Diagram

2.8 Flow Chart

The flow of Sniffer IDS is very simple. Once the Sniffer is on capturing mode it initially converts the NIC to promiscuous mode. It starts capturing frames from the network. If the files are not malicious it will allow them. If the files are not malicious it will capture them and match its signature with the saved signatures in the database. If the signature pattern is matched, it will generate an alert and stop the file from activation. Otherwise it will allow the file to flow on the network. This whole process iterates again and again until the IDS is stopped.

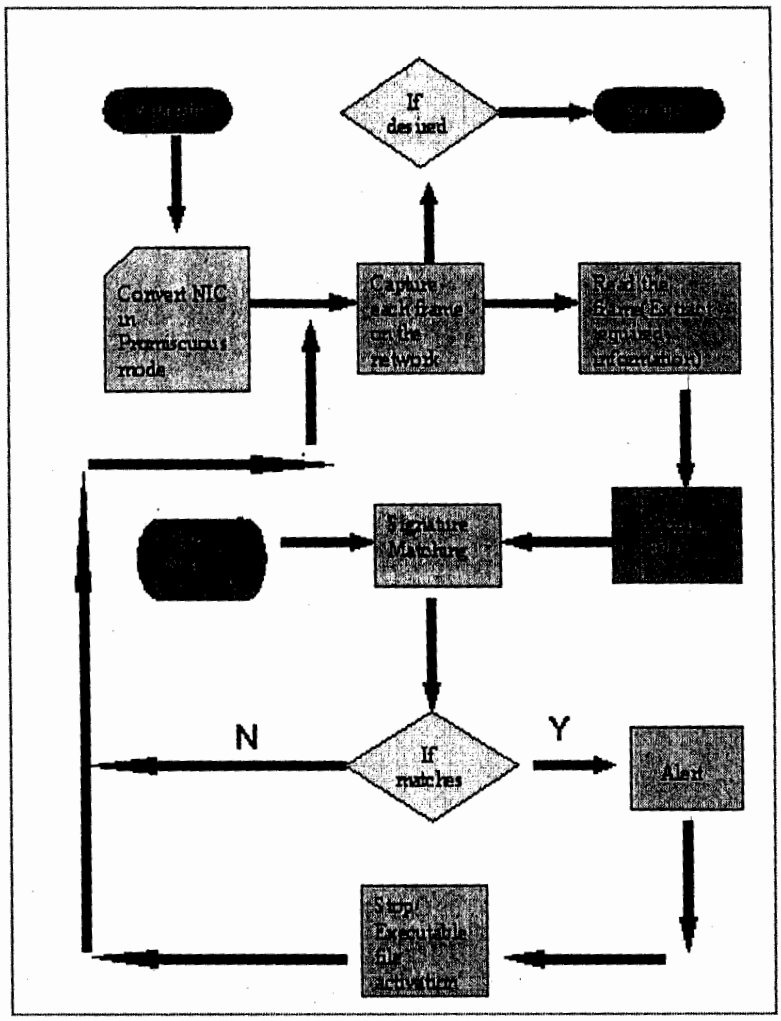

Figure 2.3 Flow Chart

Chapter No.3
Implementation

Chapter No.4
Results & Discussions

4. Results & Discussions

The TCP filter proved to be extremely accurate in its analysis of anomalies and port scans. It can for example make the difference between a normal scan and an OS fingerprinting scan. The filter does not solve on the other hand the problem of slow or distributed port scans. Yet, if configured to be extremely sensitive to port scans, the filter can be used as a first detection layer sending alerts for even very small scans concerning only a few ports for a given period of time. Later on, a data-mining engine can analyze these alerts and detect the slow or distributed scans.

4.1 Viruses Captured

Once the software is run on the PC it matches the patterns of the malicious files, which are passing in and out of the network. With the passage of time the number of caught viruses increases. The software updates the signature database every time it catches a new virus. Hence it makes it more efficient and reliable.

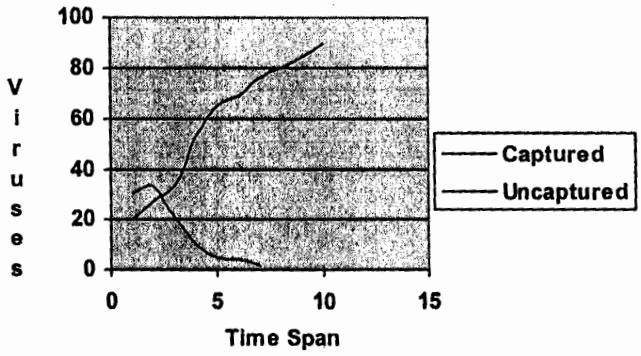

Figure 4.1 Shows the Virus Captured with Time

Fig. 5.1 shows that the number of caught viruses increases gradually with the passage of time. Where as the number of uncaught virus is decreasing. The main reason behind it is that every time a virus is caught the database is updated, which in return saves

the time for searching the information of the virus. Every time a virus is caught it generates an alarm, and if it is an exe file it is halted or stopped before it can be executed.

4.2 Reduction of PINGs

This function is used to block all ICMP traffic, it means the by applying that we can block all pings. Although we are primarily dealing with TCP/IP packets only but this is added to show that the blocking is working properly.

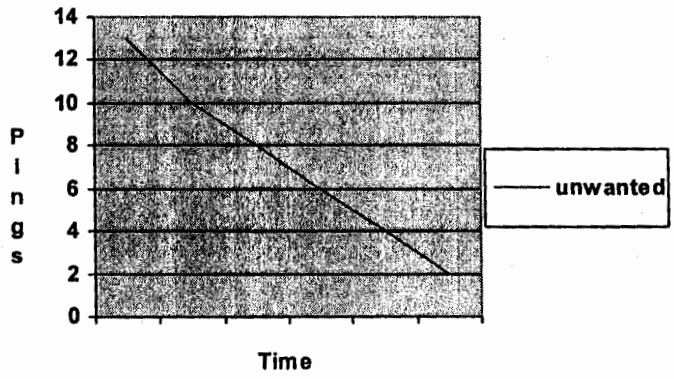

Figure 4.2 Shows the Reduction of Unwanted PINGs

Graph in fig. 4.2, shows that the number of unwanted PINGs has decreased. The reason behind this decrease is that we can easily block the PING and for that particular System our system would appear disconnected from the network. This feature of this software makes it very reliable for the network. It enables the user to report the abuse of a particular system to the administrator. We know the the IP of abuser, it would be easy from the admin to locate the system on the network and take actions against it.

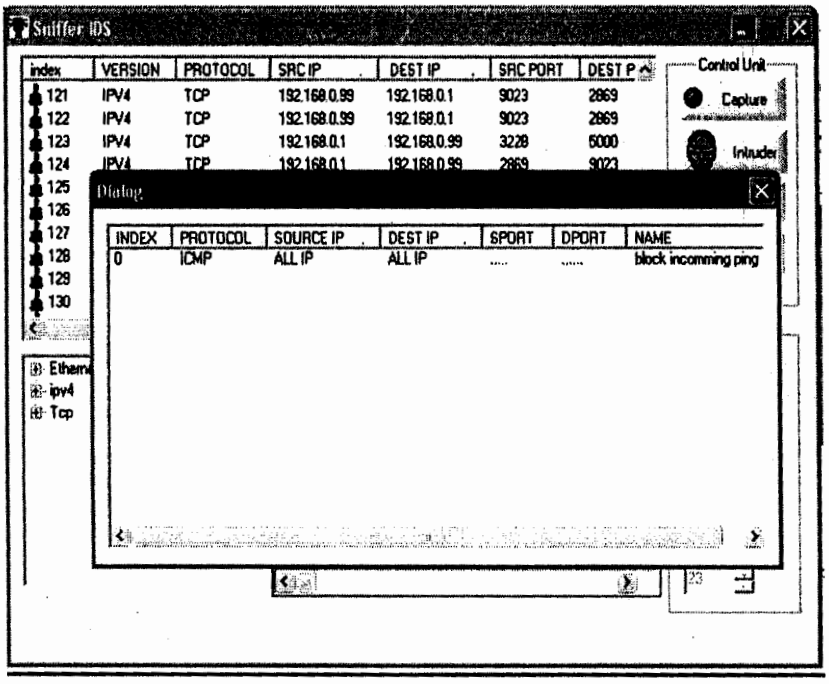

Figure 4.3 Shows how PING can be Blocked

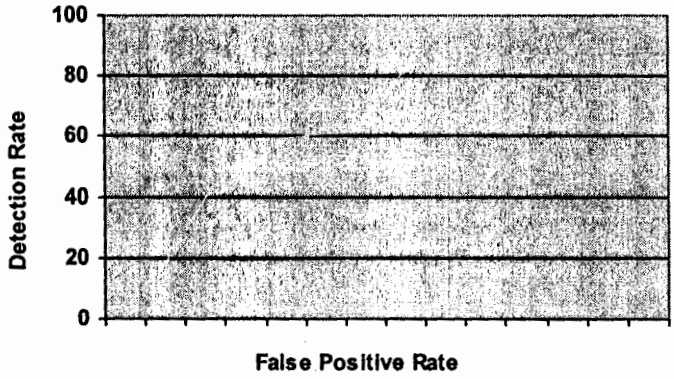

Figure 4.4 Shows that how Secure Signature based IDS is for False Positive

4.3 Conclusion

Everyone now has no doubt that "Intrusion detection systems have become an essential component of computer security to detect attacks that can occur despite the best preventative measures." Deploying the right tools to defend and protect a perimeter

requires man-hours, patience and knowledge. Security is more complex than any other organization, business process, or person's view or agenda. The IDS research community is developing better techniques for collecting and analyzing data in order to handle intrusions in large, distributed environments. In order to take an advantage of this work, ID systems must be able to quickly adapt to new, improved components, and changes in the environment. After many years in the security field, we believe no one product today or tomorrow will solve every security need.

There are too many variables to take in considerations in knowing everything about security. That is why security teams exist such as CERT/CC with many analysts, each with their own areas of expertise. Every member provides their own strengths and experiences to complement one another. With new intrusions appearing each day, it has become a race between upgrading the intrusion detection system and attackers finding new ways of getting into the various systems deployed on a network.

However, these security teams usually face obvious challenges. Organizations collect huge volumes of data in their daily operations. This wealth of information is often underutilized because of economic reasons (weak or no database search capability) also, lack of trained personnel to correctly interpret the data. Therefore, in order to sift through large amount of data to discover hidden clues, data mining (also known as Knowledge Discovery in Databases) can be used to dissect the information. Data mining helps revealing relationships or trends to answer specific questions too complex for traditional query and reporting tools.

Recent years have seen a dramatic increase in the amount of information stored in electronic format. It has been estimated that the amount of information in the world doubles every 20 months and the size and number of databases are increasing even faster. The business world has provided some important research and testing by creating knowledge discovery database applications designed for managing the growth of on-line data volumes. IDS, a router, a firewall, or a server can generate mountains of data with very little means of merging the data to extract the centre and drill down on the attack. A

security analyst's nightmare faced daily, is the amount of false positive data collected by IDS sensors.

Being able to recognize low and slow reconnaissance probes or correlating information when amalgamated together. Therefore, yielding significant amount of intelligence is very important. Tools such as Intellitactics' Network Security Manager [9], can be used to drill down the correct information. The approach Intellitactics has taken regarding data mining and the manipulation of huge volumes of information is opening everything and letting the Network Security Manager (NSM) do all the work. NSM uses a six-step approach: collection and data consolidation (awareness process), normalize, classifies the assets, prioritize (understanding process) and analyze and response (appropriate response process).

Bibliography
& References

Bibliography & References

[1] J. P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, James P. Anderson Co., Fort Washington, PA, April 1980.

[2] Dorothy E. Denning. An Intrusion-Detection Model. IEEE Transactions on Software Engineering, 13(2): 222{232, February 1987.

[3] Dorothy E. Denning, D. L. Edwards, R. Jagannathan, T. F. Lunt, and P. G. Neumann. A Prototype IDES| A Real-Time Intrusion Detection Expert System. Technical report, Computer Science Laboratory, SRI International, 1987.

[4] Dorothy E. Denning and Peter G. Neumann. Requirements and Model for IDES { A Real-Time Intrusion Detection System. Technical report, Computer Science Laboratory, SRI International, August 1985.

[5] T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. G. Neumann, H. S. Javitz, A. Valdes, and T. D. Garvey. A Real-Time Intrusion Detection Expert System (IDES) { Final Technical Report. Technical report, SRI Computer Science Laboratory, SRI International, Menlo Park, CA, February 1992.

[6] Teresa F. Lunt, R. Jagannathan, Rosanna Lee, Sherry Listgarten, D. L. Edwards, P. G. Neumann, H. S. Javitz, and A. Valdes. Development and Application of IDES: A Real-Time Intrusion-Detection Expert System. Technical report, SRI International, 1988.

[7] Teresa F. Lunt, R. Jagannathan, Rosanna Lee, Alan Whitehurst, and Sherry Listgarten. Knowledge based Intrusion Detection. In Proceedings of the Annual AI Systems in Government Conference, Washington, DC, March 1989.

[8] Sandeep Kumar. Classi_cation and Detection of Computer Intrusions. PhD thesis, Purdue University, West Lafayette, IN 47907, 1995.

[9] Stephanie Forrest, Steven Hofmeyr, Anil Somayaji, and Thomas Longstaff. A sense of self for Unix processes. In Proceedings of the 1996 IEEE Symposium on Security and Privacy. IEEE Computer Press, 1996.

[10] Terran Lane and Carla E. Brodley. Temporal sequence learning and data reduction for anomaly detection. In Proceedings of the Fifth ACM Conference on Computer and Communications Security, pages 150, 158. ACM, 1998.

[11] Teresa F. Lunt. Automated Audit Trail Analysis and Intrusion Detection: A Survey. In Proceedings of the 11th National Computer Security Conference, October 1988.

[12] Teresa F. Lunt. A Survey of Intrusion Detection Techniques. Computers & Security, 12(4):405, 418, June 1993.

[13] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A Network Security Monitor. In Proceedings of the IEEE Symposium on Research in Security and Privacy, May 1990.

[14] Judith Hochberg, Kathleen Jackson, Cathy Stallings, J. F. McClary, David DuBois, and Josephine Ford. NADIR: An automated system for detecting network intrusion and misuse. Computers and Security, 12(3): 235{248, May 1993.

[15] Biswanath Mukherjee, Todd L. Heberlein, and Karl N. Levitt. Network intrusion detection. IEEE Network, 8(3):26{41, May/June 1994.

[16] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, Tim Grance, L. Todd Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukherjee, Douglass L. Mansur, Kenneth L. Pon, and Stephen E.Smaha. A System for Distributed Intrusion Detection. In Proceedings of COMPCON Spring '91, 36th IEEE Computer Society International Conference, pages 170{176, San Francisco, CA, February 25 March 1 1991. IEEE Computer Society, IEEE, IEEE Service Center, Piscataway, NJ.

[17] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype. In Proceedings of the 14th National Computer Security Conference, pages 167, 176, Washington, DC, October 1991. National Institute of Standards and Technology.

[18] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS: A graph based intrusion detection system for large networks. In Proceedings of the 19th National Information Systems Security Conference, volume 1, pages 361, 370. National Institute of Standards and Technology, October 1996.

[19] Naji Habra, B. Le Charlier, A. Mounji, and I. Mathieu. ASAX: Software Architecture and Rule-based Language for Universal Audit Trail Analysis. In Proceedings of ESORICS 92, Toulouse, France, November 1992.

[20] Abdelaziz Mounji, Baudouin Le Charlier, Denis Zampunieris, and Naji Habra. Distributed audit trail analysis. Technical Report RP-94-007, Institut d'Informatique, FUNDP, Rue Grandgagnage 21, Namur, Belgium, November 1994.

[21] Phillip A. Porras and Peter G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In Proceedings of the 20th National Information Systems Security Conference, pages 353, 365. National Institute of Standards and Technology, 1997.

[22] Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez, Eugene Spafford, and Diego Zamboni. An architecture for intrusion detection using autonomous agents. Technical Report 98-05, COAST Laboratory, Purdue University, May 1998.

[23] Mark Crosbie and Eugene Spafford. Defending a computer system using autonomous agents. In Proceedings of the 18th National Information Systems Security Conference, Oct 1995

[24] Bruce Barnett and Dai N. Vu. Vulnerability assessment and intrusion detection with dynamic software agents. In Proceedings of the Software Technology Conference, April 1997.

[25] Stephanie Forrest. Personal communication, 1999. Department of Computer Sciences, University of New Mexico.

[26] Kymie Tan. An application of neural networks to UNIX computer security. In Proceedings of the IEEE International Conference on Neural Networks, November 1995.

[27] Jeremy Frank. Artificial intelligence and intrusion detection: Current and future directions. In Proceedings of the 17th National Computer Security Conference, Baltimore, MD, October 1994.

Glossary

Glossary

agent

A computer program that reports information to another computer or allows another computer access to the local system. Agents can be used for good or evil. Many security programs have agent components that report security information back to a central reporting platform. However, agents can also be remotely controlled programs hackers use to access machines.

algorithm (encryption)

A set of mathematical rules (logic) for the process of encryption and decryption.

alias

A shortcut that enables a user to identify a group of hosts, networks, or users under one name. Aliases are used to speed user authentication and service configuration. For example, in configuring a Firebox a user can set up the alias "Marketing" to include the IP addresses of every network user in a company's marketing department.

armed

When a Firebox is armed, it is actively guarding against intrusion and attack.

attack

An attempt to break into a system.

authentication

1. The process of identifying an individual, usually based on a user name and password. Authentication iusually requires something a person has (such as a key, badge, or token), something a person knows (such as a password, ID number, or mother's maiden name), or something a person is (represented by a photo, fingerprint or retina scan, etc). When authentication requires two of those three things, it is considered strong authentication.

2. A method of associating a user name with a workstation IP address, allowing the tracking of connections based on name rather than IP address. With authentication, you can track users regardless of which machine a person chooses to work from.

backbone

A term often used to describe the main network connections composing the Internet.

backdoor

A design fault, planned or accidental, that allows the apparent strength of the design to be easily avoided by those who know the trick.

bandwidth

The rate at which a network segment can transfer data.

bridge

A piece of hardware used to connect two local area networks, or segments of a LAN, so that devices on the network can communicate without requiring a router. Bridges can only connect networks running the same protocol.

broadcast

A network transmission sent to all nodes on a network.

broadcast address

A special type of networking address that denotes all machines on a given network segment.

Client/Server

A network computing system in which individual computers (clients) use a central computer (server) for services such as file storage, printing, and communications.

cracker

Another term for someone who attempts to defeat network security measures, with hostile intent. Commonly used in popular media as a synonym for hacker.

DES

A commonly-used encryption algorithm that encrypts data using a key of 56 bits, which is considered fairly weak given the speed and power of modern computers. Until recently it was the US government's encryption standard, but it has largely been replaced by Triple-DES and AES.

denial of service attack (DoS)

A type of attack aimed at making the targeted system or network unusable, often by monopolizing system resources. For example, in February 2000 a hacker directed thousands of requests to eBay's Web site. The network traffic flooded the available Internet connection so that no users could access eBay for a few hours. A *distributed* denial of service (DDoS) involves many computer systems, possibly hundreds, all sending traffic to a few choice targets. The term "Denial of Service" is also used imprecisely to refer to any outwardly-induced condition that renders a computer unusable, thus "denying service" to its rightful user.

Ethernet

One of the least expensive, most widely deployed networking standards, enabling the transmission of data at 10 million bits per second (Mbps), using a specified protocol. A more recent Ethernet standard, called 100BaseTx, enables data to be transmitted and received at 100 Mbps.

External network

Any network that can connect to yours, with which you have neither a trusted or semi-trusted relationship. For example, a company's employees would typically be trusted on your network, a primary vendor's network might be semi-trusted, but the public Internet would be untrusted — hence, External.

fast Ethernet

An Ethernet networking system that transmits data at 100 million bits per second (Mbps), ten times the speed of an earlier Ethernet standard. Derived from the Ethernet 802.3 standard, it is also known as *100Base-T*.

firewall

Software or hardware components that restrict access between a protected network and the Internet, or between other sets of networks, to block unwanted use or abuse.

gateway

A system that provides access between two or more networks. Gateways are typically used to connect networks that are dissimilar. The Firebox often serves as the gateway between the Internet and your network.

hexadecimal

A base-16 numbering system (from *hexadecem,* Latin for 16) particularly important in computer programming, since four bits (each consisting of a one or zero) are succinctly expressed using a single hexadecimal digit. Hexadecimal resembles decimal (base-10) numbering with the digits 0 through 9, but the decimal equivalents of 10 - 16 are represented in hexadecimal by the letters A through F.

host

A network-connected computer.

hub

A device that serves as a common connection point for multiple devices on a network. There are several different types of hubs, but in general each receives and sends signals to all the devices connected to it.

intranet

A self-contained network with a limited number of participants who extend limited trust to one another in order to accomplish an agreed-upon goal. For example, a manufacturer and its key vendors might create an intranet to facilitate managing the process of turning raw materials into finished products

IP spoofing

The act of inserting a false (but ordinary-seeming) sender IP address into the "From" field of an Internet transmission's header in order to hide the actual origin of the transmission. There are few, if any, legitimate reasons to perform IP spoofing; the technique is usually one aspect of an attack.

node

A computer or CPU on a network.

packet

A unit of information formatted according to specific protocols that allow precise transmittal of data from one node in a network to another. Also called a datagram or a data packet, it contains two parts: a header and a payload. The header is like an envelope; the payload is the contents. In Internet Protocol, any message that is larger than 1,500 bytes gets fragmented into packets for transmission.

packet filtering

Controlling access to a network by analyzing the headers of incoming and outgoing packets, and letting them pass or halting them based on rules created by a network administrator. A packet filter allows or denies packets depending on where they are going, from whom they are sent, or what port they use. Packet filtering is one technique, among many, for implementing security firewalls.

peer-to-peer

Sometimes abbreviated as P2P, this is a method of distributing files over a network where all computers are treated as equals (in contrast to a *client/server* architecture). Using P2P *client* software, a client can receive files from another client. Some P2P file distribution systems require a centralized database of available files (such as Napster), while other distribution systems like Gnutella are decentralized.

ping

A utility to determine whether a specific IP address is accessible. It works by sending a packet to the specified address and waiting for a reply; hence, it was named after the sound echo sonar makes when trying to locate an object.

port

1. A physical hole in a computing device where you plug something in (such as, "this PC communicates with the printer via the serial port").
2. When used in relation to IP services, a made-up, or *logical*, endpoint for a connection, conceived so that the computer can handle multiple applications over one network connection. Your system figures out how to treat data coming at it partially by looking at what port the data is destined for (for example, HTTP, or Web traffic, by convention uses port 80; SMTP, or e-mail traffic, uses port 25).

server

A computer that provides shared resources to network users. The network users are often referred to as *clients* of that server.

spoofing

Altering data packets to falsely identify the originating computer. Spoofing is generally used when a hacker wants to make it difficult to trace where the attacks are coming from.

worm

A self-replicating program that seeks access into other computers by exploiting security flaws. After a worm penetrates another computer, it continues seeking access to other areas. Worms often steal or vandalize computer data. Many viruses are more accurately termed worms, and use e-mail or database systems to propagate themselves to their victims.

Appendix A

User Manual

The working of our software is relatively easy and user friendly. This is the basic design for the software. The interface of the software is simple and has simple functionalities. Here we are going to show the working of our software.

Getting Started

Once the software is running the interface which opens up is as following.

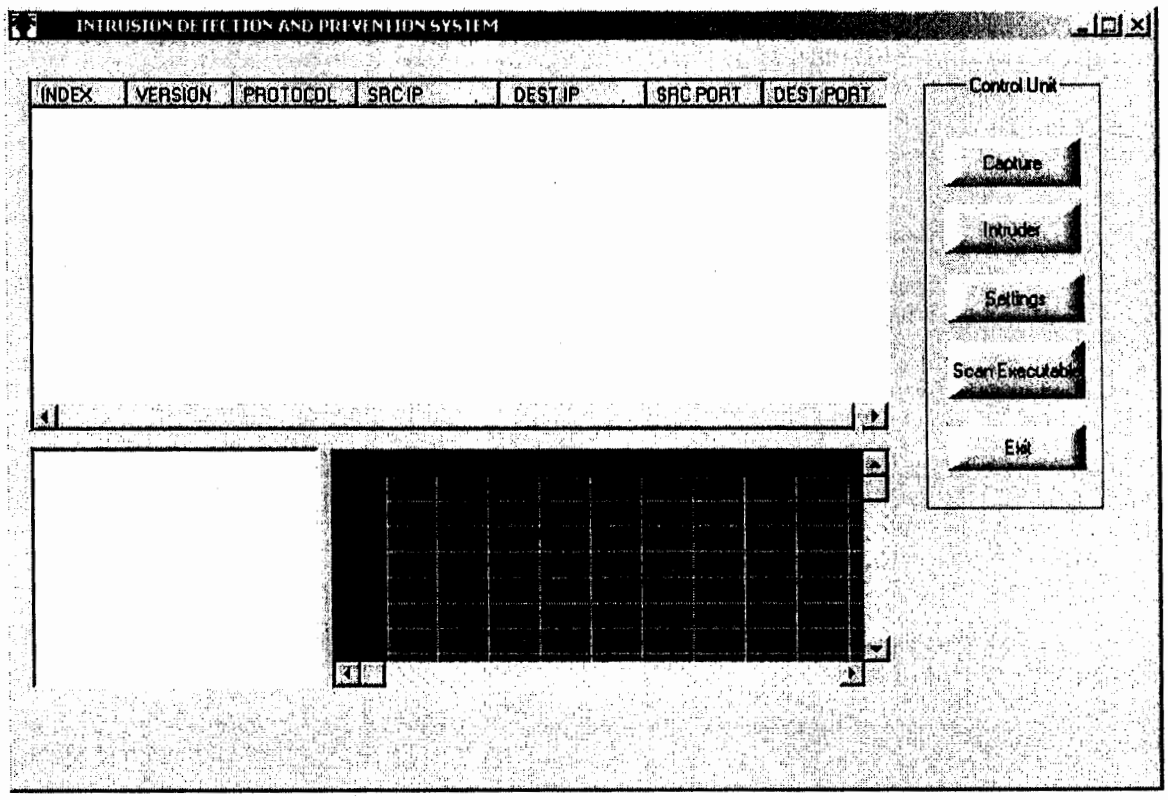

Figure A.1 The Main Interface of the Software

Before the software loads it asks the user for the medium of security. That is if the user wants medium security or high security. Once that is done the software loads completely and we get this interface, which is in figure A.1

On the left hand side of the software there is a complete description displayed, like Source IP, Destination IP, Index, Protocol etc. once there is a connection between the two systems it will show the flow of traffic. On the right hand side we have a simple control panel, which is very easy to use. This control panel is used to control the working of the software. There are five basic buttons that are

- Capture
- Intruder
- Settings
- Scan Executables
- Exit

Capture: This button is used to activate the software while the computer is connected to a network

Intruder: This button is used to check the intruders, who have been caught by the software. It gives a list of all the viruses, which have been caught by the software. If someone wishes to see more info about the virus, then simply click on the name of the virus and it will take the user to a website from where the user can get more information on that particular virus.

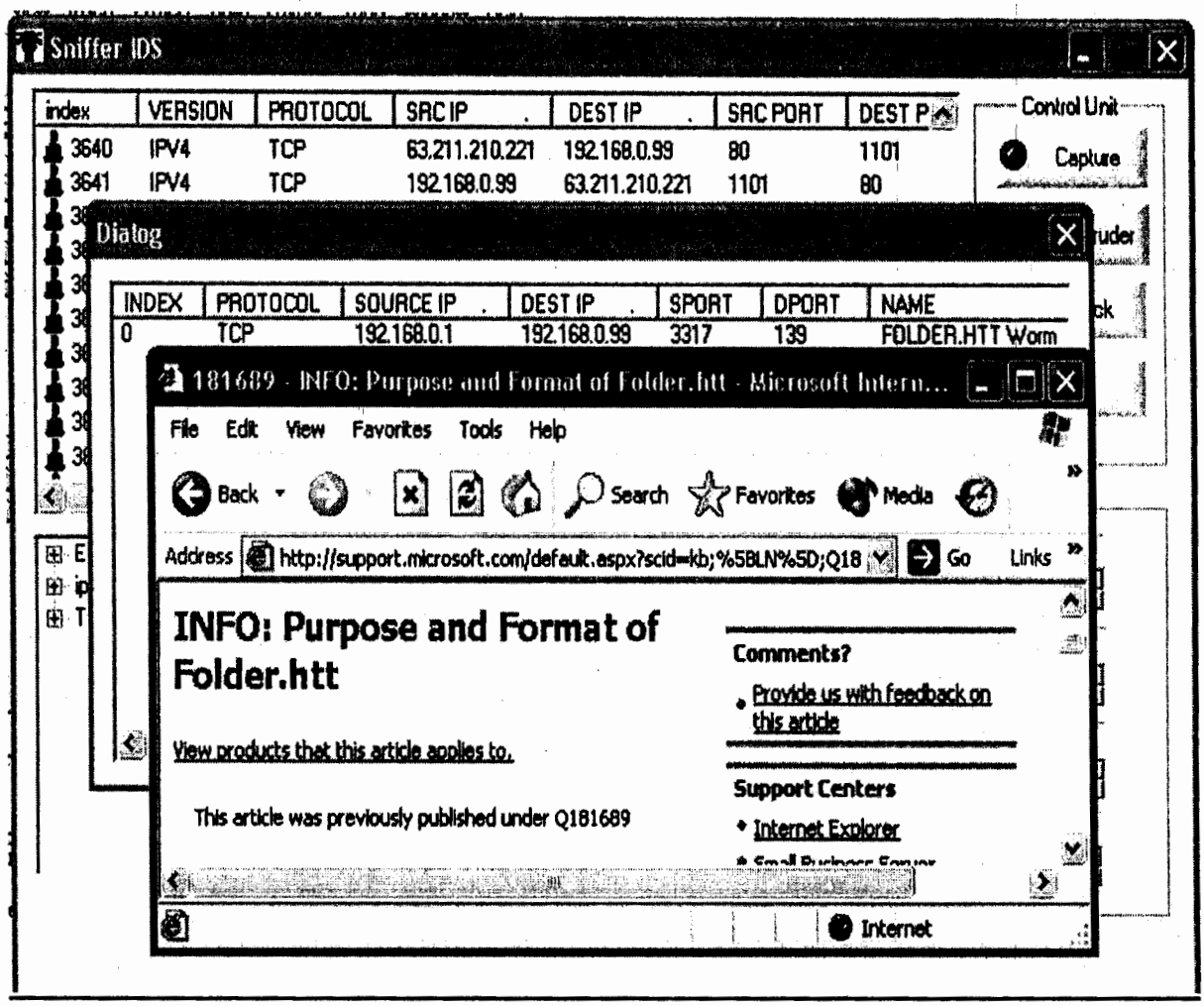

Figure A.2 Intrusion Detected along With the Online Help

Settings: From this button the user can set different levels of security as shown in the figure A.3

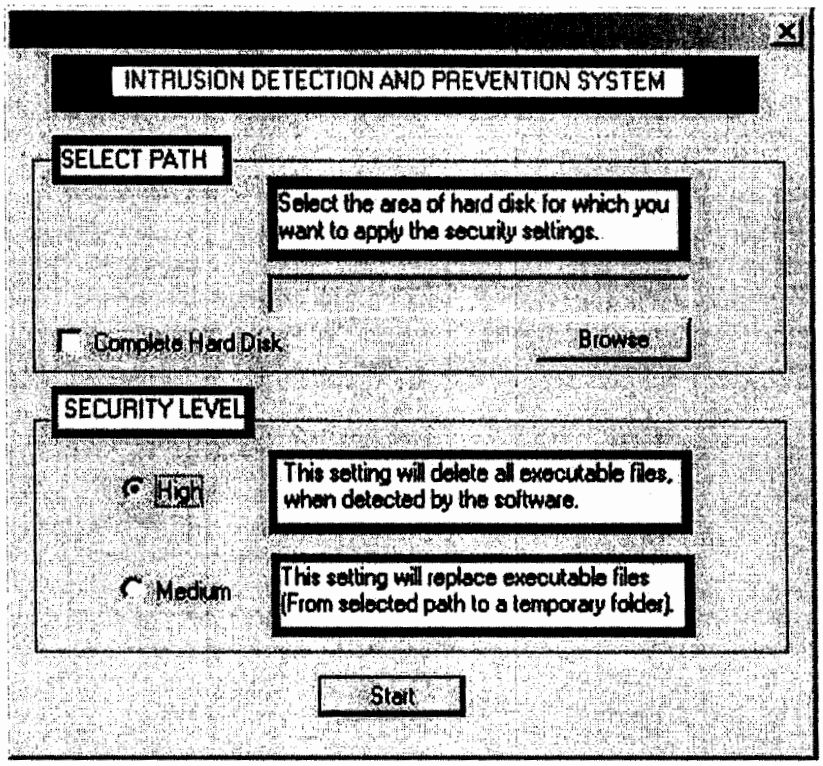

Figure A.3 the Settings Dialogue Box

Scan for Executables: This button is used for scanning the exe files which might exist on the system or might come into the system via network.

Exit: Before the user can exit from the software the software should be halted first. This is done by clicking on the Capture button. Then the Exit button would be activated and now the software is ready to be closed

Software in Working Mode

Once the capture button is clicked the software would now be in the working condition or in other words it would be activated to detect any sort of intrusion. The software converts the network card into promiscuous mode. This mode indicates that the MAC-Filter on the network interface is turned off, so accepting all data on the wire. Between network segments, normally the IP-address is used. But within one network segment, computers find their counterparts by using MAC-addresses. Also all packets

that arrive via the router of your network segment, find there recipient by his MAC-address.

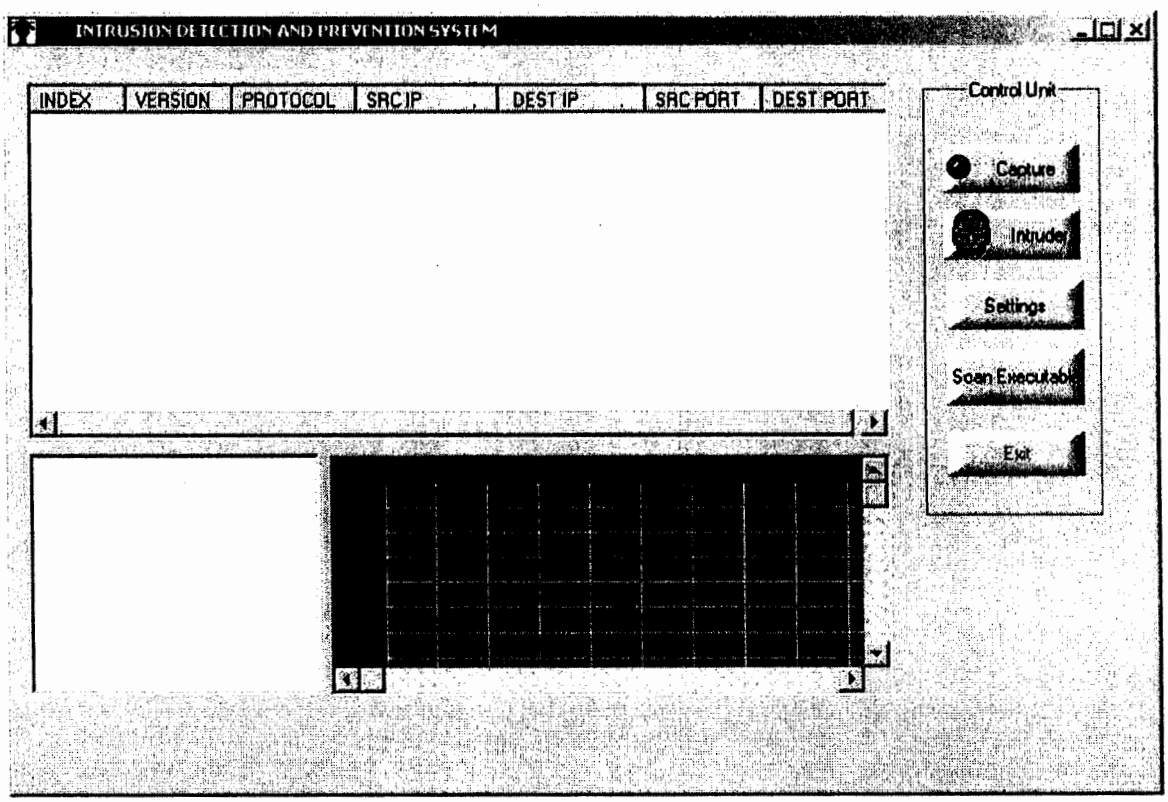

Figure A.4 Capture Mode

Here the software is now in capture mode. It is now activated on the network and is ready to capture all sorts of intrusions.

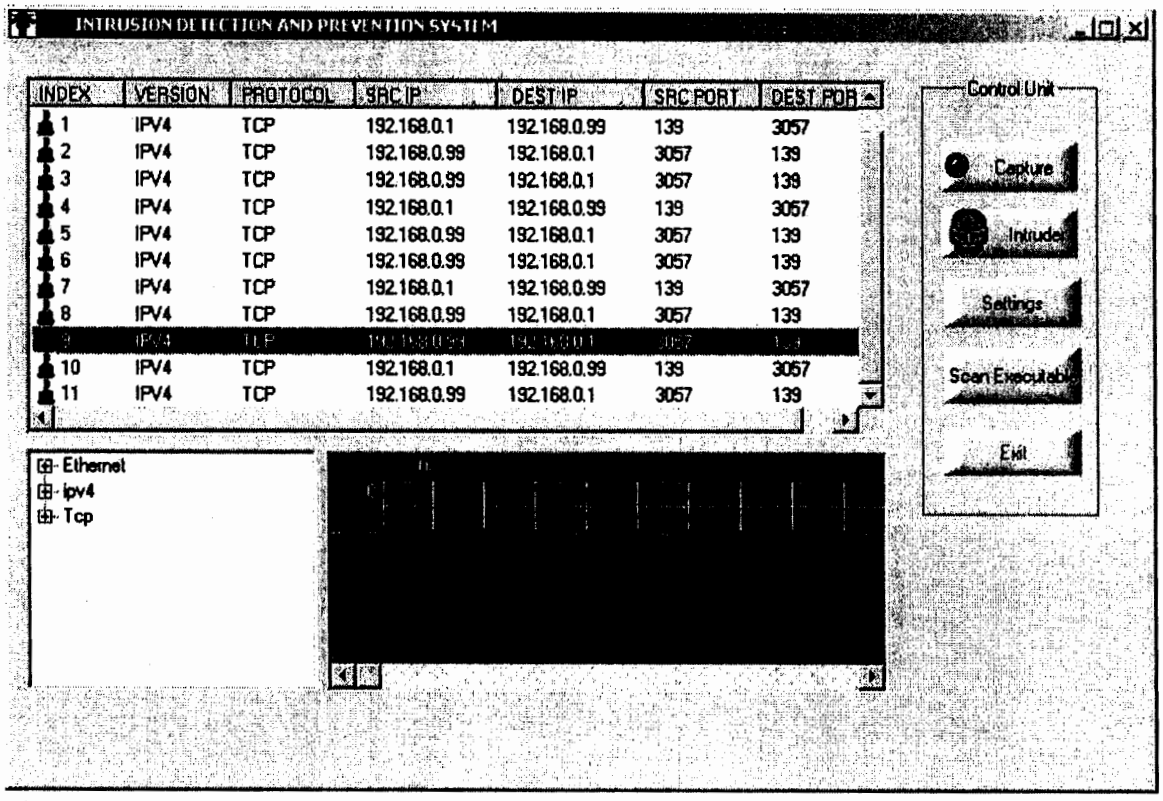

Figure A.5 Software in Working Mode

As it is shown in the Figure A.5 the software is now in working condition. It is capturing all the data which is passing through it. This software reads the IP Header, TCP Header and the Ethernet Header and displays the information on the bottom left corner box.

Intrusion Detection

Once the software captures any sort of virus or any file which is a threat for the computer it will prompt to the user about it and give the name of that particular file. It will give the complete name of the file along with its signature and the port from where it has entered into the computer.

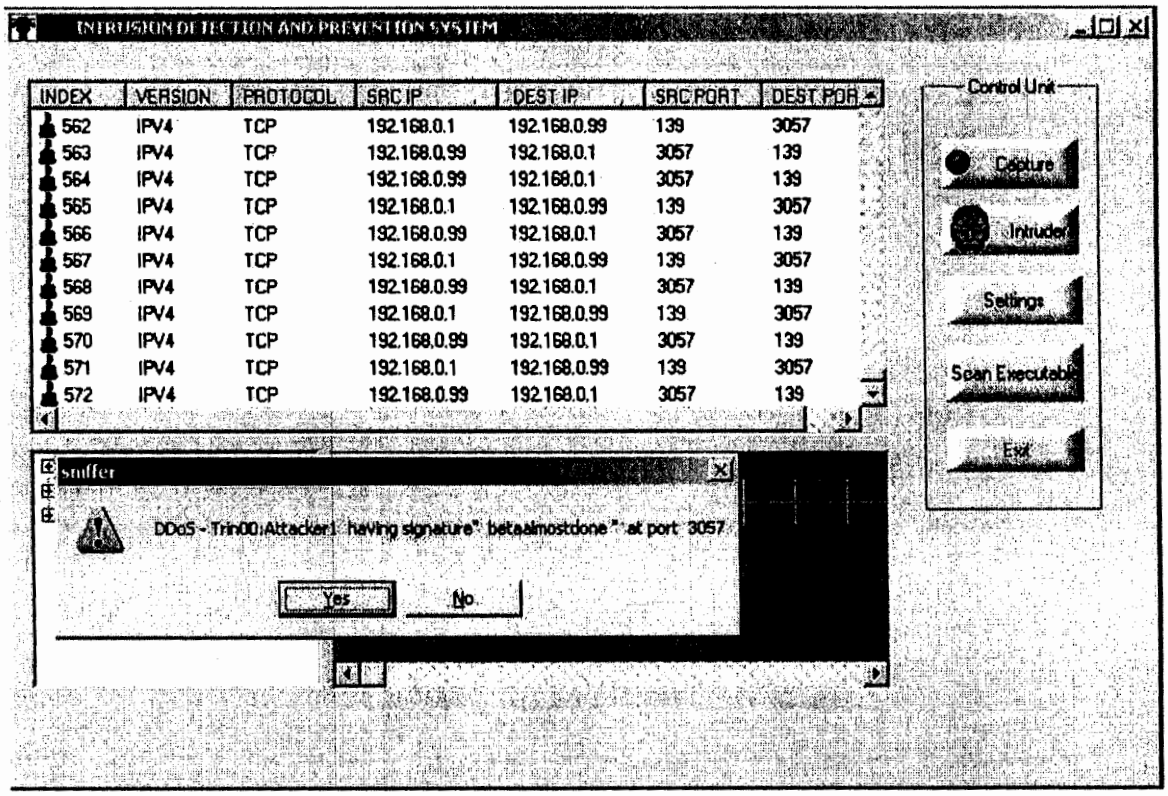

Figure A.6 Virus Caught

The Index which is assigned to this particular file would turn red indicating that it's a dangerous file.

Checking the Red Index

Once the communication between the two computers has been completed we can see how many indexes is RED in it. To see the name of that particular file which was assigned a red index all we have to do is double click on that index and it will display its name to us. Further more the Skull on the right hand side would change its color indicating that the file is a dangerous file for the computer.

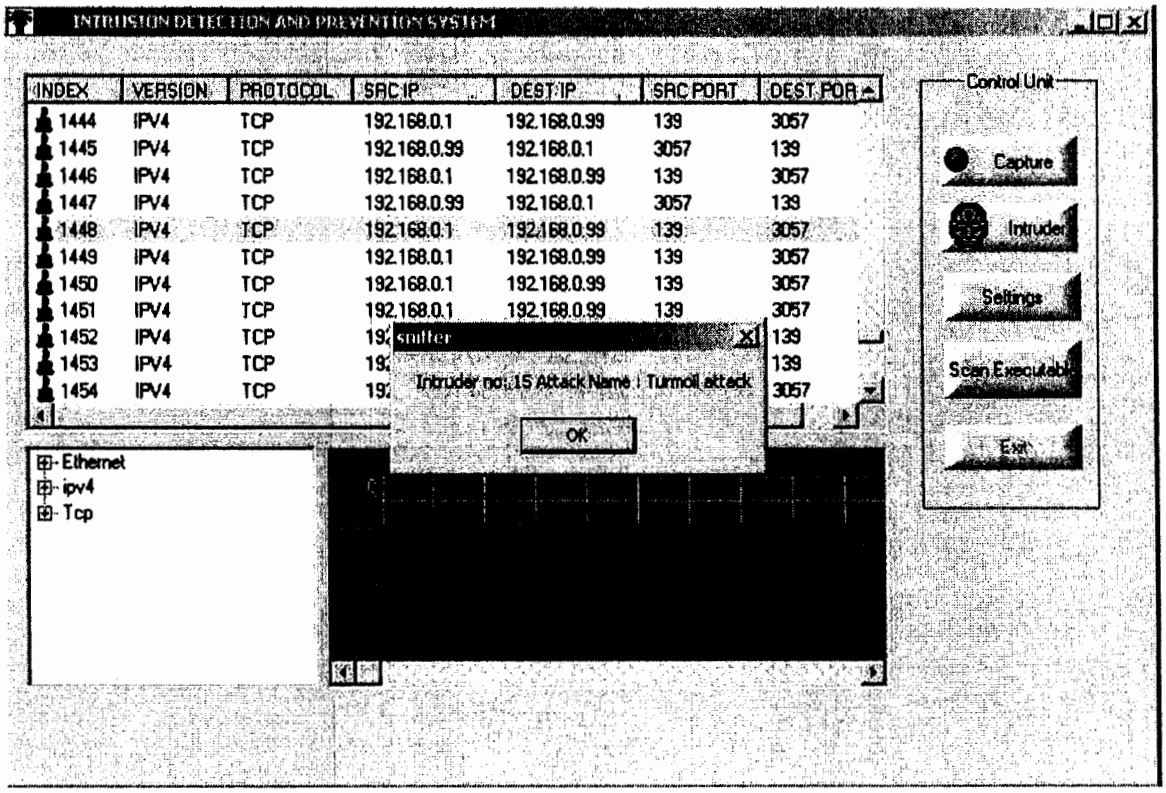

Figure A.6 Showing the Name of the Intruder

Appendix B

MSN Hotmail [redacted] Today | Mail | Calendar | Contacts

matrix_sam@hotmail.com Messenger: Offline Free Newsletters |

Reply | Reply All | Forward | Delete | Junk | Put in Folder | Print View | Save Address

From : <editor@eurojournals.com> | | | Inbox

Sent : Thursday, July 21, 2005 7:56 AM

To : matrix_sam@hotmail.com, editor@eurojournals.com

Subject : your article

ï»¿

Hello ,

This is a request for payment from EUROJOURNALS INC.

Billing Details:
www.EuroJournals.com Sale
Number - 00032
Bill Amount - 89 US Dollars
Comments: Your article has been accepted for publcation and will be processed upon the receipt of your publication fee.

Your secure order will be processed by 2Checkout.com Inc.

You will have the option to pay in the currency of your choice during checkout.

Please use the following URL to make your purchase.
https://www2.2checkout.com/2co/buyer/purchase?sid=353876&total=89&cart_order_id=00032

If the payment link above does not work properly with your email program, copy the url directly into your browser.

Please make sure that the entire url appears in your browser.

If you have any questions please contact editor@eurojournals.com.

| | | | | Inbox

2CO

Order Processed

Order Number 1990178705

A confirmation email detailing your purchase has been sent to **qjccie@yahoo.com**.

Charges will appear on your credit card bill under the name 2CO.COM *JOURNAL.

2CO.COM
1785 O'Brien Rd
Columbus, OH 43228

Contact Us

Qty	Purchase Item	Price	Total
1	Cart 00032	$ 89.00 USD	$ 89.00 USD
	Total		**$ 89.00 USD**

Payment Information

Cardholder Name	Qaisar Javaid
Email Address	qjccie@yahoo.com
Phone	+92-300-5000-435
Address	Block 16, Flat 2, Cat III, I-8/1, Islamabad
City	Islamabad
State	XX
Country	Pakistan
Postal Code	44000

Previous | Next | Back to Messages Prir

Delete | Reply | Forward | Spam | Move...

This message is not flagged. [Flag Message - Mark as Unread]

Date: 22 Jul 2005 15:04:13 -0000

Subject: 2CO.COM Sales Receipt

From: "2CO.COM Sales Receipt" <no-reply@2co.com> Add to Address Book

To: "Qaisar Javaid" <qjccie@yahoo.com>

2Checkout.com Order Number: 1990178705

Quantity: 1
Product ID:
Product Description:
Cart 00032 ($ 89.00 USD)

Total: $ 89.00 USD (US Dollars)

The order was billed to:
Qaisar Javaid
qjccie@yahoo.com
+92-300-5000-435
Block 16, Flat 2, Cat III, I-8/1, Islamabad
Islamabad
44000
Pakistan

Charges will appear on your credit card bill
under the name "2CO.COM *JOURNAL".

Distributed By:
EUROJOURNALS INC (http://www.eurojournals.com)

2Checkout.com (2CO) and EUROJOURNALS INC thank you for your

2Checkout.com Inc
1785 O'Brien Road
Columbus, OH 43228
Contact us: http://www.2checkout.com/contact

##

Please do not reply to this email. Your response wil
recieved.
Please use our Customer Service number or visit our w
to contact us about your order.

##

Distributed Security Cluster(Network Firewalls)

Ahmed Salman Mirza* Qaisar Javaid* Prof. Dr. Khalid Rashid Dr. S. Tauseef-Ur-Rehman**
salmannnn@yahoo.com qiccie@yahoo.com Khalid@iiui.edu.pk stauseef@iiui.edu.pk

* FAS
International Islamic University

Abstract This paper presents the design architecture of distributed security cluster of computers to be saved from the malicious attacks of intruders. Intrusion detection has traditionally been performed at the operating system (OS) level by comparing expected and observed system resource usage. OS intrusion detection systems (OS IDS) can only detect intruders, internal or external, who perform specific system actions in a specific sequence or those intruders whose behavior pattern statistically varies from a norm. It describes the new way of representing the *Network firewalls* on the network hosts which would enable the safety of all the nodes which are present on the network along with the safety of the whole cluster. This approach of distributed security enables the network to be safe from all sorts of attacks may it be from inside or outside of the network.

1. Introduction

Security is a critical issue in reliable network computing. The security of a network is at stake when the network is exposed to people through intranet. Such networks can easily be affected by or destroyed by the unauthorized intruders. Private networks use enclosed network to protect them selves from the intruders. Single gateway firewalls are use to repel malicious attacks. In such gateways all the network nodes are considered as trusted where as it distrusts all the external hosts.

With the advanced techniques used by the hackers, external attacks can easily immerse into the gateway and become a threat from inside. If the attack is initiated from the internal node of the network all the hosts compromise each other by domino effect. A hybrid firewall vendor (one that provides both proxies and stateful packet inspection) has a T.120 proxy that enforces controls on what specific T.120 services are allowed [30]. Here we propose a distributed *Network Firewall* to solve the security problems threatening the nodes of the network. A new type of cluster based intrusion detection algorithm, unsupervised anomaly detection, which trains on unlabeled data in order to detect new intrusions [23]. To improve efficiency, the computational costs of features are analyzed and a multiple-model cost-based approach is used to produce detection models with low cost and high accuracy. This network is protected by the gateway firewall from external attacks not internal attacks.

A seminal paper defining an early second-generation intrusion detection system implementation (IDES) appeared in 1987 [29].Research work in the past on fire walls architecture was concentrated mainly on the gateways [20]. Linux IDS [4, 8] was developed for the use in Linux systems. Mobile agents for network security can be found in [13]. For the purpose of developing firewalls and IDS software tool have been developed. Bellovin was the first one who introduced the concept of distributed firewall [3]. Several effective data mining techniques for intrusion detection have been developed [24, 25, 26, 27] IDS have been suggested [22]. Another distributed approach was offered by the IDIP (*Intrusion Detection and Isolation Protocol*) [19]. Database and server reconfiguration methods were suggested for implementing adaptive security policies. This paper we have proposed dynamic security with Network firewalls, which are very well supported by mobile agents.

The network firewalls, distributed IDS design and the security infrastructure of IIUI is taken as an example. It includes the functional mechanisms built on the network firewalls in Linux kernel. Mobile agents or CORBA was used for the purpose of auditing records, anomalies detections and reporting intrusions. For the updating policy of security we have suggested the use of Java based RMI (*remote method invocation*) package for broadcasting the policy changes. In other words both network firewalls and CORBA perform distributed intrusion detection and Java based RMI is used to change the security policies dynamically when ever the pattern is changed.

2. Adaptive Security Over Distributed/ Network Firewalls

This section will focus on the adaptive and dynamic security in an intranet supported by distributed ID and responses. Before that we will have to see the problems associated with basic firewalls. Later on we will present the strength of distributed approach over the traditional firewalls.

2.1. Distributed Firewalls in a network

Bellovin suggested the use of distributed firewalls to remove some of the problems present in the network which are mentioned above. Distributed firewalls impose access controls on hosts rather then on the gateway of the network. The purpose of this is to achieve fine security processing with no restrictions on the network topology. Bellovin used IPsec policy language and Keynote trust management system. End-to-end encryption is used on all the hosts. The establishment of IPsec connections between all the hosts is a high runtime overhead. Network firewalls have a lower overhead and also a very low cost of implementation. This enables real time intrusion detection and response to it in an intranet or a cluster. Work on host based intrusion detection has been done. The key ideas of such host based systems are to initially train the model and then use it to detect malicious registry access [28].

Scalable infrastructure and recoverability from intrusions or system faults must be supported in a highly secured intranet domain. In addition to that the framework should also be able to monitor the software based agent's behavior to assure trusty and reliable operations.

2.2. Comparison between Distributed and Conventional Firewalls

Comparison of both distributed firewalls and conventional firewalls has been discussed in [9]. The things that were not done in the conventional gateway firewalls are present in the distributed firewalls. Both are used in a clustered environment to provide double protection and aggressive response to intrusions that might be from inside or out side of the cluster. Both provide adaptive ID, responses and enable dynamic security. They are more fault-tolerant, cost effective, scaleable and robust to protect exposed intranets.

2.3. Dynamic Security for Changing Patterns

Mostly the security policies implemented on clusters are basically static in nature. Such policies cannot be used with the dynamic changes in the threat patterns. Clusters cannot be done in real time and often involves an expert in the loop. Online-policy change, when new threats or new intrusions are detected is a demand of dynamic security. Dynamic security fortifies adaptive intrusion responses. Gateway firewalls alone, are unable to achieve dynamic security effectively. To cope up with the dynamic changing patterns, distributed firewalls should be used on each and every host individually.

Now a day, Intranets are mostly structured as client server clusters of PCs, workstations and servers. In such LAN based intranets some middleware and groupware are needed. A central security policy is used to govern the entire cluster.

2.4. Well minced Security of Cluster Network

Figure 1 shows the architecture of distributed network firewalls installed in a boundary of cluster networks. The administration of central security policy is distributed to all the nodes and is coordinated via policy manager. For the purpose of screening at the front end a gateway firewall is used between the cluster network and the external world. As all the hosts of the intranet acts as an enforcer of central security policy, all intranet nodes form a single security unit or domain. The distribution of security functions removes some of the constraints associated with the conventional firewalls used on the gateways.

Table 1 Security Functions in Three Defense Levels

Defense Level	Security functions and Actions Taken in Response to Detected Intrusions
Micro-Firewalls at Cluster Nodes	• Local screening of arriving packets and monitoring node events • Detect local incidents and anomalies to the IDS manager. • Maintain the access log and perform auditing duties • Implement the policy changes by manager with local responses
Policy Manager at the DMZ	• Supervise the node monitoring by sensor agents • Work with cluster nodes to detect intrusions from all sources • Update intrusion databases and change policies • Broadcast security policy changes to micro-firewalls • Stop the spread of detected intrusions
Gateway Firewall at the Front-end	• Filters boundary traffic between cluster and outside networks • Merge threat reports from nodes to yield global policy changes • Enable global intrusion response at the network boundary

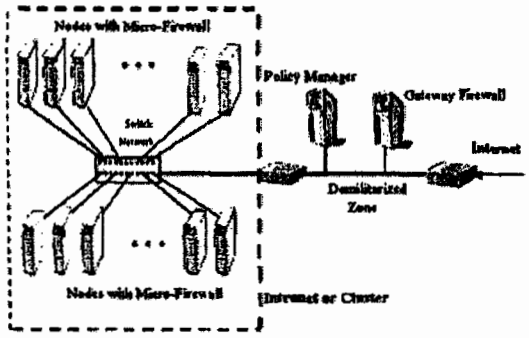

Fig. 1

The function of intrusion prevention, detection and responses are distributed to three levels of security control, which are gateway firewall, network firewalls and the policy manager. The main purpose and aim of this is to repel intrusions from all sources in real-time. Our focus is on protecting the intranet in file system, software processes, system administration and access control, etc.

2.5. Profound Defense

We consider clusters containing a large number of computer nodes that are open to frequent intrusions from all possible sources. The functional characteristics and accesses the structural complexities of security control. The roll of gateway firewall is of intrusion prevention. All the cluster nodes work with the policy manager to implement distributed IDS (DIDS). The network firewalls on the network reports abnormal situations and incidents to the policy manager. Trust is implemented with the RMI package.

The highest level of trust is available with the DIDS and can request policy updates to the policy manager. Intrusions are detected collectively by the nodes and are reported to the manager. The manager sends the policy update via RMI to invoke all nodes responses to avoid further spreading of the intrusion. In the Fig.1 we denote the MFW (micro-firewall) as level 3, the policy manager at level 2, and the gateway firewall as level 1. External attacks are treated differently from the internal attacks. At level I, 2 sets of screening rules, Si (external attacks) and Ti (internal attacks). Following the inclusion property S3 (S2 (S1 and T3) T2) T1. The complexities of the set at different levels are related to the number of interacting hosts.

3. Network-Firewall Built in Linux Kernel

It is constructed as a functional module at the kernel space. A network based sensor monitors the network traffic on a network segment, and is quite similar to a network sniffer. It consists mainly of a network adapter put in promiscuous mode [32], a TCP/IP stack implementation, and some programming interface to this stack, providing the filters with relevant traffic information. Our design and prototype implementation are given as follows. We used the IP chains for implementing packet filtering rules in the Linux kernel of all the nodes.

3.1. Security Responsibility

The Node level firewall, monitors local event and reports to the manager. The manager supervises the intrusion detected, plus it also informs all the other nodes of the policy change. The gateway implements global security policy and enforces global responses. The role of gateway is enlarged from conventional firewalls, because it should interact with the IDS manager to respond to intrusions.

3.2. Implementation layers

The network firewalls are implemented at the kernel level as shown in the Fig. 2. Sockets library are used at the next kernel level to interact with the JVM at the user level. At higher user levels, communication protocols and agent frame works are applied. Linux security tools, such as IP chains LIDS and log check are available for the purpose of implementation of DID and response systems.

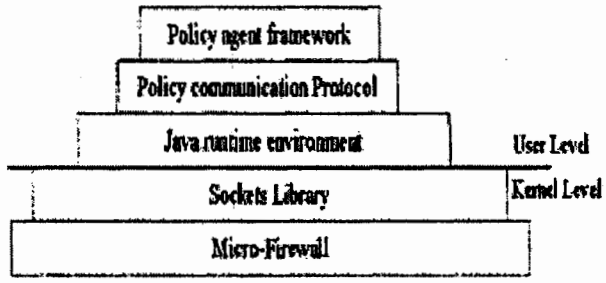

Figure 2

Fig. 3 shows the MFW as a functional module at the network access part of the operating system kernel. This module is placed in between the TCP/IP stack and system call interfaces.

Network cards are directly connected with the TCP/IP stack. There are three functional blocks of MFW module packet filter, access logging and anomaly detector.

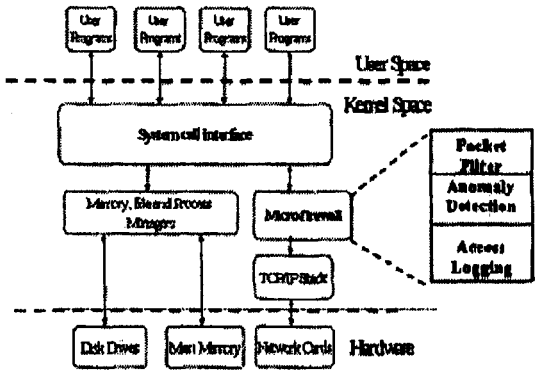

Figure 3

3.3. IPchains for Packet Sieve

Following three filtering rules input, and forward and out put are specified and enforced by the IP chains. The input rules are implemented on the packets as soon as they enter the hosts. Where as the forward rules are applied when they are inside the host and the output rules are applied when the packet is ready to leave the host. These three rules are very useful for screening all the packets that enter the hosts. Linux kernel is placed with built in packet filtering options in every host. The IP chain package is available in every host to enforce the rules that are carried by the policy agents. These agents are carriers of policy certificates containing the policy change. The policy certificates are verified and decoded by the policy interpreter, which translates the policy into scripts which can be specified by IP chains. This is then executed on the local host to make the specified policy updated to the kernel packet filtering module.

3.4. Executing LIDS for Anomaly Detections

A LID has been developed to reside in the Linux kernel, so it was planed to use the LIDS as the basis of the anomaly detection. To prevent the root from tampering with important parts of the system. The enhanced version of LIDS detects the unauthorized scans by intruders. Violations of local rules are recorded in a log file. The anomaly detection must avoid the increased over head in the logging, auditing and detecting processes. Cluster administrators spot strange login behavior and quickly responds. The treat patterns are then utilized to detect unusual events. We used multi layered approach for intrusion detection on each node.

The nodes are protected with each layer, with additional data protection. For example the top layer is used for packet filtering and the second layer port for entry, and third for optional LIDS protection and the last layer for log checking.

3.5. Log Examination for Auditing

Access logging demands more protection against direct port access, memory access or raw disk access. A log check program is available for the Linux hosts. It automatically monitors the system logs and mails the violation to the user periodically. It sniffs changes in the firewall rules and takes actions accordingly. User can be shutdown if any violation of rules is found.

There is a port scan detector that takes active stance to shutdown attacking hosts while it also notifies the administrators. The attacking hosts are denied accesses to the cluster by dropping of local routes.

4. Agent based DID

The structure of DIDS is shown below in fig.4. It is built on top of individual LIDS on each network node. Mobile agents are developed to provide intrusions between the cluster nodes and central policy manager, where the core of the DIDS is located

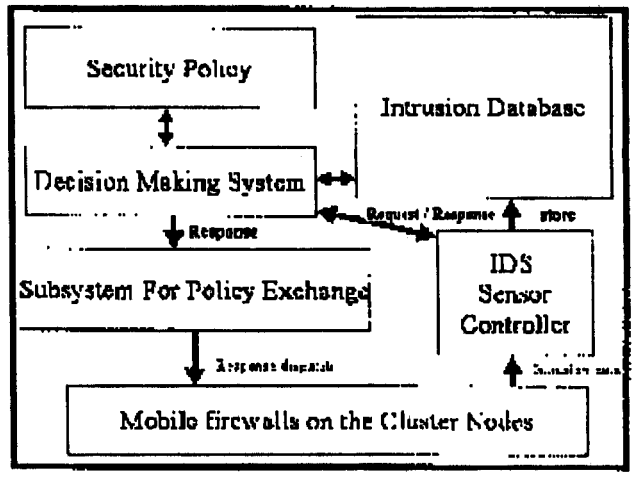

Figure 4

4.1. Mobile Agents

They have been actively explored for enforcing computer security and for distributed AI applications [10] [14]. There were chosen because of their number of advantages for building DIDS: To over come the latency problem, to execute asynchronously and autonomously, the agents can be made adaptive to any environment change; they require no coordination once dispatched. However these agents also have the following main disadvantages: they rely on authentication and encryption to protect their own security [5], and are written in slow interpreted language for portability and security.

4.2. Foundation of DIDS

Figure 4 describes the core architecture of the security manager, which is an agent-based ID [1]. This manager host maintains a central security policy. This policy can be dynamically changed. The IDS sensor controller is built with an agent name system (ANS) for establishing a unique name space for all the agents. This ANS controls the access of the agents and eliminates malicious agents from the system. The ANS needs a simple public key infrastructure (PKI) to enable secure communication and agent authentication.

The intrusion database is created to keep track of creation, entries, exits and activity records of the agents. The policies are established depending on the levels of security required, which is used during the decision making processes. The decision making subsystem determines the timing and countermeasures to betaken. This subsystem operates adaptively with respect to the changing threat patterns.

4.3. Agent Sanctuary and Discretion

Mobile agents executed on the cluster nodes which it visits. An agent server runs on each node by creating a trusted computing base (TCB). The work of TCB is to provide CPU resources and memory to execute it. It also provides local security by using encrypted functions and black box mechanism [18]. The agents also carry some integrity information generated inside the TCB; nodes executing the agents verify this integrity information. The agent uses the public key mechanism to make sure the identity of the new TCB before it starts execution.

Using both methods the agent process must be very robust and available all the time. If any agent is killed the TCB replicates that particular agent, if required. The TCB also limits the usage of recourses by agents and prevents any denial of services attack on the host. The focus of our agent is on three goals: integrity, availability and confidentiality. These can be built as lightweight soft ware processes. The agents should be immune to any modification by the malicious hosts.

Confidentiality of the agent process requires that only authorized hosts have the access to the program and data carried by the agents. This is achieved using both TCB and the agents. The agent autonomously generates an itinerary of hosts which it will visit. This is based upon its findings on a certain host regarding an intrusion. The agent then visits all the hosts to reveal the suspected intrusion. The PCB encrypts with the public key of the host which the agent visits next. In this way the contents of the agent can not be viewed or modified by the host.

4.4. Feeler Agents

These agents contain decision-making system which consists of sensing mechanism, and interaction sequence and a small state memory. The feeling requires the agent to check with local hosts about suspected intrusions. The host intrusion sequence tells what the agent is supposed to do during its visits. The state memory stores the state information. This memory is used to keep account of various interactions and also list of hosts that it has to visit in order to make policy update. The communication subsystem moves the agents physically from one node to another inside the cluster. As shown in Fig. 5

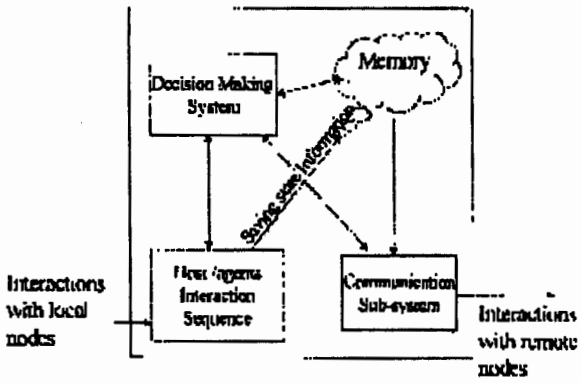

Figure 5

Agent based security systems introduce new problems such as management complexity (i.e. setting their itineraries, resending them when they die, destroying them if necessary). We have designed the agent management to be minimum and autonomous. The management is done at the initiation of the agents. The agents are mainly Java programs written to interact with network firewalls on cluster nodes to obtain data detect and verify intrusions.

The sensor agents carry a small state that contains information for auditing the execution of the agents. The agent detects an intrusion based upon the data obtained from the cluster nodes to initiate a response. The TCB allocates all the recourses required by the agent for interacting with cluster nodes. We used Java aglets for building sensor agents and sue the JRE (java remote environment) for the implementation of TCB on the cluster nodes.

5. Security Plan Update Mechanisms

For network firewalls three alternative mechanisms are considered for updating security policies. These mechanisms are assessed for the purpose of dynamic policy update.

5.1. Strategy Update Goals

Frequent policy change is the demand of dynamic security change. Network firewalls, gateways and managers are to be changed. They differ from each other in granularity and scopes. Their one primary goal is to maintain the critical mission functionality. At the host level, the security update must be done to avoid the risk of further penetration. These timely actions should not stop the critical processes that are taking place. At the security manager level, major goals in policy reconfiguration include the reloading of new security DB and the expedition of state and mode of operations. Inter managed communication among the hosts should be done to exchange the policy update information. The managers achieve distributed intrusion collectively and cooperatively. Further more they stop the wide spread of threats beyond each security domain. These managers also interact with the gateway to update the global policy.

5.2. Mobile Agents

Mobile agents have been actively explored for enforcing computer security and also for distributed AI applications. The reason why we preferred mobile agents for a number of reason, related to cost effectiveness, fast response time and easy to make policy changes. Mobile agents are dispatched from the manager to all hosts in a same domain. These domains automatically update the policy on the micro-firewall visited. Four major advantages are found for using mobile agents: help to overcome the network latency problem, can execute asynchronously and autonomously, can be made adaptive dynamically to the environment requirements and robust and fault tolerant. They are weak in speed, which has been accessed in table 2.

Significant progress has been made on the just in time complication software fault isolation, and other techniques which allows the mobile codes to execute nearly fast as natively compiled codes. Approximately all the mobile agent systems allow a program to remove freely among heterogeneous hosts. The agent code is compiled into platform-independent representation, as Java byte codes, at the target machine they are either compiled into native code upon its arrival. For the wide usage of mobile agents, the code must be portable across mobile-code systems. Significant standardization effort is required for making agent code portable across platforms.

5.3. CORBA Middleware

Object orientation is the basis of CORBA, by which applications are composed of objects, it combines data and functionality. Interfaces to the object are defined in IDL *(Interface Definition Language)*. This fixes the operations it perform and I/O parameters that are to be used. This maps to all programming languages with a high

degree of transparency. Interface definition language is compiled into client stubs and object skeletons. Skeletons and stubs serve as proxies for clients and servers. IDL defines interfaces so strictly, due to which the stub on the client side can communicate easily with the skeleton on the server side. This is true even when the two are compiled with different languages, or when they are running on different ORBs (Object Request Brokers).

In CORBA, unique object references are there for every object instance. Object references are used by the Clients to direct their invocations. The client works as if it is invoking an operation on the object instance, but it's actually invoking on the IDL stub which works as a proxy. Passing through the stub on the client side, on the object side the invocation continues through the ORB and the skeleton. The IIOP is a standard protocol for use in communication between different ORBs. This facilitates the client to invoke objects across platforms. CORBA does need coordination by the security managers and demands the ORB support in each host. However, the CORBA is faster than agents for requiring less execution time in each host. CORBA middleware is more secure for using the CORBASec. RPC-like semantics are needed to terminate the policy update process. The advantages of using CORBA for policy update include the provision of a multi-language and distributed object infrastructure, multi-platform environment, network transparency, and location transparency.

5.4. Remote Method Invocation (RMI)

RMI offers another alternative mechanism for policy update. The use of RMI is for the purpose of updating security polices on network firewalls installed on the cluster nodes. Let us initially enumerate the aims of dynamic policy update. Then implement the policy reconfiguration, which applies to all network nodes under the supervision of the policy manager.

RMI is also a Java based model used for distributed computing using java objects. In other words we say that RMI is java's RPC (remote procedure call). RMI is more efficient then the traditional RPC system because it is part of java's OO approach. The normal RPC systems are language neutral. They can not provide functionality that is not available that are not available on all the target platforms. The RMI can take a direct, natural, and fully-powered approach providing users with a distributed computing technology that allows one adds Java functionality in an increasing and seamless manner.

Table 2. Comparison of Agents, CORBA, and RMI for Security-Policy Update

Capabilities	Mobile Agents	CORBA Middleware	RMI Middleware
Central policy coordination	Agents are autonomous and require no coordination once dispatched	The policy manager in each domain coordinates all communications	The policy manager acts as the RMI registry and coordinates all communications
Reaction time to policy change	The time increases with the number of agents dispatched.	Faster than agents or RMI to react to a policy change	RMI is slower than CORBA and is faster than agent based system for policy updates
Hosts fortified with micro-firewalls	Agents carry most mechanisms required to update security policy	Requires the ORB middleware support on all hosts in the Internet	Requires JVM to be present on all the hosts.
Security Mechanisms	Use authentication and encryption. Still prone to attacks from malicious hosts.	Security is implemented with the CORBASec.	Security is the best among the three, implemented with the Java sandbox model.
Update Process Termination	Multiple agents are used autonomously, Policy update will always be completed.	Implemented at application level using RPC-like semantics	Implemented at application level using RPC-like semantics

The very basic advantages of using RMI for security policy update lie in object orientation, high mobility being safe and secure, and support of parallel processing.

5.5. Comparison of Three Mechanisms

The comparison of three mechanisms is shown in table 2 for the purpose of security update policy. Their comparison is based upon four functional features: namely the ability of central policy coordination, security of the mechanisms, response time, and process termination. Entries in the table show the fact that agents are autonomous, require no coordination after dispatch, and always terminate in the policy update process. Agents rely on authentication and encryption to protect its own security. Due to which the agents are easier to be attacked by other agents or hosts. Network latency and bandwidth of mobile-agent systems save at the expense of imposing higher workload on their hosts. Agents are often written in a slow interpreted language for portability and security reasons. Upon arrival agents must be injected into an appropriate execution environment. Agents may take longer time to accomplish tasks, since the timesavings from avoiding intermediate network traffic is less than the time penalties from slower execution.

5.6. Relative Strengths and Weaknesses

The above mentioned policy update mechanisms are rated in terms of their operating speeds,

security base, scalability, and robustness. The mobile agents are strong in scalability and robustness. The number of active agents can be scaled from top or bottom. The agents are created, reborn, suspended or terminated dynamically. The major weakness of agents lies in its high overhead and payload plus lower security base. Thus agents are easily attachable by other agents or hosts.

CORBA has its major strength in its high speed and lower overhead experienced. Application, which requires high speed in policy update CORBA, is best suitable. But it is weak in scalability and robustness. The RMI has the highest base security among the three. The main reason behind is, Java security, which is based on the sandbox model. And it is relatively poor in scalability and robustness as CORBA. When talking about speed then it is ranked in the middle amongst the three.

5.7. Policy Update Using RMI

As shown in figure 6, policy distributing using RMI has three components: remote method, stubs and skeleton, and the RMI registry. The RM does the interface provided for policy updates to be done on the host with the micro-firewall. The interfaces are implemented using Java and RMI packages. JVM is used to run the interface and make policy update on local firewalls. The program is available on the host and only the parameters required for policy update are transported by the RMI across the network.

The Stubs and the Skeleton provide the desired protocol to the host and the policy manager to interact with each other. It gives the protocol for the hosts to register its RM with the RMI registry. The stubs play as communication interfaces for the host and the policy manager when the remote methods are invoked in the hosts. The database of all the hosts is provide by the registry and also provides the mapping of all the hosts and the remote methods. The registry implements some authentication procedures to allow only authorized access as shown in the diagram.

We have also seen the four additional communication mechanisms: Email, SNMP, HTTP, and RPC for policy update. All of them are not suitable for use in implementing dynamic security for Intranets. Weaknesses of these mechanisms lie in the fact that all these 4 mechanisms are basically insure themselves have poor scalability and higher overhead to use, compared with agents, RMI, and CORBA.

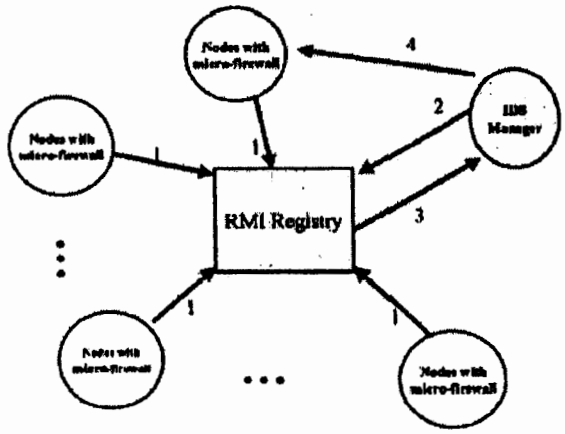

Figure 6

6. Vibrant Rejoinders to ID

To illustrate the process of dynamic intrusion rejoinder a cluster fortified with network firewalls and distributed IDS was considered. We assess the effectiveness of the adaptive responses to such DID. An example is shown in figure 7 to show the response to an attack in 3 steps: In Step I, an external attack has penetrated from the gateway firewall and hide in a network host at the upper right corner. Intruder program is now ready to attack from the node. In Step 2, the intrusion is detected by the micro-firewall locally; an agent is dispatched to alert the IDS at the manager with the intrusion record. In Step 3, the manager works a policy change to interact with the intrusion.

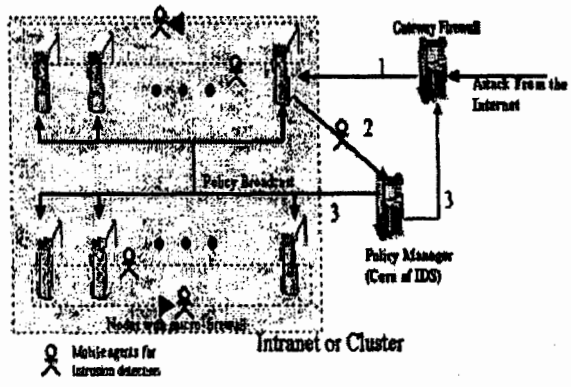

Diagram 7

6.1. Disseminated Intrusion Detection

Out of the three any can be used for policy update. If security is the main concern, then we would suggest the use of RMI over CORBA and agents. The RMI process is initiated and broadcast to all the network nodes to make changes in all the network firewalls along with the gateway firewall. If the objective is scalability then we would suggest the use of agents. It depends upon the network size; we can send multiple sensor agents around the network nodes to make appropriate changes. In the diagram 7 we have shown two visiting loops in dashed lines for multiple agents to pass through. The bigger the network the more agents would be required to dispatch to sense the results of local detection. The policy change is handled by the RMI, because it takes less overhead and more secure themselves. With more improvements in base security CORBA could perform even better than the agents policy update.

6.2. Usefulness of Distributed Security

The usefulness of using the network firewalls in Intranet for proactive intrusion response is described below. The security affects the cluster hosts, manager and gateway hosts in the DMZ. The assessment is based on checking the design objectives against various threat categories from inside or outside of the Intranet. At the time of this reporting, this is mere a qualitative assessment, based on limited simulation results, a comprehensive quantitative evaluation backed by benchmark results.

7. The state of IDPS technology

The state of IDPS / Fire walls technology is immature if you define it as a single vendor, all-encompassing product that detects, monitors, prevents, updates, and reports on every transmission for in-bound and out-bound access through a particular network choke-point. Recently, enterprises have spent millions of dollars on products to help them secure their networks. Today's newly emerging IPS products are focused almost exclusively on Port 80 and so they are not replacing existing systems. They are instead augmenting them. An all-encompassing multi-protocol firewall solution will have to be developed and proven before any such systems would be taken seriously as actual 'replacements' for already deployed systems.

7.1. Long term goals

In the future, an inline security gateway solution should achieve these goals.

- The ability to detect and prevent attacks based on logical or physical use of multiple enforcement technologies. Broadly, this includes the ability to prevent both known and to some degree unknown attacks using *Application Defenses*.
- The ability to interoperate with deployed security infrastructure for the purposes of supporting data collection, electronic evidence, surveillance, and regulatory compliance as needed.
- The ability to not disrupt business operations because of lack of availability, poor performance, false positives, or inability to interoperate with required authentication infrastructures.
- The ability to support IT Security professionals in delivering their organization's risk management plan, which includes the cost of implementation, operating, and work outcomes from the alerts and reporting from the system.

7.2. Challenges to reaching these goals

- There are currently no acceptable third-party ROI studies demonstrating the efficiency of IPS as a solution. The market hype surrounding *Intrusion Prevention* is confusing what the technology can really provide versus what it promises.
- The capabilities required to build a complete *Intrusion Prevention* system do not all currently reside within the same technology segments (vendors), which will require industry integration and consolidation. The multi-layered approach to IT security continues to be validated as the industry evolves. It does not appear that the migration is away from layered defense-in depth, just how it is organized.
- Many of the IPS solutions will require IDS-like human-power requirements for tuning, monitoring, and reporting. "There are still logs to parse through (if the administrator is doing his/her job), and there is still the need for 24x7 personnel responsible for the device, unless the

systems are powered off nightly (which is highly unlikely)."[31]

Conclusion

It has been very clear that the use of agents, CORBA and RMI are suitable depending upon their usage. All the three have been clearly specified in the paper. To make a distributed network more secure and more reliable and safe from and safe from any sort of intrusions, network firewalls should be installed on all the hosts of the cluster. Its authentication and verification is clearly specified in the paper. Where as its complete effectiveness can be verified easily by benchmark experiments.

Future Work

People bound by organizational roles and work culture select security solutions in various ways, but are always restricted by time and budget. Currently, there is no workable one-size-fits-all product that meets broad market needs at a level where it could *replace* existing firewall, Network Intrusion Detection System (NIDS), layer 7 switches, and other components that may (or may not) become the inline security gateways of tomorrow. However, if one such product appears, it would need to meet a significant portion of the goals discussed previously in this document, including *Application Defenses* capability. What's next? Evolution is not something that is generally predictable many steps in the future. Go back to step 1: the *threat-countermeasure cycle.*

Future threats, unknown to us today, will drive the direction of our future solutions. There may be new threats and new system vulnerabilities discovered that may affect the *Intrusion Prevention* security concepts of today in fundamental ways; or maybe there won't be. But, *Intrusion Prevention Systems* evolution is *most likely* to be a gradual merging over time of various security concepts into one true *Application Defenses* model. Don't be surprised if it ends up being in your tried-and-true hybrid firewall. So, stay tuned for the future.

References:

1. M. Asaka, S. Okazawa, A. Taguchi, and S. Goto, A Method of Tracing Intruders by Use of Mobile Agents , INET'99, June 1999.
2. J. Balasubramaniyan, J. O. Garcia-Fernandez, E. H. Spafford, and D. Zamboni, An Architecture for Intrusion Detection using Autonomous Agents , Coast TR 98-05, Department of Computer Sciences, Purdue University; W. Lafayette, IN. 1998.
3. S.M. Bellovin, Distributed Firewalls, Journal of Login, Nov 99, pp.37-39.
4. P. Biondi, Linux Intrusion Detection System (LISD) Specification, www.lids.org/documents.
5. D Chess, "Security Issues in Mobile Code Systems" , Mobile Agent Security, G, Vigna (Ed.), Lecture Notes in Computer Science , Vol.1419,1998, Springer, pp 1-14
6. W. R. Cheswick and S. M. Bellolvin, Firewalls and Internet Security Repelling the Wily Hacker, Addition Wesley, Reading, MA. 2000.
7. COBRA Specifications http://www.omg.org/technology documents/formal/corbaiiop.htm.
8. Elson, "Focus on Linux; Intrusion Detection on Linux" , source http;//www.securityfocus.com/focus linux/articles/linux-ids,html
9. M. Gangadhran and K. Hwang. "Internet Security with Micro-Firewalls and Mobile Agents for Proactive Instruction Response'', IEEE int' t Conference on Computer Networks and Mobile Computing, Beijing, China. October 16-19 2001.
10. G.Helmer, J. Wong, V. Honavar and L. Miller. "Intelligent agents for ID," IEEE information technology conference
11. K. Hwang, H. Jin, and R. Ho "RAID-x" Proc. Of 9th IEEE international symposium on high performance distributed computing. August 1-4 2000.
12. S. Loannids , A. D. Keromytis, and S. M Bellovin "Implementing distributed fire walls" 1-4 November Athens Greece.
13. W. Jasen, P. Mell, T. karygiannis and D. Marks "Applying mobile agents to ID and response" NIST.

14. B. Lange and M Oshima "Mobile agents with java: the Aglet API", WWW Journal 1998
15. M. Petkae and B lee "security agility in response to ID " USA. December 11, 2000
16. Psionic Software Inc. "The Logcheck package"
17. IPchains Http://www.linuxdoc.org/howto/IPCHAINS-howto.html
18. T. Sander and C.F, Tschudin, "protecting Mobile Agents against Malicious Hosts" ,in Mobile Agents and Security, Vigna, Ed., Vol 1419, Springer Verlag, 1998,pp. 44-60.
19. S. Schnackenberg, K. Diahandari, and D. Sterne, "Infrastructure for Intrusion Detection and Response" , DARPA Information Survivability conf. and Exposition (DISCEX) , Jan 2000
20. R. N Smith and S.Bhattacharya, "Firewall Placement in a Large Network Topology" , Proceedings of the 6th IEEE Workshop on future Trends of Distributive Computing Systems (FTDCS '97), 1997.
21. S.Snapp, J. Brentano, G, Dias, T, Goan, T, Heberlein, C, Ho, K, Levitt, B, Mukherjee, S.Smaha, T, Grance, D, Teal, and D, Mansur, "DIDS (Distributed Intrusion Detection System) – Motivation, Architecture, and an Early Prototype." Proceedings of the 14th National Computer Security Conference, Washington, D.C, October 1991.
22. Sun Microsystems Corporation, "Java RMI Specification" , http;//java.sun.com/products/jdk/1.1/docs/guide/rmi/spec/rmiTOC.doc.html.
23. Leonid Portony, Eleazar Eskin and Sal Stolfo "Intrusion Detection with unlabeled data using clustering"
24. E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *In Proceedings of the 1998 USENIX Security Symposium*
25. A.Ghosh and A. Schwartzbard. A study in using neural networks for anomaly and misuse detection. In Proceedings *of the Eighth USENIX Security Symposium 1999*
26. *Risk Management*, pages 50–56. AAAI Press, 1997. W. Lee, S. J. Stolfo, and K. Mok. data mining in work flow environments: Experiences in intrusion detection. In p*roceedings of the 1999 Conference on Knowledge Discovery and Data Mining (KDD-99)*, 1999.
27. C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: alternative data models. In *In 1999 IEEE Symposium on Security and Privacy*, pages 133–145. IEEE Computer Society, 1999.
28. Sal Stolfo, Frank Apap, Eealzar Eskin, Katherin Heller, Shlomo Hershkop, Andrew Honig and Krysta Svore "Detecting malicious software by monitoring anomalous Windows Registry Access"
29. Denning, D. "An Intrusion Detection Model." *IEEE Transactions on Software Engineering*, 13.2 (1987) 222.
30. Secure Computing Corporation. *G2 Firewall Admin Guide version 6.0*: 7-31
31. Firewalls Direct.com. *Glossary. http://www.firewallsdirect.com/store/glossary*
32. 'Intrusion Detection, network security beyond the firewall', by Terry Escamilla, ISBN 0471290009