Analysis of
Intrusion Prevention Methods

By

Hakan SEMERCİ

**A Dissertation Submitted to the
Graduate School in Partial Fulfillment of the
Requirements for the Degree of**

MASTER OF SCIENCE

**Department: Computer Engineering
Major: Computer Software**

**Izmir Institute of Technology
Izmir, Turkey**

March, 2004

We approve the thesis of **Hakan SEMERCİ**

Date of Signature

…… **29.03.2004**

Assist. Prof. Dr. Tuğkan TUĞLULAR
Supervisor
Department of Computer Engineering

…… **29.03.2004**

Assoc. Prof. Dr. Ahmet KOLTUKSUZ
Department of Computer Engineering

…… **29.03.2004**

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

…… **29.03.2004**

Prof. Dr. Sıtkı AYTAÇ
Head of Department

ABSTRACT

Today, the pace of the technological development and improvements has compelled the development of new and more complex applications. The obligatory of application development in a short time to rapidly changing requirements causes skipping of some stages, mostly the testing stage, in the software development cycle thus, leads to the production of applications with defects. These defects are, later, discovered by intruders to be used to penetrate into computer systems. Current security technologies, such as firewalls, intrusion detection systems, honeypots, network-based antivirus systems, are insufficient to protect systems against those, continuously increasing and rapid-spreading attacks.

Intrusion Prevention System (IPS) is a new technology developed to block today's application-specific, data-driven attacks that spread in the speed of communication. IPS is the evolved and integrated state of the existing technologies; it is not a new approach to network security. In this thesis, IPS products of various computer security appliance developer companies have been analyzed in details. At the end of these analyses, the requirements of network-based IPSs have been identified and an architecture that fits those requirements has been proposed. Also, a sample network-based IPS has been developed by modifying the open source application Snort.

ÖZ

Günümüzde teknolojinin hızla gelişmesi ve ilerlemesiyle yeni ve daha karmaşık (kompleks) uygulamaların geliştirilmesini zorunlu hale getirmiştir. Hızla değişen isteklere kısa sürede uygulama geliştirilme zorunluluğu, yazılımın geliştirilmesi sürecinde bazı aşamaların, genellikle de test aşmasının, atlanmasına ve dolayısıyla, hatalar içeren yazılımların üretilmesine sebep olmaktadır. Bu hatalar, daha sonra, saldırganlar tarafından bulunarak sistemlere sızma amaçlı kullanılmaktadır. Ateş duvarları, nüfuz tespit sistemleri, "honeypot" lar, ağ tabanlı antivirus sistemleri gibi mevcut güvenlik teknolojileri, sürekli artan sayıdaki, hızla yayılan ve bu saldırılara karşı sistemleri savunmada yetersiz kalmaktadır.

Nüfuz Önleme Sistemi, günümüzün iletişim hızında yayılan, uygulamaya özgü, veri odaklı saldırılarına karşı geliştirilen yeni bir teknolojidir. Nüfuz Önleme Sistemi, mevcut teknolojilerin evrim geçirmiş halidir; ağ güvenliği için yeni bir yaklaşım değildir. Bu tezde, bilgisayar güvenliği ürünü geliştiren çeşitli firmaların mevcut Nüfuz Önleme Sistem gerçekleştirimleri detaylı olarak incelenmiştir. Bu inceleme sonucunda, ağ tabanlı Nüfuz Önleme Sistemlerinin gereksinimleri belirlenmiş ve bu gereksinimlere uygun bir mimari önerilmiştir. Ayrıca, açık kaynak kodlu Snort yazılımı üzerinde değişiklikler yapılarak örnek bir ağ tabanlı Nüfuz Önleme Sistemi geliştirilmiştir.

TABLE OF CONTENTS

ABSTRACT iii
ÖZ iv
TABLE OF CONTENTS v
TABLE OF FIGURES vii
TABLE OF TABLES vii
CHAPTER 1 – INTRODUCTION 1
CHAPTER 2 – BACKGROUND AND OVERVIEW 4
2.1 Security Approaches 4
2.1.1 “I Can See the Future” 4
2.1.2 “The Fall Guy” 5
2.1.3 “You Should Know Better” 6
2.2 Phases of attacks 7
2.2.1 Reconnaissance, Assessment and Strategy 8
2.2.2 Exploitation and Invasion 10
2.2.3 Maintaining Access 11
2.2.4 Operations 11
2.3 The State of Perimeter Security 14
2.3.1 Firewalls 14
2.3.2 Intrusion Detection System 20
2.3.3 Honeypots 31
2.3.4 Network-based Antivirus Systems 34
CHAPTER 3 – INTRUSION PREVENTION SYSTEMS 38
3.1 Definition 39
3.2 Deployment 41
3.2.1 Network-based IPS: 41
3.2.2 Host-based IPS: 43
3.3 Types of Existing IPS Products 46
3.3.1 Inline Network Intrusion Detection Systems 46
3.3.2 Layer Seven Switches 49
3.3.3 Application Firewall/IDS 52
3.3.4 Hybrid Switches 55

3.3.5 Deceptive Applications 56
3.4 Network-based IPS 59
3.4.1 Implementation Challenges 59
3.4.2 NIPS Detection Methods 60
3.4.3 NIPS Responses 73
CHAPTER 4 – NIPS SPECIFICATIONS 75
4.1 NIPS Requirements 75
4.2 NIPS Architecture 78
4.2.1 NIPS Sensor 79
4.2.2 NIPS Management Server 85
4.2.3 NIPS User Interface 87
CHAPTER 5 – NIPS IMPLEMENTATION 90
5.1 WinPcap Architecture 91
5.2 Snort Architecture 92
5.2.1 Packet Decoder 95
5.2.2 Preprocessors 95
5.2.3 The Detection Engine 95
5.2.4 Logging and Alerting System 96
5.2.5 Output Modules 96
5.2.6 Snort_inline 97
5.3 A NIPS Implementation 98
CHAPTER 6 – DISCUSSION AND CONCLUSION 103
SUMMARY 105
REFERENCES 106

TABLE OF FIGURES

Figure 3.1: Out-of-band IPS installation 42
Figure 3.2: In-line IPS installation 43
Figure 3.3: Host-based IPS 45
Figure 3.4: Connections of a typical Network Intrusion Detection System 47
Figure 3.5: Installation of an inline NIDS 48
Figure 3.6: Location of an inline NIDS 48
Figure 3.7: Packet scrubbing 48
Figure 3.8: Layer seven switch as load balancer 50
Figure 3.9: Layer seven switches stop attacks based on predefined rules 51
Figure 3.10: Application IPS/Firewall 53
Figure 3.11: Policy based filtering of requests 55
Figure 3.12: Determination of legitimate traffic 57
Figure 3.13: Deceptive application returns “marked” response to the attacker 57
Figure 3.14: Top Layer Stateful Network Intrusion Prevention and Response Engine .. 63
Figure 3.15: Phase 1 – Identifying the reconnaissance activity 67
Figure 3.16: Phase 2 – Responding to the reconnaissance activity 67
Figure 3.17: Phase 3 – Blocking the attacker 68
Figure 3.18: Intrusion protection with Cisco Threat Response 71
Figure 4.1: Three-tier NIPS architecture 78
Figure 4.2: NIPS Sensor engine..... 81
Figure 4.3: NIPS Management Server..... 85
Figure 5.1: Components of WinPcap architecture 91
Figure 5.2: Components of Snort 94
Figure 5.3: Snort/WinPcap deployment..... 99

TABLE OF TABLES

Table 3.1 : Detection Methods 62
Table 5.1 : Structure of the PacketQueue 100

CHAPTER 1

INTRODUCTION

The current improvements in modern technology have enabled the use of computer systems in conducting business and in gathering and sharing information in corporations and academic institutions using the Internet. Today, banks make use of networks to perform its financial operations, hospitals have the records of their patients in databases, and many companies has been presented on the Internet, so that any user with Internet access is able to choose the product that he/she desires and buy it online. The data that is handled in this type of businesses should be saved from attacks.

The Transmission Control Protocol and Internet protocol (TCP/IP), which is the protocol that Internet and many of today's networks based on, was first developed in 1979. The primary focus was to ensure reliable communications between groups of networks connected by computers acting as gateways. At that time, security was not a primary concern due to the size of this Internet and that most of the users knew each other. However, the base technologies used to construct this network contained many insecurities, most of which still exist today. Due to a number of well reported attacks on private networks originating from the Internet, security has become a primary concern for organizations connecting to the Internet. Organizations need to securely conduct business and protect their data and computing from attacks. Such needs are heightened as businesses link geographically distant parts of the organization using private networks based on TCP/IP [OLMS97].

Nowadays, guarantee of secure communication is as important as the traditional computer and information security assurance. Information in transit (as messages) must be protected from unauthorized release and modification, and the connection itself must be established and maintained securely. Prevention of illegitimate traffic is one of the goals of communication security and seeks to prevent an eavesdropper from gaining any meaningful information about network users' behavior or objectives by observing the legitimate traffic on the network.

To protect the enterprise, security managers have deployed a variety of technologies. While these technologies are useful for defending corporate assets, they have limitations. For example, firewalls may be configured to block certain types of traffic, but attackers still find ways to exploit legitimate traffic types to mount their attacks.

Intrusion detection presents its own difficulties. Intrusion detection systems (IDSs), mostly, detect attacks that fit an established pattern or "signature." This leaves the network vulnerable to new, undocumented attack strategies. IDSs also tend to yield a large number of false positives – thereby wasting staff time and eventually causing a real attack to be ignored. Other types of anomaly recognition systems are similarly prone to generating false positives, since they trigger alerts whether a deviation has an innocuous or a malicious cause. Finally, intrusion detection and anomaly systems are reactive; the action against an attack is taken as it occurs by resetting TCP connections or requesting a firewall rule change, which are mostly not fast enough to prevent the attack.

Some organizations have also deployed so-called "honeypots" to lure potential attackers away from the enterprise and document attack attempts. Honeypots, however, can't fully guarantee that they will be the target of the next attack rather than the enterprise network.

As a result, these technologies are insufficient to prevent new application-specific, data-driven, rapidly-spreading attacks, thus computing resources of the organizations are being penetrated every day. Intrusion Prevention System (IPS) architectures serve as the next generation of network security software that proactively strengthens networks and desktop computers against damage from the cyber-attacks by blocking them in real-time.

"Intrusion Prevention is gaining visibility in corporate and government organizations due to the inherent limitations in existing security technologies, as witnessed by the significant financial loss experienced by organizations in 2001. Intrusion Prevention can be thought of as the logical follow-on to signature-based technologies such as intrusion detection and antivirus, and to network-oriented

protection solutions such as firewalls" [DT02b]. In other words, Intrusion Prevention evolves from the existing security technologies; it is not a revolutionary new approach to network security.

Intrusion Prevention is becoming a very hot topic in the network security world. This thesis aims to provide a guide for the research field on "Intrusion Prevention Systems." The methodology of this thesis is to review the existing security technologies, define the new "Intrusion Prevention System" technology and provide a detailed analysis of existing methods applied by various vendors to eliminate the limitations of this technology.

This thesis is organized in six chapters. Chapter 2 provides background information about security approaches, the stages of an attack, and current defense methodologies against the attacks. In Chapter 3, IPS is defined, the classification of IPSs is provided and the types of existing IPS products are explained. Also the implementation challenges and various detection methods and response mechanisms implemented by many IPS vendors are explained. Chapter 4 identifies the requirements of a network-based IPS and proposes a scalable architecture with definitions of its components. Chapter 5 provides a network-based IPS implementation example which is a modified version of the open source IDS, Snort [Snort04]. In Chapter 6, discussions and suggestions are provided for future researches on IPSs.

CHAPTER 2

BACKGROUND AND OVERVIEW

2.1 Security Approaches

In the past few years, there have been discussions within the security community about the network security concept of protecting an information asset against unknown cyber attacks. As a result, several hardware and software vendors have announced products that attempt to make this vision a reality. There are three popular security approaches used today. The following section exposes strengths and weaknesses of those approaches.

2.1.1 "I Can See the Future"

In real world, nobody has the ability to see the future. Adequate estimates made from current data are possible and sometimes very helpful. Analyzing current and past trends allows us a degree of success in forecasting the future. This concept is used to try to pick investments, gamble on horses, and predict tomorrow's weather [eEye03]. This technique is also used by "Early Warning Systems" that offer advanced notification of threats based on whatever process they deem to be accurate.

Using this type of "crystal ball" technique, you can notice and respond to some events as they are happening, or shortly before they are anticipated to happen. This offers a limited degree of awareness, but often requires a precise view of a very specific data. In more technical terms, this concept relies on a bunch of distributed IDS sensors on a large network topology in an attempt to get the complete view of attacks. The advantages and disadvantages of this approach are summarized below:

Advantages:

- Utilizing the historical data and analyzing this data can yield predictions on new attacks,
- Analysis can yield predictions on spread rate of an attack within that network topology.

Disadvantages:

- General predictions often fall short,
- The further you attempt to forecast into the future, the more "hazy" things can seem.

2.1.2 "The Fall Guy"

This approach utilizes the fact that having a stand-in take the fall for you is often preferable to you taking a fall yourself. This concept shows up in the real world by the use of stunt men in the entertainment industry and by bodyguards in the physical security realm. The basis of this technique involves having someone act as a proxy or decoy for you with the understanding that they will be hit by an attack and not you. The piece in front gets hit (and potentially damaged) while allowing your infrastructure to continue to operate, sometimes at reduced capacity [eEye03]. This is often the case used in many of the proxy-type software applications that are available to protect your assets.

By acting as an intermediary between the asset being protected and a potential attacker, adding complexity and more points of failure to the equation is possible. Sadly, due to the cost of these types of solutions, the defensive proxy is often called upon to protect numerous assets like web servers and e-mail servers. As a result, a directed attack against the defensive proxy will result in a much larger loss than if the solution wasn't there at all!

"This is based primarily on an application, which, as a proxy, sits in front of a web farm. There are several products in this vein, but the concept boils down to a decoy system, where the protector seems as it is the original protectee to any attackers" [eEye03]. The core problem here is that after the decoy takes the attack and potentially

fails; your information is no longer available. Network topology decisions place these solutions at “choke points” on the network to flow all traffic through them. When that choke point fails, the network is basically cut off.

The advantages and disadvantages of this approach are summarized below:

Advantages:

- This approach protects the real asset against direct attacks by taking the fall for,
- In some ways this approach may lower the attack profile.

Disadvantages:

- This approach costs a lot to implement compared to security improvement it provide,
- You are opening yourself up to different types of attacks, other than the you are being protected,
- This approach lowers the network performance as it is placed at the “choke points”,
- If this solution fails, the availability of the system sustains an injury.

2.1.3 “You Should Know Better”

It revolves around the concept of forcing an application to think more about the type of information that it will consider legitimate. By residing within an application, like an ISAPI (Internet Server API) filter in Microsoft’s web server IIS, it wraps around the application and provides protection at the most crucial layer by monitoring all incoming and outgoing traffic as it passes through from the network to the kernel layer thus enabling the application (IIS) to change the way that it “thinks” about its inputted data and avoid being compromised [eEye03].

This approach is the process of “teaching” an application how to protect itself. This technique is the cornerstone of a true application firewall. This method, like none other found, is the most thorough in its testing and painless in terms of implementation.

All inputted information must be considered as hostile and not passed on to the host application until it has passed stringent checks for sanity and safety.

Products advance this approach looks for unique classes of attacks. They generally use multiple security filters to inspect Web server traffic for such issues as buffer overflows, parser evasions, directory traversal and other attacks. Therefore, they are able to block against attacks that have not yet been discovered. The true application firewall will actually protect from unknown attacks. This is due to the sophistication in searching for entire classes of attacks. It is the only way to proactively protect your web server from vulnerabilities [eEye03].

The advantages and disadvantages of this approach are summarized below:

Advantages:

- All application inputs are checked,
- This approach prevents exploitation of vulnerabilities because of programming errors,
- It can provide protection against unknown attacks,
- It can be configured differently for each web server.

Disadvantage:

- It is too expensive to use for all applications, so mostly used for web server protection,
- It is difficult to manage the configuration distributed on web servers in a site.

2.2 Phases of attacks

It is not an easy task to provide security and prevent attacks. In order to protect digital information and other network assets, thinking methodology and behavior of the attacker can help to find out a way to prevent them. A professional attacker will thoroughly investigate the target systems, and ensure that everything is safe for the attack without being detected. Then they attack in a very structured manner while

consistently monitoring the effect. This type of attacker threat tends to be far more thorough than the hobbyist or novice attacker who generally downloads ready-made exploitation scripts as they become available. Due to this thoroughness, attackers have a much higher success rate and do not usually get caught in the act.

To properly assess the security of a system, an understanding of the different phases of a successful attack or intrusion is necessary. By understanding the risk of exploitation, both can be applied to a structured list of possible controls to assess the current state of security, and the directions that need to be investigated.

All successful intrusions share the following characteristic phases [JK01]:

1. Reconnaissance
2. Assessment and Strategy
3. Exploitation / Invasion
4. Maintaining Access
5. Operations

Attackers place different priorities on each stage. In essence, the more time spent on one step ensures better results in the following steps. Also, each phase is conducted in such a way as to ease the way for the next step, and lower the chance of getting caught.

2.2.1 Reconnaissance, Assessment and Strategy

Attacks don't just happen. They are preceded by a phase of information collection. Potential attackers scan and probe the target network for potential vulnerabilities to determine which type of attack to attempt [ForeScout02].

Reconnaissance, or Recon, is the act of scoping out a target. This information gathering stage is the most important step an attacker takes, and all key information is considered. The Assessment and Strategy stage is the sorting of the gathered data to piece together an idea of what the hacker is attacking [JK01].

Recon can go undetected for considerable lengths of time and the Assessment and Strategy stage is often completely undetectable, as it is usually done without contact with the target.

These two stages are assessed together because Recon is the part of the act that involves interaction of some sort with the target, and the Assessment and Strategy stage is usually done remotely by reviewing the gathered data.

To launch successful attacks, attackers need information about the topology of the network, about accessible network services, about software versions, about valid user/password credentials, and about anything else that will help them succeed in their efforts.

Without such information, it is virtually impossible to successfully attack a network. Unlike attacks themselves, reconnaissance can only be performed in some very basic ways. These reconnaissance methods may change subtly over time, but they inevitably share some basic attributes. Typical recon techniques include [ForeScout02]:

- **TCP/UDP port scan:** This method accounts for at least 70% of all recon activity. Port scan consists of sending a message to each port, one at a time. The kind of response received indicates whether the port is used and can therefore be probed for weakness. This is extremely valuable information, since it reveals any applications running on the host that are accessible from the network. The attacker can gain the information about "listening" or "open" ports. This is a list of programs on the system that will respond to network requests as there are well-known port numbers common to all hosts. There is no way to stop someone from port scanning a computer while a host is on the Internet because accessing an Internet server opens a port, which opens a door to that host.

- **NetBIOS probes:** NetBIOS probes interrogate an IP host for computer names, user names, shared resources (such as shared folders or printers), and so forth. Responses to such probes will disclose the fact that the probed IP host actually runs a NetBIOS layer, and will reveal the objects sought by the attacker. NetBIOS probe can provide

valuable information about the host's status and help the attacker to find out weaknesses of the host.

- **SNMP probes:** These probes capitalize on the Simple Network Management Protocol (SNMP), which is used almost universally for communication between networked devices and management consoles. SNMP carries information about the nature, configuration, topology, and health of those devices. As a result, attackers can gain a plethora of valuable information about all types of network resources like router table.

Other recon methods include HTTP-based probes, "finger" probes, DNS zone transfers and SMTP-based interrogation. Altogether, there are about twenty basic recon categories – all of which are well understood [ForeScout02].

Typically, attackers use a variety of recon techniques. With each successive recon, the attacker gains more detail about the network's vulnerabilities: an unpatched service, a visible NetBIOS resource, an open FTP port. Even when recon doesn't yield any data, the attacker learns something about the network – i.e. that a host is not easily accessible. This helps the attacker further refine the attack strategy. Therefore, in the Assessment and Strategy stage these data are used to make the decision of where to attack.

2.2.2 Exploitation and Invasion

Once an attacker has gathered enough information and has pieced together a reasonable amount of information about the network or system they are attacking, and have devised an initial plan of attack, it is then possible to begin the Exploitation and Invasion stage. At this point, the attacker uses the gathered knowledge and attempts to access the server through the channels that were found open [JK01].

In this phase, the intruder has gained access to desired system facilities. Penetration and exploitation create a spiral of increasing intruder authority and a widening circle of compromise. For example, penetration at the user level is typically a means to find root-level vulnerabilities. User-level authorization is then employed to

exploit those vulnerabilities to achieve root-level privileges. Finally, compromise of the weakest host in a networked system allows that host to be used as a stepping-stone to compromise other more protected hosts. The success of this stage is mainly depends on the time spend on the recon stage and the experience of the attacker in well assigning the target host.

2.2.3 Maintaining Access

Once an attacker has penetrated the network (or if the attack is an inside job) steps are usually taken to make future accesses easier to conduct. This often includes installing a back-door program, but sometimes may be something as simple as setting up a home base under a seldom-used account name or identifying a misconfigured user account with suitable permissions to use to regain entry [JK01].

Attackers install tools and manipulate existing software on a system to maintain access to the machine on their own terms. They install backdoors, apply “rootkits” (the process of substituting binary executables with nasty variations), and sometimes even manipulate the underlying kernel itself to hide their evil deeds. Attackers also cover their tracks by hiding files, sniffers, network usage, and running processes. Finally, attackers often alter system logs, all in an attempt to make the compromised system appear normal.

2.2.4 Operations

This is the most dangerous part of a penetration; the attacker has all the access required to carry out his/her plan. If it is a spy operation, data could be sent to a remote collection repository [JK01]. While this process does not give a concrete damage, it can damage the trust on the organization, and this will be reflected to the finance of that organization. Sometimes, this financial effect could be so harmful that it could lead to the bankruptcy of that organization.

If the attack is a system-mapping reconnaissance mission, existing levels of access may be used to compromise more systems on the network [JK01]. This mapping process can be used to find out the weaknesses of the systems to gain access to them.

Then, those compromised systems, then, can be used to create a distributed denial of service (DDoS) attack. Or, that attack may be the first step for a spy operation and the compromised system could be used to gain access to more important systems on the network that could not be directly accessed.

Understanding the stages of attack process is central to effective defense. In fact, security administrators can take advantage of inherent flaws in the attack process to actually prevent attacks before they reach to its target. Just as attackers exploit vulnerabilities in the network to mount attacks, security administrators can exploit vulnerabilities in the attack process to protect themselves.

In each stage of an attack, there is the risk of exploitation. The types of exploits are [JK01]:

1. **Confidentiality–** implies that the information be agreed exclusively is same as the authorized person.
2. **Integrity**– consists of the need to maintain the stable information.
3. **Availability–** refers to the need to offer a service uninterruptedly, so that can be agreed in any moment and since any place, avoiding the possible thing that some type of incident stop it.
4. **Authenticity–** assures that the information is from the source that it claims to be from.

Security is defined through a “triad of concepts”. The stage of the attack plus the type of exploit identifies the risk. As an example, in reconnaissance an attacker is primarily collecting data. There is no intention to alter data integrity or availability, although confidentiality is affected. Therefore at the reconnaissance stage of the attack, there is a risk of loss of confidentiality. At the other end of the scale stands, the operations stage where the attacker performs his or her intent. If he is spying, confidentiality is at risk. If he is malicious and intends on causing damage to the company, integrity and availability are at risk. This may also be the case, if the attacker intends only to spy, but mistakes made along the way have affected integrity and availability.

Strictly speaking, a control is a mechanism to reduce risk. This may entail blocking data flow to outside networks, ensuring data integrity, or maintaining its accessibility. Controls also provide functions to notify when an attempt has been made to circumvent allowable access, and an audit trail to accurately document differences. Most controls are focused on a limited number of threats or vulnerabilities, and singularly can be defeated. Because of this, a robust suite of controls is necessary to mitigate risk [JK01].

There are five categories of controls:

1. **Deterrence–** discourage individuals from intentionally violating information security policies or procedures
2. **Prevention–** avoid the occurrence of unwanted events
3. **Detection–** identify unwanted events after they have occurred
4. **Correction–** remedy the circumstances that allowed the unauthorized activity or return conditions to what they were before the violation
5. **Recovery–** restore lost computing resources or capabilities and help the organization recover monetary losses caused by a security violation

Given the three types of exploits and the five characteristic phases of a successful intrusion; ways to mitigate possible risks can be developed around the five categories of control. Policies, procedures, assessments and assurance are the some of those ways. For each stage of an attack, at least one of the risks of exploitation are used, and often in several possible ways. In other words, for each stage of the attack, each of the five categories of controls are tested and weighed against their potential associated risks.

A gap in any one of the risk-control areas during any phase of an attack, as described above, is a potential security vulnerability. Vulnerability is a security hole in computer operating systems, system software, or application software. Attackers exploit vulnerabilities to gain control of, damage, or bring down a device on the network. Threats to computing systems are circumstances that have the potential to cause loss or harm, such as human attacks, natural disasters, inadvertent human errors, and internal

hardware and software flaws. “A control is a protective measure – an action, device, procedure, or technique – that reduces a vulnerability” [PPC97].

2.3 The State of Perimeter Security

In a physical structure, such as a building, strong materials for construction are used to provide the necessary security. By means of security, windows are located so that thieves can not access them easily; barriers are placed around the building and access is controlled on each entry. Besides these, systems of caution, alarms and cameras are placed to monitor the inside, in addition to properly equipped personnel, continuously, patrolling the installation.

Similar to physical security, information security managers have utilized multiple technologies to keep their networks safe. However, as an effect of the improvements in technology, networks are now connected to one or more outside networks – including, of course, the Internet. Hence, the corporations face with a wide range of threats. The fact that internal systems are actually quite vulnerable to all kinds of exploits makes these threats even worse. Plus, the widespread availability of reconnaissance tools has made it easier than ever for even novice attackers to bypass the enterprise security. So, security managers are under a lot of pressure to prevent any penetration to the network perimeter.

Luckily, similar to the physical security, there are numerous security tools to help security managers in setting up complex protection strategy plans for their computer systems. Mostly common ones are commented subsequently.

2.3.1 Firewalls

Firewalls are usually the first component of any perimeter defense. Firewalls provide a barrier of security among networks of different levels of confidence or security, utilizing network level access control politics. The major functional requirement of a firewall is to protect a private (internal) network from unauthorized external access.

Firewalls act like traffic cops and perform the critical task of filtering traffic crossing the network boundary. This filtering is done according to predefined security policies, which can be specified at the network layer and/or at the application layer. Firewalls utilize these static, manually configured, security policies to differentiate legitimate traffic from non-legitimate traffic.

Typical reasons for using a firewall to protect a private network include the following [SW00]:

- To prevent unauthorized external users from accessing computing resources on the internal network. This is necessary because it is extremely difficult and costly to attempt to secure all the hosts within a private network,
- To control internal user access to the external network to prevent the export of proprietary information,
- To avoid the negative public relations impact of a break in,
- To provide a dependable and reliable connection to the Internet, so that employees do not implement their own insecure private connections.

Firewalls must be installed at the choke points to control network traffic and implement network security policy of the organization for its external network connections, especially for the Internet. Because many Internet-based services are inherently insecure, a firewall must help an organization to disable some services and restrict others according to the organizational security policy [YY00]. Firewalls achieve this by examining the source and destination of all incoming and outgoing network traffic. All network traffic must pass through the firewall, which ensures that only permitted traffic are allowed through [CP00].

Set of rules specifies which packets can pass, which can not. For example, a request addressed to an email server is allowed through; a request addressed to the corporate accounting system is denied. Usually, traffic destined for a Web server (port 80) or an email server (port 25) is granted access. Unless you specify otherwise, a firewall typically blocks all traffic addressed to other locations (i.e., servers, databases,

or application servers) on the network, thus protecting those hosts against unauthorized external access.

There are various firewall products but they are grouped into three major types based on their mechanisms: packet filtering, stateful inspection, and proxying.

Packet filtering is a mechanism that control which packets can go to and come from a network by examining their headers. There are no content-based decisions. The decision is solely based on the packet headers which include source address, destination address, type of traffic (such as TCP, UDP, ICMP), and characteristics of the transport layer communications sessions (such as source and destination ports). Packet filters are associated with interfaces therefore; the interface that the packet comes from or will go through can be restricted. Packet filter firewall can have rules such as; letting some hosts send email via SMTP (Simple Mail Transfer Protocol) or not letting any outside host connect to an internal host using Telnet.

Packet filter firewalls provide transparent security as they work at lower layers. It does not require any user knowledge or any configuration on private network hosts. Since it is easy to implement, it is widely available in many routers. However, some protocols are not well suited to packet filtering and, some security policies can not be enforced by packet filtering. Moreover, packet filter firewalls make decisions for each network packet alone and does not examine the status of the connections that the packets belong to.

Another technology, stateful inspection, evolved from the need to accommodate certain features of the TCP/IP protocol suite. In essence, stateful inspection firewalls are packet filter firewalls with the connection status awareness capability. This awareness is done by making a dynamic list of active connections between hosts, called state table. A packet that does not belong to an active connection and is not a connection request is refused by the firewall. A packet that belongs to an active connection is allowed through, bypasses the firewall rules; therefore optimizing the inspection process.

Stateful inspection firewalls share the strengths and weaknesses of packet filter firewalls, but due to the state table implementation, stateful inspection firewalls are

generally considered to be more secure than packet filter firewalls. A stateful inspection firewall also differs from a packet filter firewall in that stateful inspection is useful or applicable only within TCP/IP network infrastructures. Stateful inspection firewalls can accommodate other network protocols in the same manner as packet filter firewalls, but the actual stateful inspection technology is relevant only to TCP/IP [NIST02].

Lastly, proxying is a mechanism that provides all internal hosts the external (untrusted) network access while appearing that a single host accessing the outside. Since all connection to the external network be done by a single host, deep packet inspection is possible before passing packets to internal hosts. Proxying examines source address, destination address, protocol used, source port, destination port and also payload (content) of packets.

Proxying allows writing complex set of rules that can not be done in packet filtering and stateful inspection. For example, “put” commands in FTP connections with a specific host can be rejected by a rule. Proxying can also provide many forms of user authentication and allow specifying different policies for different users. However, these benefits come with a lot of process cost and it requires a configuration at internal hosts. In other words, it does not provide transparent security. Therefore, it is mostly used for only HTTP protocol.

Firewalls, independent of the mechanism they use, may have an additional capability: Network Address Translation (NAT). NAT is the process of translating internal IP addresses to IP addresses that are visible to the external network. This prevents the disclosure of critical information about the structure of the internal network behind the firewall, thus provides an additional security.

NAT technology was developed in response to two major issues in network engineering and security. First, NAT is an effective tool for “hiding” the network-addressing schema present behind a firewall environment. In essence, NAT allows an organization to deploy an addressing schema of its choice behind a firewall, while still maintaining the ability to connect to external resources through the firewall. Second, the depletion of the IP address space has caused some organizations to use NAT for mapping non-routable IP addresses to a smaller set of legal addresses [NIST02].

Another valuable capability of firewalls is the construction of Virtual Private Networks (VPNs). Firewalls can also act as VPN gateways. A virtual private network is constructed on top of existing network media and protocols by using additional protocols, and usually encryption. If the VPN is encrypted, it can be used as an extension of the inner, protected network. Thus, an organization or agency can send unencrypted network traffic from systems behind the firewall to other remote systems behind a cooperating VPN gateway; the firewall encrypts the traffic and forwards it to the remote VPN gateway, which decrypts it and passes it on to the destination systems. Most of the popular firewalls nowadays incorporate this type of functionality [NIST02].

In most cases, VPN is used to provide secure network links across networks that are not trusted. For example, VPN technology is increasingly used in the area of providing remote user access to organizational networks via the Internet. This particular application has increased in popularity due to the expenses associated with implementing private remote access facilities, such as modem pools [NIST02]. By using VPN technology, an organization purchases a single connection to the Internet, and that connection is used to allow remote users access into private networks and resources. This single Internet connection can also be used to provide many other types of services. As a result, this mechanism is considered to be cost-effective.

These above capabilities can be used for protecting organization resources but there are also host-based firewalls to protect home users or as secondary defense for network users. Also, many people telecommute or work at home and operate on organization- or agency-proprietary data. Host-based firewalls provide network connection security with its packet filtering and VPN gateway capabilities. Host-based firewalls have been developed to perform many of the same functions as larger firewalls for organizational use. Host-based firewalls usually do not offer protection to other systems or resources. Likewise, host-based firewalls do not typically provide controls over network traffic that is traversing a computer system. They only protect the computer system they are installed on [NIST02].

Even with these capabilities, firewalls do not provide airtight perimeter protection. After all, they have to allow acceptable traffic through. The average firewall

is designed to deny clearly suspicious traffic – such as an attempt to Telnet to a device when corporate security policy forbids Telnet access completely – but is also designed to allow some traffic through – Web traffic to an internal Web server, for example.

Unfortunately, the main problem is different that is many exploits attempt to take advantage of weaknesses in various protocols that are allowed through perimeter firewalls, and once the Web server has been compromised, this can often be used as a springboard to launch additional attacks on other internal servers. Once a rootkit or back door has been installed on a server, the attacker has ensured that he will have unfettered access to that machine at any point in the future [NA03]. That is to say, firewalls can prevent most of the malicious access attempts but they can not stop attacks if they are not specified in their rule sets. The advantages and disadvantages of firewalls are summarized below:

Advantages:

- Firewalls can stop non-legitimate traffic at first point,
- Firewalls can filter protocols and services that are either not necessary or that cannot be adequately secured from exploitation [NIST02],
- A firewall can "hide" names of internal systems and internal network schema, thereby revealing less information to outside hosts [NIST02],
- Firewalls can concentrate extended logging of network traffic on one system.

Disadvantages:

- Firewalls utilize manually configured set of rules to differentiate legitimate traffic from non-legitimate traffic,
- Once a static policy is defined, the firewall can't react to a network attack – nor can it initiate effective counter-measures [NIST02],
- Firewalls only examine network packets that pass through them, do not examine network traffic between any two inside hosts,
- Most firewalls do not analyze the contents of the data packets that make up network traffic,

- Firewall policies can vary in effectiveness, depending on the expertise of the security manager and the complexity of the network environment.

2.3.2 Intrusion Detection System

A second layer in the perimeter defense is intrusion detection systems (IDSs). The audits of security existed before the intrusion detection. Audit is the process of generating, storing and revising events of a system chronologically. IDS is the evolved version of the traditional audits [GDG03].

The term audit, in Latin "audire" (to hear), is defined as "to examine the economic management of a company in order to verify if it is adjusted to the established rules by law or custom" [EC94]. Intrusion detection is the process of monitoring and searching networks of computers and systems for security policy violations [BR00]. Intrusion Detection Systems (IDSs) are software or hardware products that automate this monitoring and analysis process. An IDS inspects all inbound and outbound network activity, system logs and events, and identifies suspicious patterns or events that may indicate a network or system attack from someone attempting to break into or compromise a system [Jupitermedia02].

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

IDSs have gained acceptance as a necessary addition to every organization's security infrastructure. Since they are first put on the security market, those organizations have several compelling reasons to acquire and use IDSs. Some of them are listed below [BM01]:

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

There exist various IDS products in the market today. These products are categorized in several ways according to their different characteristics:

- **Misuse detection vs. anomaly detection:** In misuse detection, the IDS analyzes the information it gathers and compares it to large database of attack signatures which causes it being also called signature-based detection. It is easy to understand the concept as it uses simple comparisons. Essentially, the IDS looks for a specific attack that has already been documented. Like a virus detection system, misuse detection software is only as good as the database of attack signatures that it uses to compare packets against.

  In anomaly detection, the security manager defines the baseline, or normal, state of the network's traffic load, breakdown, protocol, and typical packet size. The anomaly detector monitors network segments to compare their state to the normal baseline and looks for anomalies. Therefore, anomaly detection is as good as its baseline definition.

- **Network-based vs. host-based systems:** In a network-based system, or NIDS, the individual packets flowing through the network are analyzed. Network-based IDSs often consist of a set of single-purpose sensors or hosts placed at various points in a

network. These units monitor network traffic, performing local analysis of that traffic and reporting attacks to a central management console. NIDSs analyze traffic moving across the network in much greater detail than a firewall. Therefore, NIDSs can detect malicious packets that are designed to be overlooked by a firewall's simplistic filtering rules. NIDSs also watch for attacks that originate from within a network. That is why, they are complement for firewalls.

In a host-based system or HIDS, activities on each individual computer or host are examined. An HIDS performs analysis of the local machine as it is running on. This commonly means that the HIDS software monitors log files, or other artifacts of incidents to detect that a security incident has occurred. HIDSs are not limited to log-file analyzers; they also include intra-kernel based mechanisms that detect ongoing security incidents. It is advisable to place HIDSs on all mission-critical systems, even those that should not, in theory, allow external access.

- **Passive system vs. reactive system**: In a passive system, the IDS detects a potential security breach, logs the information and signals an alert. Security manager has to examine the logs constantly and take the required measures.

  In a reactive system, the IDS responds to the suspicious activity by resetting the connection, by logging out the user or by reconfiguring the firewall to block network traffic from the suspected source.

These various types of IDS products have the same critical job of being accurate enough to differentiate between the good and bad traffic that gets into the network. The following are all possible results of intrusion detection:

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

Undetected bad traffic: Failure to identify malicious traffic as an attack.

This is the worst thing that can happen, because it means the IDS failed to do its job. Failing to detect an attack can occur when an IDS does not have adequate or comprehensive intrusion detection mechanisms in place. It also occurs when new attacks are created and then missed by poorly implemented detection mechanisms [NetScreen02]. While it is virtually impossible to detect every attack, the goal of any system should be to minimize the number of undetected attacks.

Detected bad traffic: Identifying "real attacks" as an attack.

This is the ideal result of an IDS. The ability to detect bad traffic with speed and reliability is referred to as intrusion detection accuracy. All other functions of the system hinge on this capability [NetScreen02]. The more accurate the system, the more you can trust its abilities. A system must have proven accuracy before enabling it to take the necessary actions (such as dropping the connection) to secure the network.

Identifying good traffic as an attack: False alarm or false positive.

This is the most troublesome and time-consuming aspect of IDS solutions. It occurs when the IDS sees something in legitimate and benign traffic that makes it believe there is an attack [NetScreen02]. It is detrimental because each and every alarm needs to be investigated in order to determine whether an attack was successful and assess any resulting damage. Every moment spent investigating a false positive reduces the time available to investigate real threats. The result is that false positives can erode trust in the product; sometimes causing real attack alarms to be overlooked (the "crying wolf" effect).

Most IDSs can be tuned to try to reduce the occurrence of false positives, however, the tuning process is often long and involved, sometimes taking weeks to accomplish [NetScreen02]. In addition, because of the management design of current IDSs, tuning is often an all or nothing approach. This means that security managers must choose whether or not to look for a certain attack. If, in the interest of reducing false positives, the detection of certain attacks is turned completely "off," those attacks will be able to go by the IDS completely undetected.

Nevertheless, false positives are not the result of poor software design by IDS vendors. As Stefan Axelsson demonstrated in his 1999 ACM presentation, [AS99] there are some fundamental mathematical constraints that make false positives endemic to the whole paradigm of real-time signature (pattern) recognition. Deviations from baseline norms can be caused by a variety of factors, many of them innocuous. So, false positives are inherently part of signature-based intrusion detection schemes or any other type of anomaly detection system.

Identifying good traffic as good traffic:

An ideal result of intrusion detection mechanisms, identifying good traffic for what it is – good traffic [NetScreen02].

Therefore, an ideal IDS should identify as many attacks as possible and limit the number of false positives. Unfortunately, there is no single detection mechanism available today that an IDS can deploy to detect every type of network-based attack. There exist several detection mechanisms today. Each approach has distinct advantages and disadvantages. These detection mechanisms include:

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

Intrusion detection using signature:

The evolution of NIDSs started with the implementation of a non-intrusive packet monitor, called a sniffer because of its ability to "sniff" the packets on the network. Intrusion detection vendors applied the packet-monitoring concept to build systems that performed packet signature detection [NetScreen02]. Signature-based detectors analyze system activity, looking for events or sets of events that match a predefined pattern of events that describe a known attack. They compare events and packets with signatures stored in their database and find out the matching ones. The most common form of signature-based detection used in commercial products specifies each pattern of events corresponding to an attack as a separate signature [BM01].

Intrusion detection that is based on recognizing and matching attack signatures is very straightforward. Basically, it entails looking for a particular pattern in traffic that has been characterized as a known exploit or vulnerability. Thus, signature detection finds attacks for which a signature is written and it is very effective at detecting attacks without generating an overwhelming number of false positives.

However, there are many drawbacks to signature-based approach to intrusion detection – especially if effort is placed entirely on building up a large repository of attack signatures, without regard to how the traffic is reassembled, decoded, normalized and analyzed. It is a problem when information is transmitted over the network, the information is split into numbered TCP (Transmission Control Protocol) segments that are sent as packets. In an ideal world, the packets would be transmitted in sequence and without loss. But, unfortunately, that's not the case. When a message is actually transmitted, the network will deliver the packets randomly (out of sequence) or as even smaller pieces of data (called fragments), which are broken down by networking devices, such as routers, to facilitate ease of transmission. Even worse, for whatever reason, packets can get "lost" or can be duplicated [NetScreen02].

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

Intrusion Detection Using Protocol Anomalies:

Protocol anomaly detection, which is sometimes called protocol analysis, is the ability to analyze packet flows (the uni-directional communication between two systems) to identify irregularities in the generally accepted Internet rules of communication [NetScreen02]. Those rules are defined by open-protocols and published standards (RFC-Request For Comment), as well as vendor-defined specifications for communication between networked devices.

Protocol anomaly detection attempts to save time by first identifying the protocol, and then looking specifically for anomalous activity or attack patterns relevant to that protocol. By doing so, it can do a much more targeted, and thus more effective search [TopLayer02]. In other words, protocol anomaly detection identifies traffic that doesn't meet specifications or violates the relevant standards. Once an irregularity is identified, it can be used to make network security decisions. This is very effective in detecting suspicious activity, such as a buffer-overflow attack.

The advantages of protocol anomaly detection are that it can detect [NetScreen02]:

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

An example of detecting the FTP bounce attack using protocol anomaly is described below:

The FTP bounce attack exploits a design flaw in the specifications of FTP (File Transfer Protocol). To download or upload files, a user (FTP client) must first connect to an FTP server. When this happens, the server requires the client to send the IP address and port number to which the file should be sent to or taken from. This is done via a mechanism called a "Port Command." However, the "Port" command specification does not limit the IP address to the user's address. Because of this, an attacker can tell the FTP server to open a connection to an IP address that is different from the user's address and then use the open port to transfer files containing a Trojan through the FTP server onto the victim [NetScreen02]. Since protocol anomaly detection is designed to look at network relationships and determine whether both sides are acting within the normal specifications, IDS can parse the requests in a "Port" command whenever seen and compare it to the IP address from which the "Port" command arrived. If they do not match, the IDS needs to send an alarm.

Protocol anomaly detection-based systems do a good job of detecting some of the unknown attacks like the one described. But their main drawback is that they are unable to identify attacks that operate without violating any protocols, such as Trojan or Worm. These attacks install and open up a backdoor on a network resource. This backdoor lays inactive until the attacker activates it and takes control over the resource [NetScreen02]. Besides that, protocol anomaly detection-based systems usually produce a large number of false alarms due to the unpredictable behaviors of users and networks.

Intrusion detection using stateful signatures:

Another alternative approach that overcomes the accuracy deficiencies of packet signature detection is stateful signature detection. This advanced detection mechanism identifies attack patterns by utilizing both stateful inspection and protocol analysis, which is performed as part of protocol anomaly detection [NetScreen02]. As a result, stateful signature detection systems understand the context of each data byte and the state of the client and server at the time of transmission. This means that stateful signatures can be compared to only relevant data bytes, according to the communication state to which each signature is relevant.

Stateful signature detection mechanism looks at the context and the placement of signature to make smarter decisions about whether it represents an attack. The IDS keeps track of the state of the connection with the outside entity, and considers the broader context of all the transactions initiated during the connection [TopLayer02]. In other words, stateful signatures only look for an attack in the state of the communication where that attack can cause damage, thus significantly improving performance and reducing false positives.

The drawback of this system is the same as the signature-based detectors; it catches only known perpetrators – and only if the signature database is constantly updated. Unfortunately, the people out there trying to exploit networks are neither lazy nor stupid; they constantly unleash new variations and new attacks. With each new attack, new signatures have to be “taught” to the IDS. Over time, this has led to a need for IDS products to hold literally thousands of attack signatures, and constantly scan for them all [TopLayer02]. In addition, while the signature database being constantly

updated, attackers know exactly how the IDS products work, so they continuously make slight alterations to elude detection. Thus, a more intelligent signature mechanism is needed to bring about more accurate results.

Once IDSs have obtained event information and analyzed it to find symptoms of attacks, they generate responses. Some of these responses involve reporting results and findings to a pre-specified location. Others involve more active automated responses [BM01]. Though researchers are tempted to underrate the importance of good response functions in IDSs, they are actually very important. Today, IDS products support a wide range of response options, often categorized as active responses, passive responses, or some mixture of the two.

Active Responses:

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

- **Collect additional information:** The most innocuous, but at times most productive, active response is to collect additional information about a suspected attack. This might involve increasing the level of sensitivity of information sources (for instance, turning up the number of events logged by an operating system audit trail, or increasing the sensitivity of a network monitor to capture all packets, not just those targeting a particular port or target system). Collecting additional information is helpful for several reasons. The additional information collected can help resolve the detection of the attack (assisting the system in diagnosing whether an attack did or did not take place). This option also allows the organization to gather information that can be used to support investigation and apprehension of the attacker, and to support criminal and civil legal remedies [BM01].

- **Change the environment:** Another active response is to halt an attack in progress and then block subsequent access by the attacker. Typically, IDSs do not have the ability to block a specific person's access, but instead block IP addresses from which the attacker appears to be coming. It is very difficult to block a determined

and knowledgeable attacker, but IDSs can often deter expert attackers or stop novice attackers by taking the following actions [BM01]:

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

- **Take action against the intruder:** The most aggressive form of this response involves launching attacks against or attempting to actively gain information about the attacker's host or site. Due to legal ambiguities about civil liability, this option can represent a greater risk than the attack it is intended to block. The primary reason for approaching this option with a great deal of caution is that it may be illegal. Furthermore, since many attackers use false network addresses when attacking systems, this action has a high risk of causing damage to innocent Internet sites and users. Finally, strike back can escalate the attack, provoking an attacker who originally intended only to browse a site to take more aggressive action [BM01].

Passive Responses:

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

- **Alarms and notifications:** Alarms and notifications are generated by IDSs to inform users when attacks are detected. The most common form of alarm is an onscreen alert or popup window. This is displayed on the IDS console or on other systems as specified by the user during the configuration of the IDS. The information provided in the alarm message varies widely, ranging from a

notification that an intrusion has taken place to extremely detailed messages outlining the IP addresses of the source and target of the attack, the specific attack tool used to gain access, and the outcome of the attack [BM01]. Another set of options is to configure the IDSs so that they send alert messages to cellular phones and pagers carried by incident response teams or security managers.

- **SNMP traps and plug-ins:** Some commercial IDSs use SNMP traps and messages to send alarms to central network management consoles. This provides the ability to adapt the entire network infrastructure to respond to a detected attack, the ability to shift the processing load, associated with an active response, to a system other than the one being targeted by the attack, and the ability to use common communication channels [BM01].

Users should be aware that most of the existing IDSs are not difficult to by-pass if the attacker is knowledgeable. In addition, users should be aware that IDSs generate voluminous logs that must be examined carefully if the IDS is to be effective [NIST02]. The advantages and disadvantages of IDSs are summarized below:

Advantages:

- The deployment of network-based IDSs has little performance impact upon an existing network,
- Host-based IDSs are able to monitor events local to a host,
- IDSs can allow security managers, regardless of their level of security expertise, to track security problems on their systems, initiating incident handling procedures [BM01].

Disadvantages:

- IDSs have a tendency to generate “false positives”. That is, they frequently generate alerts about an attack when none is taking place,
- The only way to eliminate false positives would be to tune the system down to the point where it would also ignore real attacks – yielding “false negatives” – an obviously unacceptable approach,

- IDSs are extremely administration-intensive. Highly skilled security professionals must constantly tune the system, update signatures, analyze alerts to determine if they are real or false, and then respond with appropriate remedial action,
- An IDS must be closely monitored and continually fine-tuned to the usage patterns and vulnerabilities discovered in its deployed environment. Such maintenance typically consumes a fair amount of administrative resources and effort,
- A substantial amount of time may pass between the attack and the remediation, allowing the attacker to do irreversible damage in the meantime,
- Any IDS system that relies exclusively on documented attack profiles will be vulnerable to new, as-yet-undocumented attacks.

2.3.3 Honeypots

A third security technology being used by many organizations is honeypots. A honeypot is a system or dataset for which there is no legitimate reason for someone to interact with it and therefore all use can be considered unauthorized [Honeypots03]. Honeypots are installed behind a firewall, although is also possible to situate them in front of them. Honeypots are designed to [BM01]:

- divert an attacker from accessing critical systems,
- collect information about the attacker's activity, and
- encourage the attacker to stay on the system long enough for security managers to respond.

Honeypots are highly flexible security tools with different applications for security. Unlike firewalls or IDSs, honeypots do not solve a specific problem. Instead, they have multiple uses, such as prevention, detection, or information gathering. There are various implementations but they all share the same concept: a security resource that should not have any production or authorized activity. Theoretically, a honeypot should see no traffic because it has no legitimate activity. This means any interaction with a honeypot is most likely unauthorized or malicious activity [SL03]. Any connection attempts to a honeypot are most likely a probe, attack, or compromise. This is what a

honeypot is, it is a security resource whose value lies in being probed, attacked, or compromised. Honeypots in a network should not affect critical network services and applications. Those series of characteristics distinguish honeypots clearly of other solutions of security.

Honeypots only capture bad activity; any interaction with a honeypot is most likely unauthorized or malicious activity. Thus, honeypots collect small data sets having high value, as the data set contains only the attacks. This means it's much easier (and cheaper) to analyze the data a honeypot collects and derives value from it [SL03].

Implementing a honeypot is very easy. The following steps specify an example implementation of a honeypot [TechTarget03]:

- Install the operating system without patches installed and using typical defaults and options,
- Make sure that there is no data on the system that cannot safely be destroyed,
- Add the application that is designed to record the activities of the invader.

Another example of a honeypot is a system used to simulate one or more network services on a host. An attacker assumes the host is running vulnerable services that can be used to break into it. This kind of honeypot can be used to log access attempts to those ports including the attacker's keystrokes [Honeypots03]. This could give security managers advanced warning of a more concerted attack. That is what honeypots used for.

Honeypots can be divided into two types: production and research. Production honeypots are easy to use, capture only limited information, and are used primarily by companies or corporations, help preventing, detecting, or responding to an attack Research honeypots are complex to deploy and maintain, capture extensive information, and are used primarily by research, military, or government organizations [Honeypots03]. They are used to collect information. That information can be used in early warning and prediction, law enforcement or understanding trends in attacker activity.

Some of the advantages and disadvantages associates to them are described subsequently:

Advantages:

- Honeypots lure attackers by presenting a more visible and apparently vulnerable resource than the enterprise network itself. Thus, they distract attackers from more valuable hosts on the network [ForeScout02],
- Honeypots are useful for detecting attacks, since they provide a single point for security managers to monitor for evidence of anomalous activity,
- Honeypots are useful for forensics, since they can be specifically designed to retain data pertaining to an attack [ForeScout02],
- Honeypots allow in-depth examination of attacks during and after exploitation of the targets,
- Unlike most security technologies (such as IDS systems) honeypots work fine in encrypted or IPv6 environments [SL03],
- Honeypots can provide early warning about new attack and exploitation trends.

Disadvantages:

- Honeypots can only track and capture activity that directly interacts with them [SL03],
- Honeypots are not especially effective at attack prevention,
- Honeypots have the risk of being taken over by the attackers and being used to harm other systems [SL03],
- Maintaining a honeypot is said to require a considerable amount of attention [TechTarget03],
- A honeypot may offer as its highest value nothing more than a learning experience (that is, you may not catch any attackers) [TechTarget03],
- An expert attacker, once diverted into a decoy system, may become angry and launch a more hostile attack against an organization's systems [BM01].

Besides, it should not be forgotten that attackers don't have to focus on a limited number of targets with today's automated tools. They can attack the honeypot and the enterprise network – and anything else in sight. In fact, if they are incorrectly configured, honeypots can actually make the enterprise more vulnerable to attack by virtue of being linked logically to it [ForeScout02]. Therefore, a high level of expertise is needed for administrators and security managers in order to use these systems.

2.3.4 Network-based Antivirus Systems

Forth security technology being used by many organizations is network-based antivirus systems. Network-based antivirus systems are solutions that are installed on a gateway between two networks to prevent the spreading of viruses across the network. The limitations of host-based antivirus softwares demonstrate the need for properly implemented network-based antivirus systems that allow security managers to deploy comprehensive antivirus protection faster, and guard against the rise of threats that endanger networks as they spread by exploiting known vulnerabilities.

After a virus is released and begins to spread and infect users, several steps should be followed to take necessary measures against it [Fortinet02]:

1. A new virus threat is recognized,
2. Antivirus companies gather suspected infected files and search for the virus code,
3. The virus is identified and a signature is developed that will uniquely identify it, without causing "false positives",
4. The new signature is added to the antivirus vendor's signature database,
5. Systems are "inoculated" against the new virus by propagating the new signature database to every device that runs the scanning engine.

Organizations are most vulnerable to new infections during the period between detection and inoculation, and any delays in the process increase the "window of vulnerability" [Fortinet02]. For large organizations especially, the biggest portion of the vulnerability window is the time required to update every PC, laptop, and server in their network with a new signature database. Reducing the window of vulnerability

maximizes the performance of antivirus systems. Security managers can achieve this by deploying antivirus protection at the network edge, using network-based antivirus as opposed to relying solely on antivirus protection deployed on each computer and server in the network.

Network-based antivirus system is a complement to antivirus protection performed on email servers and individual desktop computers [AM03]. Network-based antivirus systems are installed in the DMZ (DeMilitarized Zone) in order to capture and compare incoming and outgoing packet contents to a database of known virus signatures. Unfortunately, network-based antivirus systems have to operate under much more difficult constraints than host-based antivirus systems have to. Files are transported over networks in the payload portions of packets, each containing a small chunk of the file. A typical packet payload on the Internet is approximately 1,500 bytes in length. However, many viruses are substantially longer than 1,500 bytes, and can exceed 100K bytes in length. As a result, it is not sufficient for network-based antivirus systems to simply scan each packet individually. If a virus is longer than 1,500 bytes, and the signature for the virus relies on patterns that occur in portions of the packet that are separated by more than 1,500 bytes, then a packet-by-packet scan will never detect it [Fortinet02].

To deal with this challenge, some network-based antivirus systems contain hard disks, and function essentially as host-based antivirus systems that are deployed at the edge of a network. Packets are reassembled into files on the disk, and streamed off the disk at a rate that will not overwhelm the software scanning engine. Thus, this requires a huge buffer space to reassemble those packet contents. Besides, to perform antivirus scanning at network speeds requires 100-1000 times more processing power than is required for other security functions such as VPN and firewall processing. As a result, a major challenge to network-based antivirus solutions is the need to provide effective antivirus protection without reducing network performance [Fortinet02]. Therefore, they are used mostly for email traffic. They are mostly not used for real time Web traffic as they reduce the network performance. But, there are also products that can scan Web traffic in real time at a price.

Another challenge of antivirus science is to ensure that all infected files are stopped (100% detection rate) without creating "false positives"; that is, without mistakenly marking a clean file as being infected. Therefore, several methods are used to detect viruses:

The most common approach to virus detection is the "signature-based" approach. Signatures are telltale patterns of bytes that are unique to a particular virus. Signature-based antivirus products are composed of two key elements: a database that contains the signatures for known viruses, and a scanning engine that compares files under investigation with the signatures in the database to detect a match indicating the presence of a virus [Fortinet02]. The simplest signatures are streams of bytes that are known to occur in a particular sequence within the code of a virus. Signatures can be made more complex by incorporating wildcard characters to account for known virus variations. However, virus codes can be encrypted, which randomizes the code and makes it much harder to develop a signature. Moreover, there are polymorphic viruses, which actually modify themselves slightly at each replication, further complicating, and in some cases defeating the ability of antivirus vendors to develop signatures.

Another approach to virus detection is developed to overcome the problem of large variety of viruses: "heuristic scanning". Heuristic scanning looks for patterns of known bad behavior, rather than looking for a specific virus signature. For example, some viruses read and write certain files or execute certain operations in a way that would never be found in legitimate programs. The sequences of operations that constitute these behaviors can also be used to develop so-called heuristic signatures, which enable antivirus engines to detect some viruses without an explicit signature [Fortinet02]. While appealing in concept, such systems are not commercially viable. A key problem is that the definition of appropriate and inappropriate behavior changes fairly rapidly in today's modern computing environment. The rules that define acceptable behavior change with new releases of operating systems and application programs. Like virus signatures, the rules that govern "anomaly-based" virus prevention must be frequently updated to avoid an unacceptable number of false positive detections. Even so-called signature-less systems are not so different in practice from their signature-based counterparts.

Therefore, despite their known weaknesses and limitations, signature-based antivirus systems are still by far the most effective and widely used method of virus detection. But there is still the truth of ineffective against new attacks until the signature database being updated. The advantages and disadvantages of network-based antivirus systems are summarized below:

Advantages:

- Network-based antivirus systems provide a single barrier behind which all hosts are protected [Fortinet02],
- A single update of the signature database or scanning algorithms on the network-based antivirus gateway protects all of the systems on either side from viruses flowing in either direction [Fortinet02],
- Network-based antivirus systems greatly reduce the risk that an unprotected host will be compromised, and mitigate the risk of memory-resident and other viruses that are a challenge for host-based antivirus systems [Fortinet02],
- Network-based antivirus systems reduce the load on email servers by eliminating infected emails before they reach the servers,
- Network-based antivirus systems are well positioned in the network to scan Web and other traffic that tends to bypass conventional host-based antivirus systems.

Disadvantages:

- For a typical file, many millions of comparisons may be required to determine if it is free from infection [Fortinet02],
- Larger signature databases and longer, more complex signatures require more time for an antivirus system to scan and reduce the performance,
- Network-based antivirus systems has to scan packets after reassembling them in buffers that can cause the drop of packets when the buffer is full,
- Network-based antivirus systems typically employ dedicated platforms that have their own vulnerabilities.

CHAPTER 3

INTRUSION PREVENTION SYSTEMS

Despite the very efforts of security managers, internal corporate computing resources are being penetrated every day, with attacks spreading at the speed of communications to other resources on the network. Thus, while firewalls, IDSs, honeypots and network-based antivirus systems have their place in the arsenal of corporate defense, they are neither pro-active enough nor labor-efficient enough to meet the needs of today's IT-dependent, resource-constrained organization. Security managers simply can't afford to rely exclusively on current security technologies, such as firewalls, intrusion detection systems, honeypots, network-based antivirus systems, to protect their network perimeters.

Most of the organizations are using firewalls but it is apparent that firewalls are not always effective against many intrusion attempts. The average firewall is designed to deny clearly suspicious traffic – such as Telnet access to a device when corporate security policy forbids it completely – but is also designed to allow some traffic through – Web traffic to an internal Web server, for example.

The problem is, that many exploits attempt to take advantage of weaknesses in the various protocols that are allowed through the perimeter firewalls, and once the Web server, email server or any other host has been compromised, this can often be used as a springboard to launch additional attacks on other internal servers and hosts. Once a "rootkit" or "backdoor" has been installed on a server or host, the attacker has ensured that he/she will have unfettered access to that machine at any time in the future.

Firewalls are also typically employed only at the network perimeter. However, many attacks, intentional or otherwise, are launched from within an organization. VPNs, laptops, and wireless networks all provide access to the internal network that often bypasses the firewall. IDSs may be effective at detecting suspicious activity, but do not provide protection against attacks. Recent worms such as Slammer and Blaster have

such fast propagation speeds that by the time an alert is generated, the damage is done and spreading fast. Network-based antivirus systems, also, are not fast enough to recognize those worms since their signature databases are not updated.

Therefore, security managers need a solution that's more effective at preventing – rather than reacting to – all types of attacks while putting less strain on the time and energy of overworked security technicians. Intrusion prevention security architectures serve as the next generation of network security software that proactively strengthens networks and desktop computers against damage from the cyber-attacks.

3.1 Definition

While the phrase "intrusion prevention system" (IPS) has entered the security lexicon, it's still too early to say exactly what an intrusion prevention system is because companies use the term a half-dozen different ways. Some use the term to describe next-generation IDS systems that can block certain kinds of attacks. Others use the term more broadly and include firewalls, for instance, in the intrusion prevention category, since firewalls can block certain attacks [RM03].

Definition of IPS varies widely because of the marketing purposes of security tool vendors. Every vendor has its own definition and so it seems that they are selling the right product; the most reliable and the most precious. Since technological developments are produced by vendors and many vendors exist in the market today, it is hard them to compromise on a definition. This confusion causes the new technology, called revolution by the vendors, be perceived as a minor improvement.

Before going any further, a definition of IPS is required in order to clearly describe what we will examine. A definition used by a group of vendors that develop network-based IDSs is:

> "… As the name implies, intrusion prevention systems (IPSs) do not simply detect attacks as do IDSs; they actually prevent attacks from taking place or automatically block them upon detection. They enable an organization to take proactive, highly automated steps to guard against intrusions…" [PD03]

> "… In fact, most IPSs have IDS at their core. The key difference between the technologies is implied by their names: IDS products only detect malicious traffic, while IPS products prevent such traffic from entering your network..." [PD03]

Some other vendors group network IPSs into two: in-line, out-band. They define the out-band IPS as IDS that can manage the firewall or a routers and instruct it to stop the suspicious activity which is the source of the confusion between IDS and IPS. These vendors also define in-line IPS as follows:

> "… In-line intrusion prevention systems are unique in that they sit on the network, where they supplement existing firewall and antivirus solutions. An IPS monitors traffic and actively intervenes by dropping packets deemed malicious, scrutinizing suspicious sessions or taking other actions in immediate real-time response to an attack…" [RM03]

The question is not how it is different from any other system on the network. The question is what it is used for. For instance, the following definition has been offered up recently:

> "… The definition of IPS is any device (hardware or software) that has the ability to detect attacks, both known and unknown, and prevent the attack from being successful..." [DN03]

The failure or success of an attack depends on its goal. As stated before, an unsuccessful attack can be successful for the attacker because he/she gains information about the system. A much better definition is:

> "… Intrusion prevention, at its most basic, requires some way of recognizing intrusions in real time, of being able to handle both known and unknown types of attacks, and then having a way of blocking the incursions…" [RB03]

We will use a mixture of the last two definitions:

"An intrusion prevention system is a hardware or software solution that has the ability to detect both known and unknown attacks in real time and, proactively prevent them before they cause any kind of harm."

3.2 Deployment

An intrusion prevention system attempts to be proactive, and is designed to stop intrusions, preventing suspected system calls and events, blocking the offending traffic before it does any damage rather than simply raising an alert as, or after, the malicious payload has been delivered. It achieves this by sitting directly in-line with the system calls and network traffic [NA03].

By sitting in-line, IPS, ideally, inspecting all system calls and packets going inbound or outbound. It performs a range of detection analyses. If the IPS deems the system calls unsuspicious or the packet harmless, it forwards it. End users are unaware of any effect. However, when the IPS detects suspicious system calls or packets, it can then initiate one of many response mechanisms that security manager has configured. It may restrict the system call or packet, by forwarding it normally up to a certain limit. Or, the IPS can discard it completely. Of course, an IPS must also have an extensive reporting mechanism – but this must be more than a simple log of activity. The IPS can create an alarm and transmit it to appropriate destinations [TopLayer02].

As with IDS systems, IPS products fall into two categories: Network-based IPS (NIPS) and host-based IPS (HIPS) [NA03].

3.2.1 Network-based IPS:

Network-based IPS (sometimes known as an In-line IDS or Gateway IDS (GIDS)) is a device put on the network in a critical data path that inspects all the traffic allowed through by the firewall [RB03].

It could be thought of a something of a hybrid system, combining features of a standard IDS, a firewall and, sometimes, a network-based antivirus system. Those prevention products use various methods to spot trouble, such as looking for the

characteristic signatures of known viruses or comparing the current traffic to a baseline of normal traffic behavior. If the devices detect anomalies, they block the traffic from continuing onto the network. Thus, they provide truly effective protection for computing resources on a large scale.

Two general types of network-based IPS deployments are available: out-of-band IPS and in-line IPS [PD03].

Out-of-band IPS (OOB IPS) systems straddle the firewall much like an IDS. Based on the IDS detection, an OOB IPS can manage the firewall, instructing it to terminate the suspicious activity. It is the actually an IDS with an add-in response mechanism that can reconfigure firewalls or routers on the network to block or limit specific traffic. They are the primitive IPSs. Figure 3.1 shows the installation of an out-of-band IPS. Since they only need a communication interface between firewall, or router, and IDS, they are easy to implement. However, the main problem in those systems is the time between detection and reconfiguration, that is long enough for an attack to make harm.

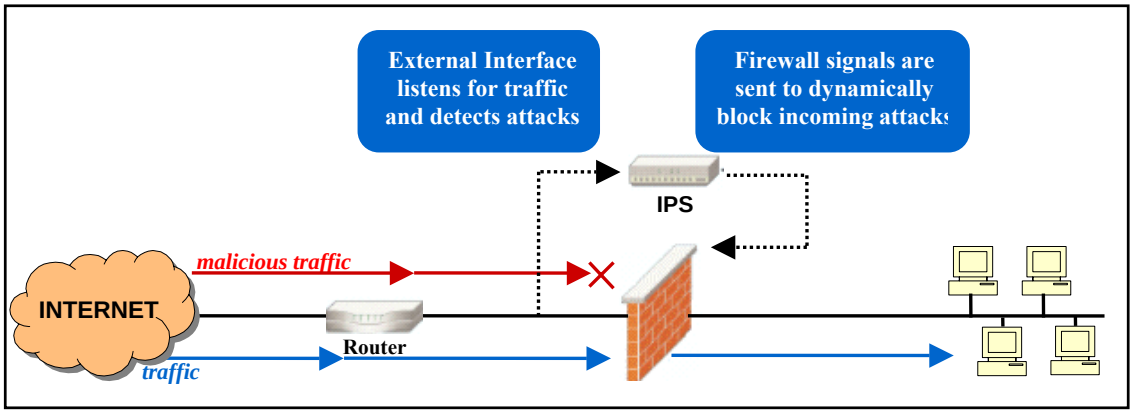

Figure 3.1: Out-of-band IPS installation [PD03]

In-line IPS products perform similarly. The key difference is that in-line IPSs have traffic-blocking functionality built in. They are the next generation. They can response to an attack much faster than an out-of-band IPS. In addition to protecting the network perimeter, in-line IPSs are well suited to guard against threats that originate behind the firewall. Figure 3.2 shows the installation of an inline IPS. There are already available hardware and software appliances of these systems.

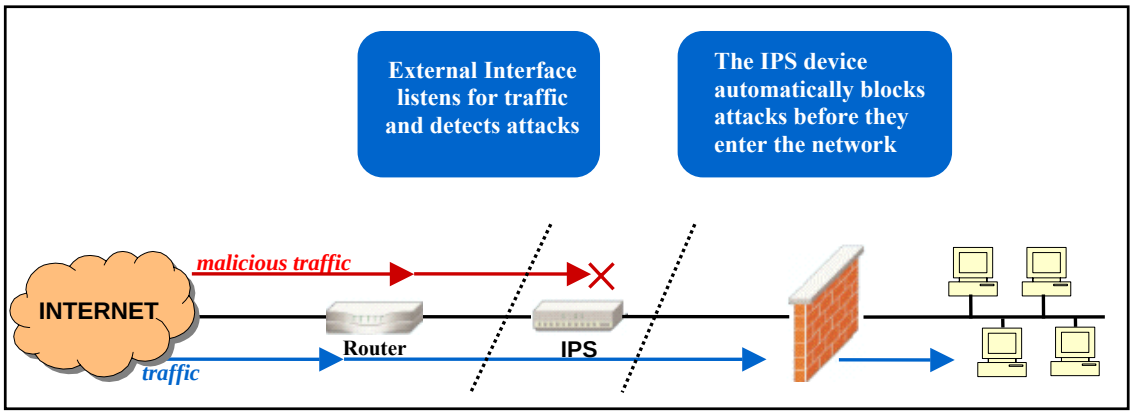

Figure 3.2: In-line IPS installation [PD03]

The advantages and disadvantages of network-based IPS is summarized below:

Advantages:

- Network-based IPS are proactive and they block attacks before they enter into the network,
- Network-based IPSs can stop suspicious traffic at first point and reduce the load on firewalls,
- Network-based IPSs is cost effective, as one can potentially protect dozens to hundreds of targets.

Disadvantages:

- An IPS is as good as its detection mechanism,
- Detailed analysis for detection can ruin the network performance,
- Network-based IPSs may not interpret traffic as target would,
- False positives can make more harm than any attack as the availability sustains an injury.

3.2.2 Host-based IPS:

To provide truly effective protection for computing resources on a large scale, security managers cannot rely only on the detection capability of a network-based IPS. It is an essential rule of security: do not rely upon any single solution or process for

protection (“defense in depth”). Thus, an additional layer of security is required, that is host-based IPS.

Host-based IPSs provide protection at the end point of attacks and allows much more targeted detection and prevention of intrusions. Host-based IPSs flag intrusions by comparing the behavior of systems against expected norms. If deviations occur, the systems then have some way of blocking the procedures that are causing the anomalous behavior without affecting the machine's normal operations [RB03].

The main approach in host-based IPSs is to define appropriate behaviors and then enforce those behaviors on every end-user desktop and network server across an enterprise. It’s because solutions that are implemented by replacing shared libraries or analyzing system audit logs can be bypassed relatively easily [DT02b]. The problem is to define good or expected behavior. For example; it is an appropriate behavior that an email management software just displays the emails that the users selects, but it is not appropriate, the software immediately attempts to send email to every contact listed in that user’s address book after it is displayed. Likewise, if a process originating from a web browser, mail software or Microsoft Office program group attempts to read, write or modify files in its program folder or temporary folder is appropriate but, it is not appropriate if it writes to Windows system files. It is by looking at system and application behavior in this way and defining which actions are legitimate and which actions are suspect that pioneering intrusion prevention technologies can preemptively neutralize an errant system action when it attempts to do something that is outside the realm of expected behavior [DT02a].

Host-based IPSs rely on agents installed directly on the host system being protected, and which interacts closely with the underlying operating system and resident services in order to detect and prevent rogue system calls. It binds closely with the operating system kernel and services, monitoring and intercepting system calls to the kernel or APIs in order to prevent attacks as well as log them. To ensure the highest levels of security and minimize the ability to bypass the security policy on a host, application calls must be intercepted at the kernel level where the determination is made

of their adherence to policy [DT02b]. Figure 3.3 shows the host-based IPS interaction with the operating system kernel and services.

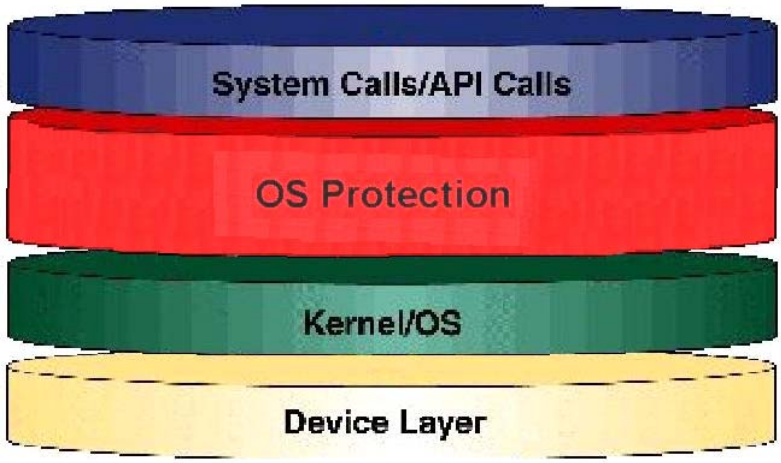

Figure 3.3: Host-based IPS (OS Protection)

A host-based IPS may also monitor data streams and the environment specific to a particular application (for instance, file locations and Registry settings for a Web server) in order to protect that application from generic attacks for which no "signature" yet exists.

Since a host-based IPS agent intercepts all requests to the system it protects, it has certain prerequisites - it must be very reliable, must not negatively impact performance, and must not block legitimate traffic. The advantages and disadvantages of host-based IPSs are summarized below:

Advantages:

- Host-based IPSs offer greater ability to understand processes on hosts,
- A single event can effectively replace interpretation of hundreds of network packets,

Disadvantages:

- Difficult to manage more than a few systems,
- Because of the tight integration with the host operating system, future operating system upgrades could cause problems.

3.3 Types of Existing IPS Products

Companies that develop network security appliances, today, selling various products as intrusion prevention system (IPS). Currently available products are providing parts of intrusion prevention tasks. These products use different ways to stop attacks. They can be divided into five different categories that focus on attack prevention at layers. These five types of IPSs are:

- inline network intrusion detection system (NIDS),
- application-based firewalls/IDS,
- layer seven switches,
- network-based application IDSs (hybrid switches), and
- deceptive applications.

3.3.1 Inline Network Intrusion Detection Systems

Inline network intrusion detection system (NIDS), mostly contain built in IDS technology that monitors and analysis network traffic. As such inline NIDSs can readily detect attacks embedded in legitimate traffic and takes the next step; terminating suspicious activity, which is core the difference between preventing and detecting systems. Since inline NIDSs have traffic blocking functionality built in, possible intrusions can not only be detected but also can be prevented.

The inline NIDS could be thought of a something of a hybrid system, combining features of a standard IDS and a firewall. Like a firewall, it supports at least two network interfaces one designated as external and one as internal. Some appliances may have more than two in order to monitor multiple network paths, but the basic requirement is for two interfaces for data and one for management [NA03].

Most NIDS would be configured with two network interface cards (NICs), one for management and one for detection (Figure 3.4). The NIC that is configured for detection usually does not have an IP address assigned to it, making it a “stealth”

interface. Since it does not have an IP address assigned to it no one can send packets to it or cause the NIDS to reply using that interface [DN03].

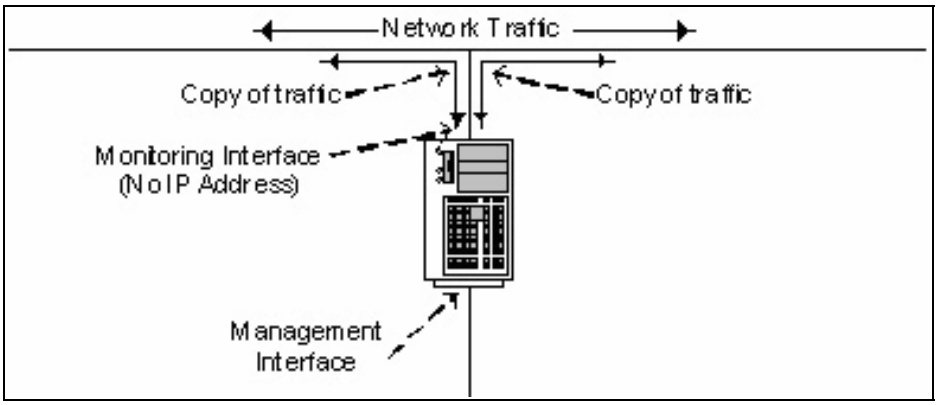

Figure 3.4: Connections of a typical Network Intrusion Detection System [DN03]

Inline NIDSs are placed in the line of packet transfer checks all packets passing through it for intrusions. Inline NIDS examines all the fields in individual data packets that enter a network unlike firewalls that are limited to checking only IP addresses and ports. Inline NIDS examines payloads – SMTP commands, HTTP URLs and headers – within a packet. This enables immediate detection of maliciously formed packets. NIDS is deployed on a network segment to compare captured network data with a file of known malicious signatures, which are compositions of patterns of attacker techniques [II02].

Some NIDSs uses anomaly detection in order to detect intrusion attempts. In the case of a protocol anomaly inline NIDS, it will be able to stop unknown attacks based on the protocols that it is able to decode, as well as the knowledge of those protocols. Another type of anomaly detection is done by inspecting statistical information to catch extreme instances. This is by identifying statistical anomalies with reference to a baseline of statistics on activities like; traffic flow pattern, user logins, file activity, disk activity, and so on. When there is an abnormal or unusually high or low value from the baseline, the IDS is able to detect it [II02].

The inline NIDS works like a layer two bridge, sitting between the systems that need to be protected and the rest of the network (Figure 3.5).

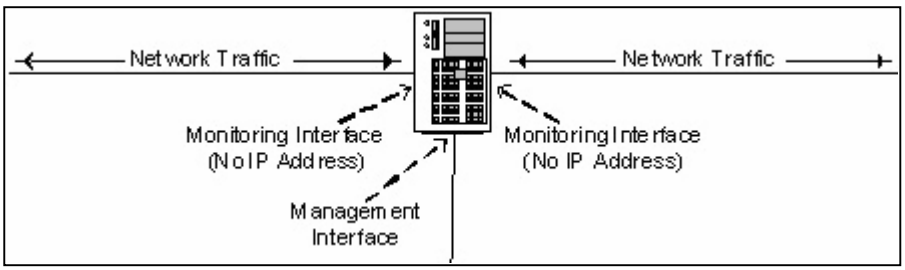

Figure 3.5: Installation of an inline NIDS [DN03]

All traffic will pass through the inline NIDS. Unlike a regular bridging device though, the inline NIDS will inspect the packet for any vulnerabilities that it is configured to look for. If a packet contains a piece of information that trips a signature the packet can be forwarded or dropped and either logged or unlogged (Figure 3.6).

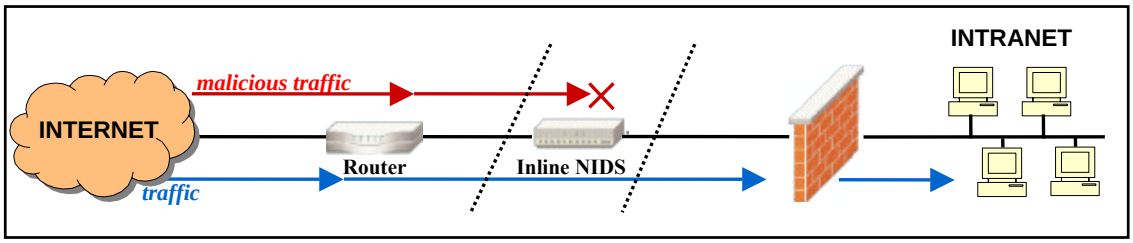

Figure 3.6: Location of an inline NIDS

Some can take it a bit further though: they have the added ability to rewrite the offending packet(s) to something that won't do any harm, a procedure known as packet scrubbing (Figure 3.7). This type of inline NIDS is useful if you don't want the attacker to know that their attacks are unsuccessful or if you want the attacker to continue to attack one of your systems in an attempt to gather more evidence. It is also useful when deploying a honeynet [Honeypots03] or honeypots so that only the outbound traffic, from the honeynet, is "scrubbed" ("rewritten") [DN03].

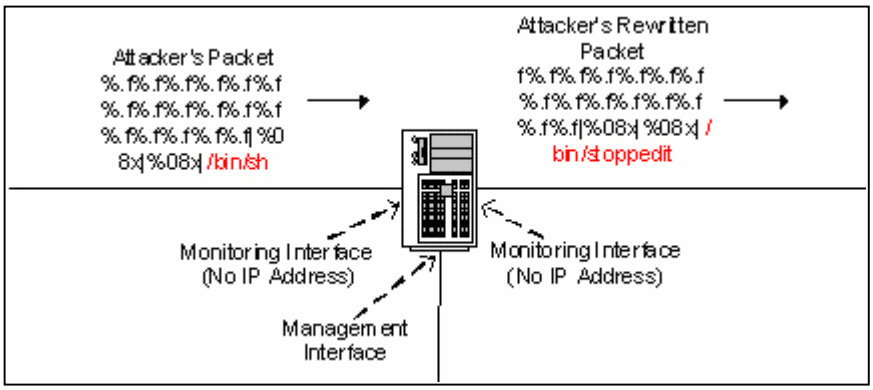

Figure 3.7: Packet scrubbing [DN03]

These inline NIDSs will feel most comfortable in the hands of security teams that already deal with NIDS. Because these inline IDSs are variants of existing NIDS, writing rules for them is very easy and offers a way to catch new attacks. To block unknown attacks with a signature-based inline NIDS, you would have to have some generic rules, like looking for NOOP sleds. This does not, however, stop all new attacks [DN03]. The advantages and disadvantages of inline NIDSs are summarized below:

Advantages:

- An inline NIDS offers the detection capabilities of a regular NIDS with the blocking capabilities of a firewall,
- Inline NIDSs allow security managers to monitor and protect many servers or networks with a single device,
- Inline IDSs provide a generic level of protection, but they still have a great place in protecting systems that are hard to protect (i.e. AS400, Tandem and mainframes) [DN03].

Disadvantages:

- If the system fails or crashes the traffic would not get through the device,
- The success or failure of an inline NIDS depends on its detection mechanism,
- Detection process causes a latency which could ruin network performance, and lost of packets and connections,
- Inline NIDSs are able to protect certain applications that are in wide use (such as, IIS, Apache, etc.) [DN03],
- Inline NIDSs offer no protection for bad programming or misconfigurations.

3.3.2 Layer Seven Switches

A layer seven switch is a network device that integrates routing and switching by forwarding traffic at layer 2 speed using layer 7 (application layer) information [DN03]. The device performing the redirection looks at each request string and determines where

the request should be redirected. Figure 3.8 shows layer seven switch as a load balancer. If the packet is redirected to a server, it is delivered just as if there were one server on the network. The server then processes the packet and takes appropriate action [EC99]. For example, an XML (EXtensible Markup Language) switch can analyze the XML tags at the application level and make forwarding decisions [DN03].

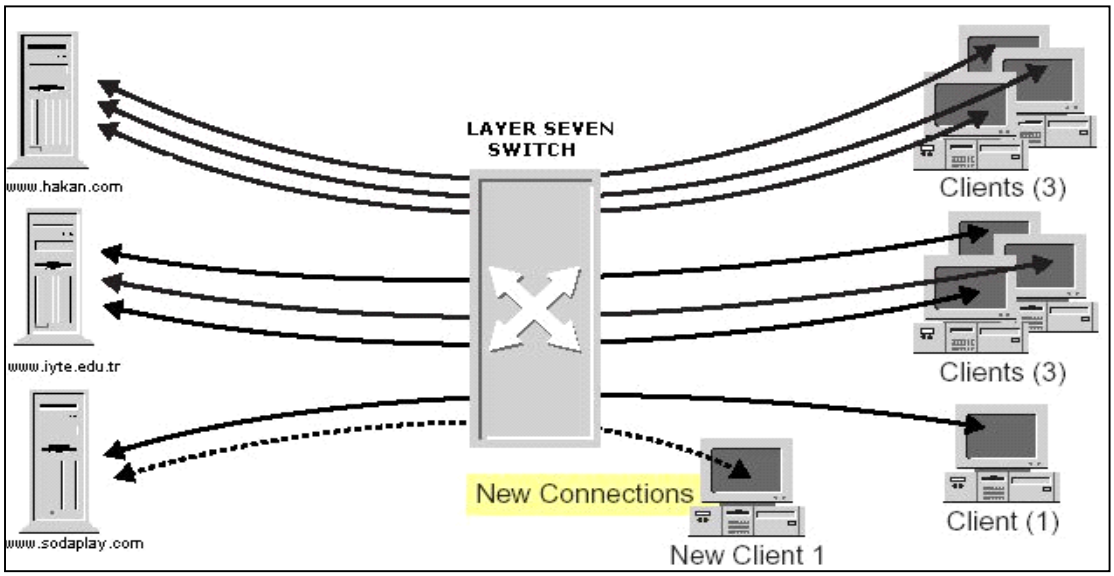

Figure 3.8: Layer seven switch as load balancer

Generally, there's a router that connects a local network to a remote network. This same router can be configured to handle transparent redirects, and is sufficiently powerful and cost-effective enough for most installations. But the drawback, in this case, is the router can't detect a failed server and configure around it, nor can it distribute workloads across multiple servers [EC99].

For some installations require a more robust redirect option than a router offers, such as redundant servers, automatic failover, and other similar features, layer seven switches support more robust methods [EC99]. With the high demands on networks and servers to deliver bandwidth intensive content, network engineers mostly use these switches to load-balance an application across multiple servers. To accomplish this, layer seven switches can inspect application layer information (i.e. HTTP, DNS, and SMTP) to make switching or routing decisions [DN03].

Most layer seven switches perform load balancing in several ways, including least number of connections, round robin, least busy (as determined by the switch), or hash algorithms [EC99]. In the case of a Web application, they can inspect the URL to direct particular request to specific servers based on predefined rules [DN03].

Layer seven switches are built on custom hardware to deliver high performance, even in the most demanding networks. These systems can easily handle gigabit and multi-gigabit traffic. They work similarly to a signature-based inline NIDS when it comes to stopping attacks. That means, the drawbacks are similar to the inline NIDS. They can only stop attacks that they know about (Figure 3.9), but they offer a way to write signatures just like a NIDS. The one attack that they can stop that most others can't are the denial of service (DoS) attacks. These devices have the horsepower to mitigate DoS attacks without affecting the rest of the network performance. Placing these devices in front of firewalls would give protection for the entire network. They offer security as a consequence of what they do in regards to inspecting application layer content for routing/switching decisions. The companies that make these devices have now started to add security features to their products, like DoS and DDoS (distributed DoS) protection [DN03].

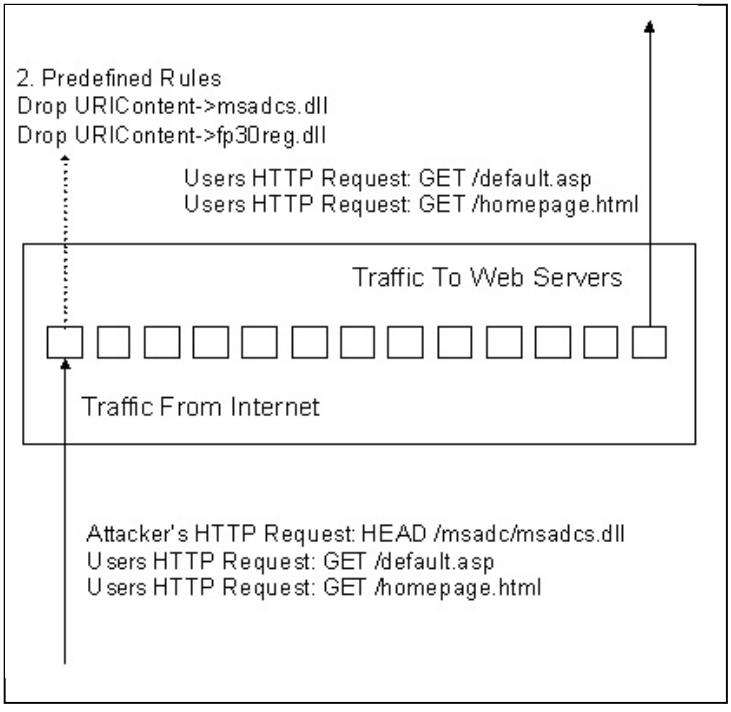

Figure 3.9: Layer seven switches stop attacks based on predefined rules [DN03]

The advantages and disadvantages of layer seven switches are summarized below:

Advantages:

- Layer seven switches are configurable for redundancy: they can be configured in a hot standby mode or in a load-balancing mode,
- A lot of the security features are available as a software upgrade, so it may be possible to use an all ready existing switch.

Disadvantages:

- Layer seven switches are mostly limited to web server protection,
- They do not provide real protection against new fast spreading attacks.

3.3.3 Application Firewall/IDS

Application firewall/IDS is a new paradigm for securing specific services. Application firewalls/IDSs are usually marketed as an intrusion prevention solution rather than a traditional IDS solution [DN03]. For this reason, it is also called application IPS. It is not meant to be a replacement for firewalls and/or IDS, but instead will be a complementary technology [RP01]. Application IDSs overlap onto the domain of traditional firewalls and IDS systems, but offer a different type of protection that neither of them or both can offer.

Application IDSs are loaded on each server that is to be protected. They are customizable to each application that they are to protect. They don't look at packet level information; rather, they look at API (Application Programming Interface) calls, memory management (i.e. buffer overflow attempts), how the application interacts with the operating system, and how the user is suppose to interact with the application (Figure 3.10) [DN03].

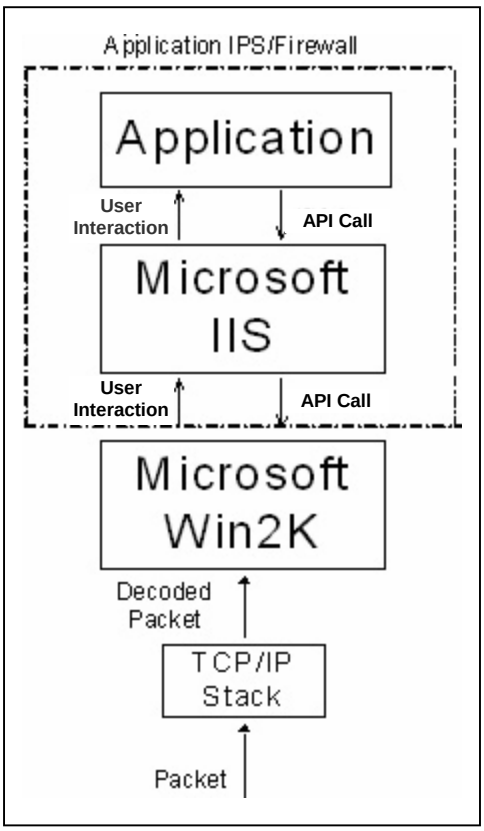

Figure 3.10: Application IPS/Firewall [DN03]

At its basest form, an application IDS is a reduced application that allows filtering of input for a specific service to allow only desired input. By defining what is acceptable and what is not, it can abort abnormal sessions of a protocol and stop them from continuing on to the actual application [RP01].

Application IDSs can profile a system before protecting it. During the profiling phase, application IDS can watch the user's interaction with the application and the applications interaction with the operating system to determine what legitimate interaction looks like. After creating a profile, or policy, of the application, it can be set to enforce that policy [DN03].

An application IDS can be finely tuned to the expected functionality of the host device. For example, an application shield on an email server would likely be configured to prohibit an incoming mail message from automatically launching any executables, because that is not a typical or necessary email function [AM03].

Unlike the inline NIDSs or the layer seven switches, the application IDSs are "fail close" type of systems, which means that if some action is attempted that is not predefined then the application IDS will stop the action from taking place [DN03]

A generic application IDS should take a "less is better" stance. It should be in place to limit the possible inputs to its service. An application IDS will reassemble protocol state information beyond a normal firewall, and can block general classes of attacks (such as buffer overflow attacks and format string attacks) before they are handed off to the actual application for processing. Also, in addition to blocking attacks, application IDSs can also be used to reduce the amount of possible information that an attacker can glean from the system it protects. This means that it should be able to stop or change banner information; often it should allow everything as if normal and just discard it before passing it on to the actual application [RP01].

If implemented correctly, this technique can stop not only specific vulnerabilities, but also general classes of vulnerabilities. The application IDS can protect against new vulnerabilities before they are found and exploited by means of its structure. This helps protect against poor programming and unknown attacks [RP01].

By profiling the application prior to enforcing the policy you can get very granular with the policies that are made. This type of IPS offers one of the greatest amounts of protection for custom written applications. Since each application IDS is loaded on each physical server you can customize each policy so that it can offer the greatest amount of protection. While the overhead in management of this many application IDSs could be daunting, it does pay off [DN03]. The advantages and disadvantages of application IDSs are summarized below:

Advantages:

- Application IDSs can monitor the each request, which often allows them to trace unauthorized activity,
- Since application IDS can profile a system, it requires less or, sometimes, no configuration,
- Application IDSs protect against attacks based on poor programming errors.

Disadvantages:

- When an application is profiled, the user needs to make sure that every aspect of the application is used so that the application IDS can see the interaction and write a rule for it. If thorough testing of the application is not carried out, then some parts of the application may not work [DN03],
- When the application is updated it might have to be profiled again to ensure that the policy does not block legitimate use [DN03]. Since applications are patched and updated frequently, this profiling and later testing process can be a significant overhead.
- It is important to note that application IDSs must take special care to make certain that they do not implement any types of new security bugs into the system [RP01].

3.3.4 Hybrid Switches

This type of technology is a cross between the application firewall/IDS and the layer seven switch. These systems are hardware based in front of the servers, like the layer seven switch, but instead of using a regular NIDS type of rule set, hybrid switches use a policy similar to the application firewall/IDS (Figure 3.11). They inspect specific traffic for malicious content defined by the policy that is configured [DN03].

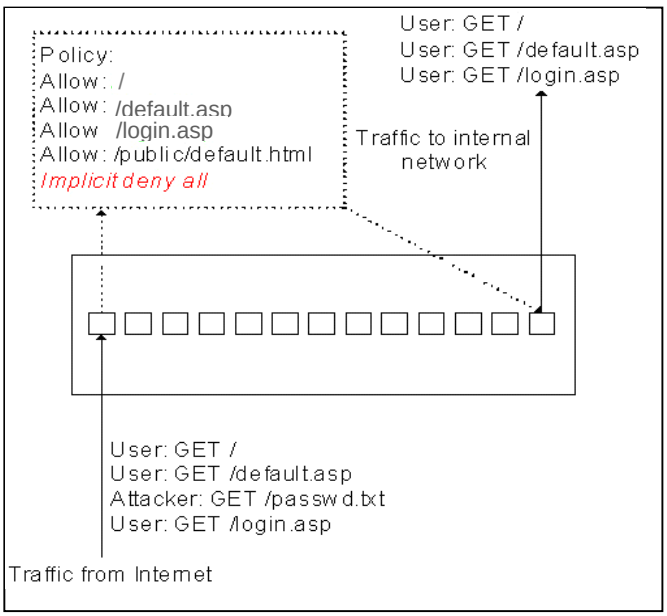

Figure 3.11: Policy based filtering of requests [DN03]

Some of these hybrid switches come with application layer vulnerability assessment product that compliment the solution. An application can be scanned with their vulnerability assessment product and the information from that scan can be imported into the hybrid switch as a policy. This saves the security manager a lot of time configuring the policy to defend the application [DN03].

The hybrid switch works in a similar manner to the layer seven switch, but instead of only having a handful of signatures that can block attacks aimed at the web server, it can have detailed knowledge of the web server and the application that sits on top of the web server. It also fails close if the user's request does not match any of the permitted requests. If the application that is being protected receives a lot of traffic, the hybrid switch can be combined with a layer seven switch to offer even higher performance. The layer seven switch can be configured to send certain types of requests to the hybrid switch for further inspection, decreasing the amount of requests that the hybrid switch has to look at and increasing performance [DN03].

The advantages and disadvantages of hybrid switches are summarized below:

Advantages:

- Since hybrid switches can be configured by vulnerability assessment applications, they require less or, sometimes, no configuration [DN03],
- Security managers can easily change the rule set of the hybrid switch.

Disadvantages:

- Hybrid switches are mostly limited to web server and web application protection [DN03],
- Application updates require reconfiguration of policy that does not block legitimate use. Frequent patching and updating can lead to reconfiguration overhead.

3.3.5 Deceptive Applications

The methodology is not new; it was first discussed in 1998 at a RAID conference. This type of technology uses some deceptive practices. First, it watches all the network traffic and figures out what is good traffic (Figure 3.12), similar to the

profiling phase of the application firewall/IDS. Then, when it sees attempts to connect to services that do not exist or at least exist on that server, it will send back a response to the attacker (Figure 3.13) [DN03].

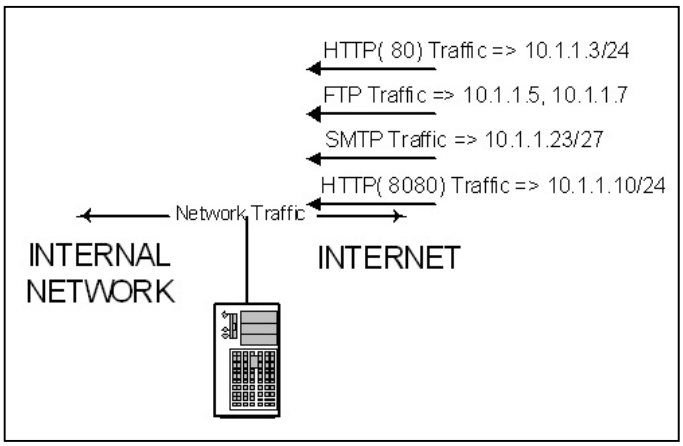

Figure 3.12: Determination of legitimate traffic [DN03]

The response will be "marked" with some bogus data so that when the attacker comes back and tries to exploit the server the deceptive application will see the "marked" data and stop all traffic coming from the attacker. The attacker does not have to try to attack the fake web server to be detected. Based on the configuration of the product, there can be "marked" data within the packet data. This would catch an attacker even if he/she was to attack a legitimate web server [DN03].

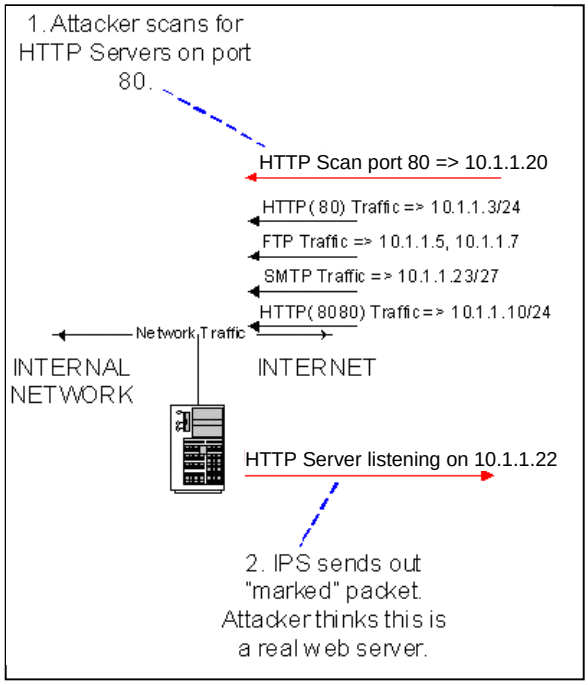

Figure 3.13: Deceptive application returns "marked" response to the attacker [DN03]

This technology misleads an attacker to a service that does not exist. This is useful preventing amateur attackers that use automated tools, sometimes the professionals. However, it does not prevent attacks that are using the knowledge that not only gained from the packets but others sources. This technology only prevents attacks that has a leading reconnaissance phase based on the data gain from responses to the send packets.

The advantages and disadvantages of hybrid switches are summarized below:

Advantages:

- Deceptive applications do not require signature updates. They first profile the network and later, they can update their profile automatically [DN03],
- Deceptive applications can prevent attacks that are followed by reconnaissance activities.

Disadvantages:

- Deceptive applications only detect attacks that are followed by a reconnaissance activity, therefore they can not be successful for preventing professional attackers,
- Trojans or backdoors can easily be overlooked during the profiling phase. Thus, deceptive applications can not protect against this kind of attacks [DN03].

Each type of existing IPS products offers a different level of protection, and each type has its own advantages and disadvantages. By looking at the way that each IPS works, security managers should figure out which solution would be best fitted for their needs. As is the case with most security technologies, there is no “one size fits all” solution. Security managers might even use a combination of these solutions. For instance, they might use a layer seven switch in front of Internet firewall to defend against DoS attacks and known attacks, using application layer firewalls/IDS software or hybrid switch to protect Web servers, an inline NIDS to protect legacy systems, such as AS400 or Tandems, and deceptive applications to mislead the attacks. Using a combination of these solutions would provide much tighter security.

3.4 Network-based IPS

Doing an examination over the existing IPS products, a network-based IPS solution that would a unified, proactive security solution by combining the capabilities of existing security technologies such as firewalls, intrusion detection systems, honeypots, network-based antivirus systems will be defined. Before developing such a network-based IPS architecture, challenges of such architecture and the proposed solution to those challenges would be examined.

3.4.1 Implementation Challenges

There are a number of challenges to implementing a network-based IPS device that do not have to be faced when deploying IDSs or honeypots. These challenges all originate from the fact that the NIPS is designed to work in-line, presenting a potential choke point and single point of failure.

If a passive IDS fails, the worst that can happen is that some attempted attacks may go undetected. If an in-line device fails, it can seriously impact the performance of the network. Perhaps latency rises to unacceptable values, or perhaps the device fails closed, which would lead to a DoS condition. There will be no attacks getting through, but that is of little consolation if none of the customers can reach the e-commerce site behind the NIPS [NA03].

Even if the NIPS device does not fail altogether, it still has the potential to act as a bottleneck, increasing latency and reducing throughput as it struggles to keep up with up to a Gigabit or more of network traffic. NIPS devices that use off-the-shelf hardware struggle to keep up with a heavily loaded Gigabit network, especially if there is a large set of signatures loaded, and this could be a major concern for both the network administrator as well as the security administrator.

Dropped packets are also an issue, since if even one of those dropped packets is used in the exploit data stream it is possible that the entire exploit could be missed. Most high-end NIPS vendors get around this problem by using custom hardware,

populated with advanced FPGAs (Field Programmable Gate Array) and ASICs (Application-Specific Integrated Circuit) [NA03].

Another potential problem is the false positive. The false positive alert arises when an exploit signature is not defined carefully enough, such that legitimate traffic can cause it to fire accidentally. While consuming time and effort of the security managers in a NIDS device, the results can be far more serious in an in-line NIPS appliance. Since the NIPS device first drops the – supposed to be – "offending" packet, and then blocks the entire data flow from the suspected attacker, these would lead to a DoS condition [NA03]. If the traffic that triggered the false positive was part of a customer order, that customer will not wait around for long as his/her entire session is torn down and all subsequent attempts to reconnect to the e-commerce site are blocked by the NIPS.

Beside performance and detection capabilities, the problem with any NIPS product, in a gigabit network, is the amount of alert data it is likely to generate. "Even with relatively low alert rates of ten per second, you are talking about 36,000 alerts every hour. That is 864,000 alerts each and every day" [NA03]. Detection accuracy is essential in order to keep the number of alerts to an absolute minimum. Once the alerts have been raised, it then becomes essential to be able to process them effectively. Advanced alert handling and forensic analysis capabilities – including detailed exploit information and the ability to examine packet contents and data streams – can amplify or reduce the trust on the NIPS appliance [NA03].

3.4.2 NIPS Detection Methods

The success or failure of a NIPS depends mainly on its detection mechanism. NIPSs use the detection mechanisms used in NIDSs. Thus, NIPSs also have a tendency to give false alarms. Since the NIPSs work inline and can block or limit the traffic, those false alarms will directly affect the network performance and, in some cases, the availability of some services or the network. Thus, accuracy of detection is the highest priority challenge in NIPS implementation. There four main approaches used by NIPS vendors to minimize false alarms in their products today:

3.4.2.1. Multi Method Detection:

Multi Method Detection (MMD) is the ability to detect intrusions using multiple mechanisms at the same time to improve detection accuracy. First generation IDSs were built using only a single intrusion detection method, typically either Signature Detection (pattern matching) or Protocol Anomaly Detection. Because some attacks can only be identified using Signatures and others using Protocol Anomaly Detection, products designed to use only a single method provide incomplete detection coverage. A recent trend in the market is the use of multiple detection mechanisms. Most of the IDS and IPS vendors use multiple methods to detect known and unknown attacks much more accurately than they do with a single detection method [NetScreen02]. For instance, Symantec's ManHunt IDS initially relied on protocol anomaly detection. Subsequent versions allow users to import Snort signatures to strengthen anomaly detection. Cisco also upgraded its IDS software, adding protocol and traffic anomaly capabilities to its signature-based detection system. NetScreen's appliances have a suite of eight detection methods, including stateful signatures, protocol and traffic anomaly, and backdoor detection [MCA03].

No single detection method can detect all attacks. In fact, each different type of attack requires a different detection mechanism to identify it. A system that implements only one detection mechanism compromises the network security by leaving the network vulnerable to the attacks that it cannot detect.

In MMD, various detection mechanisms (see Table 3.1) work together to deliver accurate and efficient attack identification. Use of multiple methods allows IPSs and IDSs to maximize the attacks detected, while reducing the false alarms.

Various detection methods are organized so that "good traffic" will pass through as fast as possible, known "bad traffic" will be blocked as soon as it is detected, and "suspicious traffic" will be analyzed until it is specified as a good or bad traffic. Various detection mechanisms are, mainly, placed one after the other. Each vendor chooses from the same detection mechanisms set, but organizing different number of mechanisms in different orders. Figure 3.14 shows the Top Layer Networks' implementation of a MMD engine.

Table 3.1 : Detection Methods [MCA03].

Method	Mechanism	Advantages	Disadvantages
Pattern matching	Scans incoming packets for specific byte sequences (the signatures) stored in a database of known attacks	– Identifies known attacks – Provides specific information for analysis and response	– May trigger false positives – Requires frequent updates of signature tables – Attacks can be modified to avoid detection
Stateful matching	Scans for attack signatures in the context of a traffic stream rather than individual packets	– Identifies known attacks – Detects signatures spread across multiple packets – Provides specific information for analysis and response	– May trigger false positives – Requires frequent updates of signature tables – Attacks can be modified to avoid detection
Protocol anomaly	Looks for deviations from standards set forth in RFCs	– Can identify attacks without a signature – Reduces false positives with well-understood protocols	– May lead to false positives and false negatives with poorly understood or complex protocols – Protocol analysis modules take longer to deploy to customers than signatures
Traffic anomaly	Watches for unusual traffic activities, such as a flood of UDP packets or a new service appearing on the network	– Can identify unknown attacks and DoS floods	– Can be difficult to tune properly – Must have a clear understanding of "normal" traffic environment
Statistical anomaly	Develops baselines of normal traffic activity and throughput, and alerts on deviations from those baselines.	– Can identify unknown attacks and DoS floods	– Can be difficult to tune properly – Must have a clear understanding of "normal" traffic environment

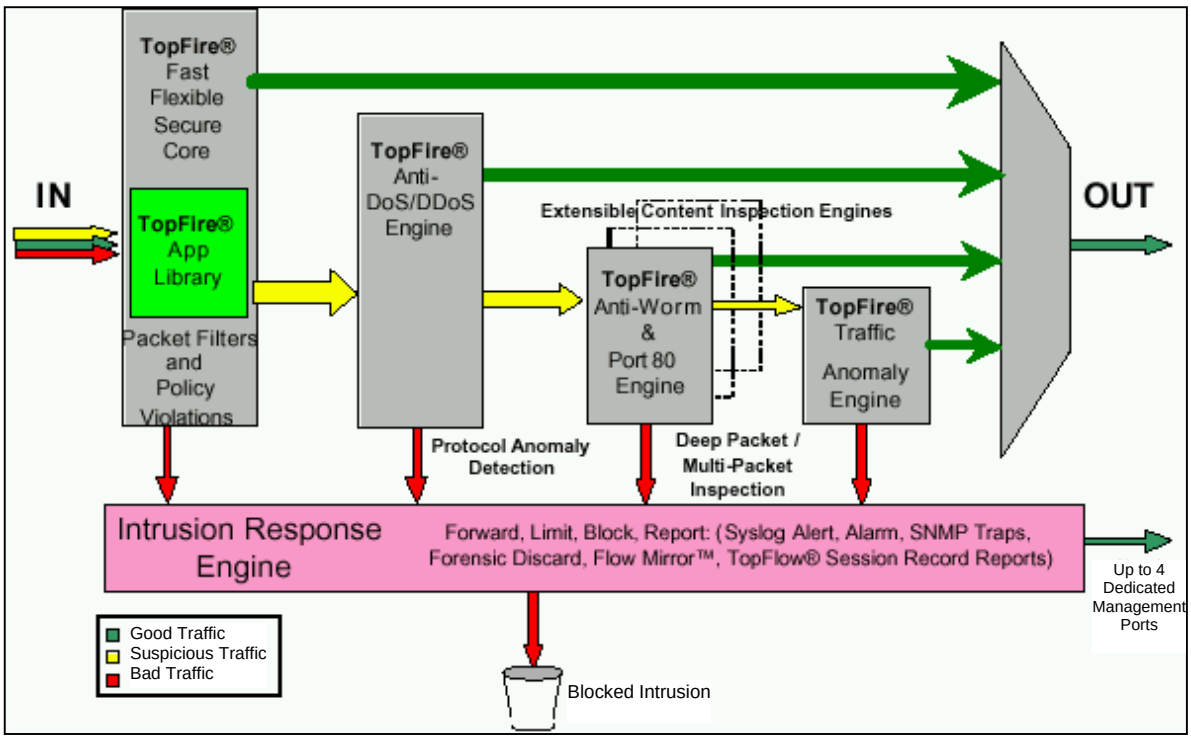

Figure 3.14: Top Layer Stateful Network Intrusion Prevention and Response Engine [TopLayer02]

It is not only important to identify all types of attacks, but also to be able to identify them efficiently. Therefore, the existing IDS architectures should be redesigned to add new detection methods effectively.

Some of the advantages and disadvantages associates to MMD are described subsequently:

Advantages:

- MMD is easy to implement for vendors since they know the mechanisms and they have the implementations in previous IDS products,
- MMD provides more accurate detection than single method detection.

Disadvantages:

- There is still the most important question: which of the existing detection mechanisms should be used in which order,
- Detection mechanisms that are fully depend on the traffic could only identify an attack based on that knowledge, but this knowledge is not enough to truly identify an attack most of the time.

3.4.2.2. Use of Neural Networks:

A research on integration of different security techniques at University of Southampton uses Neural Networks to detect attacks for their integrated security system [UY02]. A main purpose of the project is to integrate as many security functionality as possible into the firewall. They call their system as intelligent firewall. Their system contains, as they call, a smart detection engine to detect malicious data packets. Their claim is that Neural Networks would be enough for detecting known and, at some degree, unknown attacks. They also claim that their detection system can also detect viruses.

"An artificial Neural Network consists of a collection of treatments to transform a set of inputs to a set of searched outputs, through a set of simple processing units, or nodes and connections between them" [PJP03]. Subsets of the units are input nodes, output nodes, and nodes in the hidden layers that are between input and output nodes. The connection between two units has some weight. This weight used to determine how much one unit will affect the other. The most important property of a Neural Network is that it is inspired from the human brain and it learns, automatically, according to data inputs and data outputs. Neural Networks are divided into two according to the training modes [PJP03]:

- **Supervised training algorithms,** where in the learning phase, the Neural Network learns the desired output for a given input or pattern (called the "training set"). The Neural Network is told that if it sees this pattern it should give the desired output [MH02]. The well known architecture of this Neural Network is the Multi-Level Perceptron (MLP); the MLP is employed for Pattern Recognition problems.
- **Unsupervised training algorithms,** where in the learning phase, the Neural Network learns without specifying desired output. The Neural Network tries to find patterns within a data set in order to group them according to their most common features. This is very useful when dealing with large amount of raw data with little knowledge of interrelations between various fields in a vector [MH02]. Self-Organizing Maps (SOM) are popular unsupervised training algorithms; SOM are employed for classification problems.

Neural Networks have been used to solve complex problems, such as Pattern Recognition, hand-written character recognition, Statistical Analysis and they are very successful. Therefore, researches concerning the application of the Neural Network techniques, for the misuse detection model and the anomaly detection model in IDSs have begun in late 1990s and they are more and more going on. Those researches main concern is to reduce the False Positive rate in IDSs.

The origin of the false alarms comes from the dynamic nature of the systems and networks. Any system that uses signature-based detection or anomaly detection can easily identify legitimate traffic as an intrusion. Another limitation of those systems, especially for signature-based detection, is their inability to detect new or modified intrusion attempts until their database being updated. On the other hand, Neural Networks can generalize the past-observed behavior to detect new or modified attacks against system. The Neural Networks also posses the ability to classify patterns, and this ability can be used in attack classification.

The most common Neural Network used in IDS researches is MLP. In early researches, this model was used mainly on the application of anomaly detection on user behavior analysis. Later researches focused on using MLP as an alternate to signature based misused detection systems. Most recent researches rely mainly on SOM Neural Networks, which is based on unsupervised learning model. This model was applied to user behavior analysis and recently application and process analysis. Besides these researches, one genuine approach was the use of two different Neural Networks for attack detection and classification [MH02]. All this researches on the application of Neural Networks in IDSs have showed that there is a lot to do to understand the effect of different Neural Network topologies and, later, make this approach better.

Some of the advantages and disadvantages associates to Neural Networks are described subsequently:

Advantages:

- Use of Neural Networks would be good to detect new attacks,
- Neural Networks can generalize the past-observed patterns and can detect modified attacks,

- Neural Networks are computational models of the human brain that people are familiar to use.

Disadvantages:

- The success of Neural Networks depends mainly on the training set used in the learning phase,
- Since Neural Networks make large amount of computations, their performance in real-time traffic would be a problem,
- Neural Networks is now an ongoing research area and the correct number of layers and number of nodes for better or best attack detection is still a question.

3.4.2.3. ActiveResponse Technology:

An alternative approach is developed by ForeScout Technologies Inc. [ForeScout04]: ActiveResponse. ActiveResponse technology is a patented technology which uses pre-attack activities against the attackers. In this technology, the system responds proactively to the initial reconnaissance (recon) activity, instead of waiting for the attack itself.

This approach assumes that the network attacks follow a consistent pattern – a three step process. The potential attacker recons the target network – scanning and probing to discover the structure of the network, its vulnerabilities and configuration details. Then, the information returns from those recon activities to the attacker. Finally, the attacker uses that information to launch attacks based on the unique structure and characteristics of the target network. The system neutralizes this process using the following three-phase approach [ForeScout04]:

Phase 1: Receptor

The system continually monitors incoming network traffic, looking for any sign of network recon (Figure 3.15). This monitoring can be done with a very high level of sensitivity. Thus, even very slow recon activities can be detected. The system does not disturb security managers for recon events.

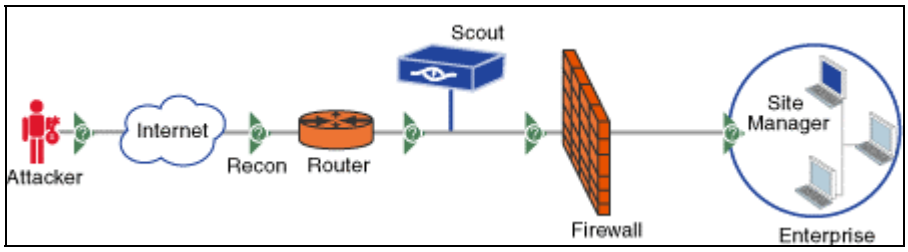

Figure 3.15: Phase 1 – Identifying the reconnaissance activity [ForeScout04]

Phase 2: Deceptor

When the system detects a recon activity; it identifies the type of the recon. Then, the system responds to the recon attempt with bogus information similar to that, which is being required. The system sends a "mark" as a response to the identified recon (Figure 3.16). The "mark" is deceptive data like the ones provided by honeypots. The "mark" usually consists of one or two packets of traffic. The "mark" specifically imitates the resource targeted by the potential attacker's recon - such as a TCP/UDP service or a NetBIOS resource. The "mark" appears completely in the expected format of the response. Therefore, the attacker will view this "mark" as valid, and will make use of it in any subsequent attack. In fact, the information in the "mark" is fabricated by the system which is all fake.

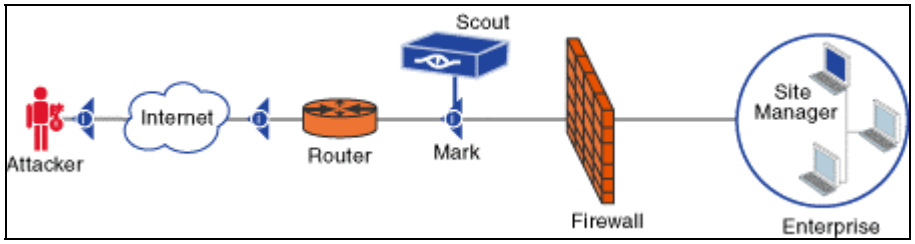

Figure 3.16: Phase 2 – Responding to the reconnaissance activity [ForeScout04]

Phase 3: Interceptor

When an attack is launched using the "mark", the system will be able to immediately identify it. Rather than depending on an attack signature, the system simply recognizes its own "mark". In other words, the system has placed a "mark" by which it can detect and intercept traffic coming from a source that previously performed suspicious recon. At this point, the system prevents the attack, alerts the security

manager, and can subsequently block all traffic from the suspected IP address (Figure 3.17). The attack can occur days or weeks after the recon activity or the attack may also come from a totally different IP address than the recon. Since the system looks for the "mark" it fabricated, its effectiveness is not influenced by either time delay or the use of a different source.

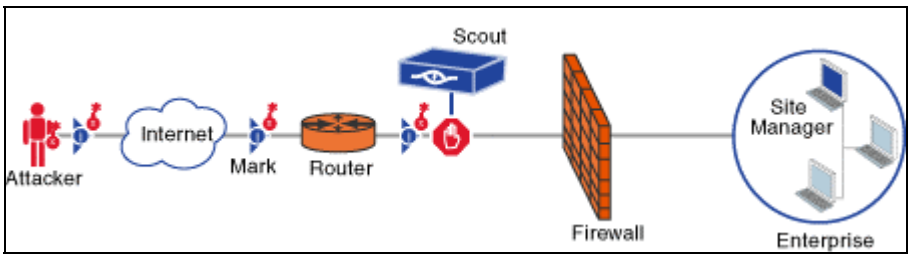

Figure 3.17: Phase 3 – Blocking the attacker [ForeScout04]

This system is distinct from other IPS products in that it is not an in-line solution; it does not sit in the traffic path and it is not another potential point of failure. It detects and blocks attackers without being in-line. If in-line functionality is required, the system can interact with an existing firewall for blocking capabilities. Thus, this system has two options to block attacks [ForeScout04]:

- **The system blocks by itself:** The system can block on its own, by injecting TCP reset to tear down TCP session. Session is terminated by sending a TCP reset message to the internal resource targeted by the attacker. The internal resource will, then, drop the session and the attack is prevented. The TCP reset is sent during the TCP handshake stage, therefore making session termination very efficient, and preventing the attack payload from reaching the internal resource.
- **The system signals the firewall for blocking:** The system can signal the firewall to change a rule and have the firewall block the attacker. This capability enables the firewall to block attackers utilizing additional protocols besides TCP. The system can update all or some of the firewalls in an enterprise network through a single management console. The system contains plug-ins for various firewall vendors.

Therefore, this system merges capabilities of honeypots and firewall to prevent attacks. Like honeypots, it provides deceptive data to the attacker. Like firewalls, it blocks the attacks. The system works like the approach of using marked bills to catch

criminals. Thus, this system owns a series of characteristics that distinguish it of other solutions of security. Some of the advantages and disadvantages associates to this technology are described subsequently:

Advantages:

- Since this technology only detects recon activities the generation of false positives would be not a case,
- This technology eliminate threats before they reach the network without requiring signature updates or manual intervention,
- This technology will prevent the attacks as long as they are followed by recon activities,
- The system would only have to know a finite number of well-known recon techniques and identify them, rather than an increasing number of known attacks and an unlimited range of unknown attacks.

Disadvantages:

- This technology does not drop the traffic during the attack detection process; in a busy network it could not prevent the attack on-time, before the payload of the attack reaches the target,
- This technology only detects attacks that are followed by a recon activity, therefore it is successful for preventing amateur attackers,
- Professional attackers can find a way to bypass this protection.

3.4.2.4. Cisco Threat Response:

A propriety technology to eliminate false alarms of IDSs is the Cisco Threat Response (CTR) [Cisco04] technology which works with Cisco Intrusion Detection System (IDS) sensors to provide an efficient intrusion protection solution. CTR technology virtually eliminates false alarms, and identifies real intrusions quickly. CTR has come out with the idea that having more information than the information gathered

from the network packets can help more accurate detection of attacks. The more you know about the network environment, the fewer false alarms you will have.

Current detection methods monitor the network traffic and detect an anomaly or attack signature. They do not know if the attacks were successful or not. It is the job of security managers to investigate the attacks where there are multiple of them against multiple hosts. Thus, security managers must select which attacks to investigate since they have a limited resource. Often, the attacks selected for investigation are failed; therefore the successful attacks may not be investigated.

CTR technology provides an automated, real-time analysis of each targeted host to determine whether a compromise has occurred. CTR technology achieves this by using a dedicated server sitting between IDS sensors and the IDS management console. For each alarm triggered, CTR scans the target host to see if the attack that triggered the alarm will actually have any effect. For example, if a sensor detects a Windows attack, CTR launches a scan to determine that if the attack will affect the target. If the target host is not a Windows system, CTR will downgrade the alarm. The alarm will be logged but no alert will be generated. But, if the target host is a Windows system, the attack would be indicated as a potentially successful attack.

CTR technology scans can be configured for a quick analysis or a detailed analysis to determine if the target host is vulnerable [MCA03]. This multiphase analysis includes the following steps [Cisco04]:

1. **Target operating system or device vulnerability**—CTR technology dispatches an agent to determine, in real-time, the operating system running on the targeted system. If the targeted attack is effective on that operating system then the target system is vulnerable to the attack.
2. **Patch-level check**—CTR technology checks patch status of the system to determine if appropriate patches have been applied to prevent the detected attack.
3. **Detailed system investigation**—CTR technology uses read-access privileges to make a detailed system investigation based on the attack type. The investigation may include:
    - Analysis of registry entries, system and log files,

- Searches for specific files or directories seeking attack traces,
- Other investigative methods to confidently determine the success or failure of an attack.

4. **Forensic evidence retrieval—**If it confirms an attack, CTR technology quickly collects forensic evidence and copies this information to a secure location for offline analysis. In this way, the intruder cannot avoid detection by altering these files.
5. **Confirmed Attack Notification**—CTR technology alerts the security managers with information about the attack, complete details on how the investigation was conducted, and copies of the forensic evidence gathered.

The following scenario describes how a common security attack is handled with CTR technology (Figure 3.18). The security attack used in this scenario is an IIS Unicode attack popularized by the Nimda worm, which targets Microsoft IIS servers.

In this scenario, the network-based IDS detects three possible attacks, and dispatches three alarms. CTR technology receives those alarms and immediately begins its real-time investigation of each individual system [Cisco04]:

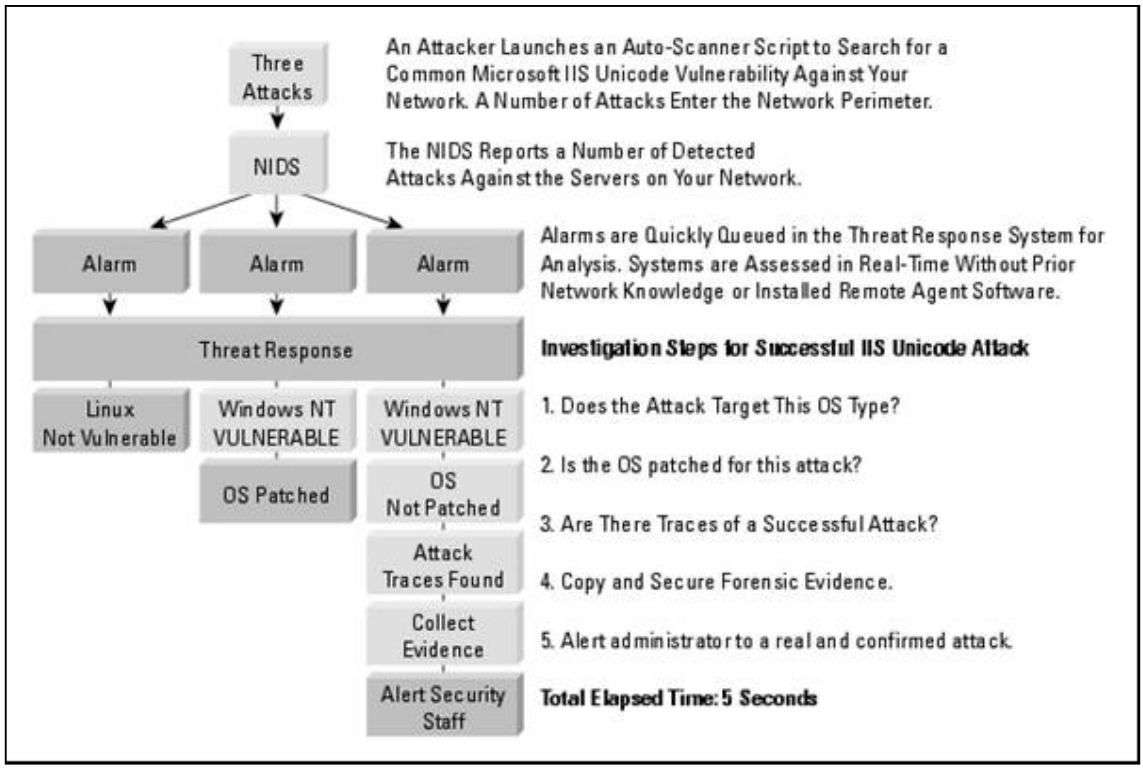

Figure 3.18: Intrusion protection with Cisco Threat Response [Cisco04]

System-1: Linux host

- The agent checks to determine the operating system of the targeted system.
- The operating system is Linux.
- This attack is not targeted to Linux. Thus, CTR downgrades the alarm.

System-2: Windows NT host (patched)

- The agent checks to determine the operating system of the targeted system.
- The operating system is Windows NT.
- This attack is targeted to Windows NT. Thus, CTR logs in to the host and check if the appropriate patches have been applied.
- Since the system is patched against the attack, CTR downgrades the alarm.

System-3: Windows NT host (unpatched)

- The agent checks to determine the operating system of the targeted system.
- The operating system is Windows NT.
- This attack is targeted to Windows NT. Thus, CTR logs in to the host and check if the appropriate patches have been applied.
- The system is not patched against the attack. Thus, CTR checks web server for signs of a successful attack.
- The signs of a successful attack are found. Then, CTR checks for other signs of intrusion.
- Dropper files or other evidence are found. Thus, CTR copies all collected forensic evidence to the central server.
- Then, CTR escalates the attack to critical status, giving it top priority on the console and alert security managers.

CTR is a new technology patching the detection mechanisms of IDSs. Cisco claims that CTR will eliminate up to 95 percent of false alarms [MCA03]. If it is successful, it will likely be used by IPSs to check the alarms correctness. The advantages and disadvantages of CTR technology can be summarized as follows:

Advantages:

- CTR technology is focused on investigating the target of the attack,
- CTR technology requires no knowledge of the network architecture other than a range of IP addresses to protect [Cisco04],
- CTR technology requires no remote-agent deployment on hosts that it protects,
- CTR technology avoids disturbing security scans by performing real-time analysis,
- CTR technology also copies and saves all the forensic evidence to a safe location before the attacker tampers them.

Disadvantages:

- CTR generates traffic for its investigations and this could affect the network performance.

3.4.3 NIPS Responses

NIPSs should have various types of responses to let security managers configure the response and reporting mechanisms according to the needs and the policies of the organization. These responses can be active (proactive response action) or passive (report action) depending on the configuration and range from "disable" (no response, no reporting) to "monitor" (selected reporting mechanisms, but no response), and finally to "prevent" (selected response and reporting mechanisms). A NIPS should provide some or all of the following responses options:

Active Responses:

- **Block the traffic:** IPSs can drop any suspicious packet and the remainder traffic from that source.
- **Forward the traffic up to a certain bandwidth or a certain number of TCP connections:** IPSs can work like traffic shapers and allow suspicious traffic up to a limit that does not affect the valid traffic.

- **Send TCP reset:** IPS can detect a suspicious packet in the flow of a connection and send TCP reset to both ends of that connection. Then, it can block it to prevent new connections.
- **Change firewall or router configuration:** IPSs can add or remove rules to firewalls or change the access control lists of routers to block suspicious traffic. This reduces the process time spend on suspicious packets as they are blocked at first point.
- **Clean viruses on files:** IPS can clean inflected files before forwarding them to the target host. This would prevent viruses and worms to effect hosts that do not have an antivirus software installed or updated.
- **Send bogus response:** IPSs can work like a honeypot and misinform the attacker. The response to a reconnaissance act will be "marked" response with some bogus data so that when the attacker comes back and tries to exploit the target IPS will see the "marked" data and stop all traffic coming from that source [DN03]. The "marked" data can be in the data field or one of the header fields of the response packets.

Passive Responses:

- **System logging:** IPSs can log the attacks, the response action that is done and the attack details such as traffic dump.
- **Send alarm to security manager:** IPS can alert the security manager that an attack has arisen and the selected response action is done. These alarms can be configured by the security managers and range from an administrative alert or SNMP trap to sending an email to the configured email address or sending a message to the security manager's pager. Depending on the alerting mechanism, the details of the attacks send to the manager can change.
- **Mirror flow:** IPSs can have a mirror port to send the unfiltered inbound and outbound traffic to a network analyzer.
- **Forensic discard:** IPSs can have a port to copy filtered packets for safety and offline observation for forensic analysis.

CHAPTER 4

NIPS SPECIFICATIONS

In this chapter, the functional requirements of a network-based IPS are identified and a scalable NIPS architecture, which is composed of distributed NIPS sensors, a centralized management server and a distributed graphical user interface, is proposed.

4.1 NIPS Requirements

Since functional requirements are at a higher level of abstraction than security architectural mechanisms, they need to be clarified before discussing an architecture. Functional requirements of a network-based IPS are outlined below:

- **Inline operation:** In order to perform real time protection, NIPSs must operate inline at choke points of the network. NIPSs can only take the necessary action immediately when they operate inline, discarding any suspicious packets before they reach their target and blocking the remainder flow from that source.
- **Ability to perform various types of detection analysis:** NIPSs are only as valuable as their detection engines. Success or failure of a NIPS depends mainly on its detection engine. There are various detection methods that are used in existing IDSs; each has success at detecting different types of intrusions. Mixing these various methods to use their superior parts and eliminating their weak points can form out a system that is more reliable than any of these methods alone. This would result in less false positives which are the main problem of existing IDSs.
- **Detection accuracy:** NIPS must detect attacks and should not block the valid traffic flow. Since NIPSs operate inline, false positives can lead to a DoS (Denial of Service) condition and become a new tool for attackers. The user must be able to trust that the NIPS is blocking only the malicious traffic [NA03]. NIPS should not block the valid traffic and prevent employees doing their jobs.

- **Low latency:** Since the NIPSs operate inline and all traffic has to flow through them, the latency on these devices affects the network performance. Packets should be processed quickly enough that end nodes can not sense the performance degradation. The over all latency of NIPSs must be as minimal as the latency of other inline devices such as firewalls, router, and load balancers.
- **High performance:** Packet processing must be at the real life traffic speed. Poor performance of NIPS will result in slow network, and even, lost of packets. Thus, a NIPS should perform analysis at very high data rates; degradation in network performance is not acceptable.
- **Reliability and availability:** Fail of an inline device will directly affect the network up-time. Since NIPSs are installed on choke points, any failure can cause the lost of a vital network path and, again, can lead to a DoS condition. Thus, an extremely low failure rate is very important in order to maximize the network up-time, and as an assurance, the device should support fail-over to another NIPS operating in a fail-over group or provide fail-open option [NA03]. It is also important that rebooting of inline devices will turn into network downtime for the duration of reboot.
- **Easy management:** NIPSs allow the security managers to choose the response they want among various response mechanisms. Since NIPSs are not only detecting attacks, but also preventing them by limiting or blocking which directly affects the network performance. Thus, configuring a NIPS is a complex job. It is important to make the security managers' job of configuration as easy and simple as possible by providing them a user friendly interface to set and change configuration and eliminate the dreadful results of configuration errors [TopLayer02].
- **Scalability:** An NIPS deployment should be scalable in performance and management. NIPS could be deployed to medium and large networks without significant performance degradation. NIPS deployment should also provide scalable management for multiple Sensors deployed at choke points of the network.
- **Safety of forensic data:** Beyond detecting and preventing attacks, NIPSs should save the evidence of an intrusion for forensic analysis. In order to do this, forensic

data could be copied for safe and offline observation. Mechanisms for the safety of these data should be in place.

- **Regular summary reporting:** Security managers spend most of their time to analyze the logs of IDSs. In order to save time for security managers, summary reports at regular bases or on demand will be very useful as they provide at a glance overview of the security state of the network.
- **Drill-down data analysis capability:** In order to minimize administration effort, NIPS should have a mechanism to allow security managers access individual packets from summary reports.
- **Virus detection and cleaning capability:** NIPSs should also detect worms and viruses and protect all hosts connected to the network from these threats. Even the hosts with antivirus software installed can be inflected by new worms and viruses if it is not updated with the new virus signatures. Virus detection capability will protect those hosts with antivirus software installed and those without it.
- **Easy update:** NIPSs will require updates in order to detect new attacks and viruses. That update process must be very easy and should be done without rebooting. Since several new viruses and attacks are being detected every day, rebooting for each update will ruin network availability. In addition, the process of updating signatures can also be an automatic process.
- **Transparency:** NIPS deployment should not require any configuration changes at hosts. This also makes it hard for attackers to perceive the existing of a NIPS since it will not have an IP address.
- **Modularity:** New detection methods and response mechanisms could be added to the NIPS. Also, the existing ones can be upgraded.
- **Active response:** NIPS deployment should have mechanisms in place for automated proactive response when an attack occurs. These responses can range from TCP reset, firewall or router reconfiguration to blocking or limiting the network traffic.

4.2 NIPS Architecture

A scalable NIPS uses a three-tier architecture that consists of the NIPS Sensor, NIPS Management Server, and NIPS User Interface (Figure 4.1).

- **NIPS Sensor –** monitors the network on which the NIPS is installed. The Sensor is a hardware or software appliance that runs NIPS Sensor software. They can operate either as a traditional passive IDS sensor or as an advanced in-line NIPS.

- **NIPS Management Server** – software component installed on a computer. It stores and manages all Attack Objects (including attack signatures and protocol anomalies), log information, rule bases, and Security Policies. A single Management Server can manage multiple Sensors. Centralized management of policies and logs makes it easy for administrators to manage the network security of a large distributed enterprise. The centralized management server reports to multiple user interface consoles for alerts. It also sends e-mail alerts to all security managers and/or sends SNMP traps to multiple managers.

- **NIPS User Interface (UI)** – a graphical interface for interacting with the NIPS. The UI is used to access remotely and manipulate the information stored on the Management Server. It is a web-based or Java-based interface for flexibility. UI can be used to work with security policies, signatures and events generated by NIPS Sensors. Multiple user interfaces can connect to a single Management Server to perform all management operations.

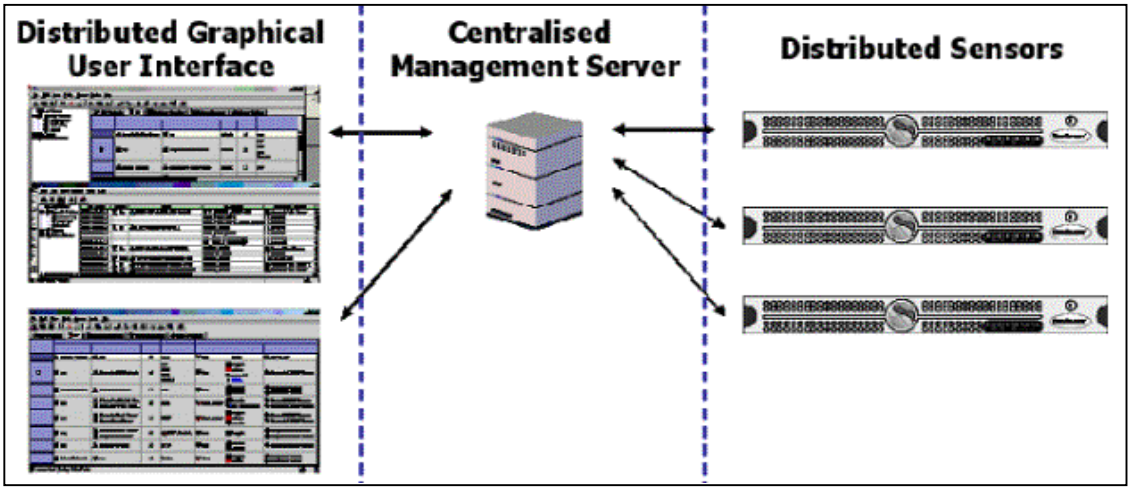

Figure 4.1: Three-tier NIPS architecture [NetScreen04]

4.2.1 NIPS Sensor

The primary task of NIPS Sensor is to detect suspicious and anomalous network traffic based on specific rules defined in NIPS rule bases. If the Sensor is running in-line, it can also take a predefined action against malicious traffic.

NIPS Sensor is a software or hardware appliance including, at least, two interfaces for packet detection/forwarding, one interfaces for management, and one interface for sending TCP reset in passive sniffing mode.

NIPS Sensors can be deployed as active gateways or passive sniffers. A passive sniffer Sensor connects to a switch or hub in promiscuous mode and sniffs the network traffic. The Sensor monitors network traffic, records security events, and can create alarms for attacks. However, because a sniffer Sensor cannot take direct action against the attack, it cannot prevent it. It can only send TCP reset packets through a non-forwarding interface.

An active gateway Sensor sits between the network and a firewall, or a network and a DMZ, and takes an active role in protecting the network. When it detects intrusions or attacks defined by security policies, the Sensor can drop the suspicious connection, or drop only the suspicious packets. Because active gateway Sensors are installed in-line they can take direct action against the attack, thus preventing it.

When operating in-line, it is essential that traffic is not interrupted by device failures. It is possible to configure the Sensors to operate in fail-open mode. It is also possible to deploy the NIPS Sensors in a high availability configuration to provide failure protection, either in a standalone configuration or using third-party hardware. In high availability, Sensors are joined together in a cluster that provides failure protection and/or load balancing. In load balancing mode, all Sensors in the cluster share network traffic equally, and if a Sensor fails, network traffic is redirected to the other Sensors in the cluster. In failure protection mode, a primary Sensor handles all network traffic while a secondary Sensor stands by. If the primary Sensor fails, network traffic is redirected to the secondary Sensor.

NIPS Sensors can be installed in Layer 2 transparent mode. This allows installation without any changes to networking architecture. Another benefit of this install mode is that Sensors can also pass non-IP traffic. This also simplifies in-line deployment.

The Sensors communicate with the Management Server through the management interface over a secure, encrypted link. This secure channel is used to transmit log and policy information. The Sensor stores logs on the local hard drive initially, and then transmits the logs to the Management Server. If communication between the Sensor and Management Server fails, the Sensor attempts to restore communication and stores new logs on the local file system until available disk space is consumed. If no disk space remains and communication has not been restored, the Sensor first deletes system logs (debug and error messages) followed by packet captures to free up disk space. If communications cannot be restored before all local disk space is consumed, the Sensor starts to override the previous attack logs or ceases to detect new attacks depending on the configuration.

The Sensors can understand many network and application layer protocols such as HTTP, FTP, SMTP, SNMP, RPC, SMB, NetBIOS as well as many Trojan communication protocols.

NIPS Sensor Engine:

The NIPS Sensor operates as an in-line, active device or an out of band passive device that looks at all traffic and determines intrusions. If instructed by the security policy, Sensors can deny traffic associated with the intrusion by dropping the connection or associated packets (when operating in-line). Sensors analyze and validate the traffic to its basic protocol elements and inspect specific protocol fields to improve accuracy. The Sensors perform IP fragment reassembly and TCP stream reassembly, and perform complete protocol analysis all the way up to the Application Layer. In addition to leveraging protocol analysis for buffer overflow detection, it is also possible to write powerful and flexible user-defined signatures. Responses are defined in the Sensor rule bases to specify what actions Sensor takes when a particular attack is

detected. Figure 4.2 shows the components of the NIPS Sensor engine and the traffic flow through those components:

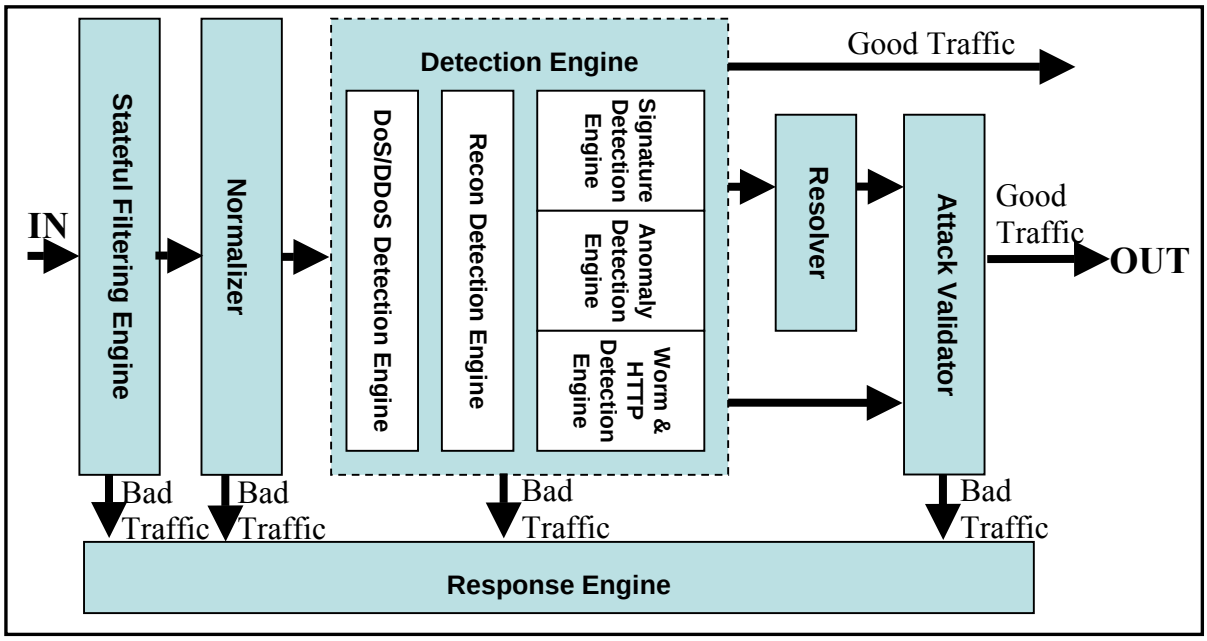

Figure 4.2: NIPS Sensor engine

Stateful Filtering Engine: It is the stateful packet filtering firewall. It tracks the active connection though a state table; allowing the active connection and checking only the new connections from its rule database. This saves time since it does not check IP addresses and ports of all packets to permit or block them.

Normalizer: It is the traffic normalizer. Its main function is to make protocol analyzes and remove any traffic protocol ambiguities; meaning that the traffic being interpreted by Sensor and the traffic received at the protected end-system is identical [GF02]. All packets are analyzed into their corresponding protocol fields and normalized before being passed to the Detection Engine. During the protocol analysis process, the Sensor gathers detailed information about the communication state, protocol and application.

Removing protocol anomalies protects the end systems by cleaning up potentially harmful traffic in real-time. It also allows Sensor to prevent hackers from "fingerprinting" a host system [GF02]. This makes it hard for hackers to launch subsequent attacks against known vulnerabilities in host network hardware or software

resources. Normalization also prevents any attempts to evade the Detection Engine while improving attack detection accuracy.

Besides removing protocol anomalies, normalizer also reassembles fragmented packets therefore, prevents fragment overlap which is used in evasion attacks such as Teardrop and other Layer 3 protocol anomaly based attacks. During the normalizing process, it drops invalid packets that have bad checksums or invalid sequence numbers [GMAL03], and detects IP spoofing by comparing the IP addresses of packets to the IP addresses of devices on the protected network. An IP address is considered spoofed if an incoming packet uses an IP address that belongs to a device on the internal network, or if an outgoing packet uses an IP address that does not belong to a device on the internal network. This normalization process allows the detection engine to be much simpler.

Detection Engine: It does the job of detecting an attack, reconnaissance (recon) activity, or DoS/DDoS attempt. Since no single technique or technology is a panacea, guaranteeing protection against all known, unknown and DoS attacks, it integrates multiple methods to detect various types of attacks [GF02]. Combining multiple methods significantly improves detection rates. It composed of the following sub-engines:

- **DoS/DDoS detection engine –** detects denial of service attacks. It can use a combination of threshold-based detection and self-learning profile-based detection techniques. With threshold-based detection, data traffic limits can be configured to ensure servers will not become unavailable due to overload [GF02]. At the same time, self-learning methodologies enable engine to study the patterns of network usage and traffic for an IP range or an individual host in order to understand the wide variety of legal, though unusual, usage patterns that may take place during legitimate network operations.
- **Recon detection engine** – detects recon activities to identify the potential attacker. There exist a finite number of well-known recon techniques. Recon detection engine analysis traffic against these techniques.
- **Worm and HTTP detection engine** – detects worms and HTTP attacks by using HTTP protocol validation, normalized string matching and URI length-checks.

Worm and HTTP detection engine decodes the URI as a web server would and observes the result to see if it is an exploit [TopLayer02]. With this decoding capability, the engine uses only one signature to detect variants of an exploit.

- **Signature detection engine** – detects known attack patterns. Signature detection engine uses so-called stateful signatures to reduce false alarms. Stateful signatures enable context-sensitive signature detection, leveraging state information within data packets, utilizing multiple pattern matches, and detecting attack signatures [GF02].
- **Anomaly detection engine** – detects attacks by employing statistical, and/or application anomaly detection techniques. Anomaly detection engine helps guard against are buffer overflow attacks, backdoor malicious attacks installed via a Trojan or by an insider, and insider violation of security policies, such as installing a game server or a music archive on the network [GF02].

As the normalized and analyzed traffic passes through the detection engine, firstly, DoS/DDoS detection engine inspects the traffic to find an indication of denial of service attack. If it finds, it forwards the traffic to the Response Engine. If no DoS attempt is discovered then the traffic is forwarded to the recon detection engine to find out a recon attempt. If recon engine finds, it forwards the traffic to the Response Engine. If it discovers no sign of a recon attempt it forwards the traffic to signature and anomaly detection engines or worm and HTTP detection engine depending on the protocol analysis done by the Normalizer. If those engines find out nothing, the traffic is forwarded to the target host. If worm and HTTP engine find out an exploit, it forwards the traffic to the Attack Validator in order to be sure that the target host is vulnerable to the detected exploit. If the signature and/or anomaly engines detect an attack, it forwards the traffic to the Resolver in order to combine and process the results of these engines. By using various detection methods, the types of attacks detected are maximized and the detection accuracy is improved.

Resolver: It is an intelligent, model-based reasoning component that makes its decisions by correlating the information gathered from the Detection Engines with more global information in a large knowledge base. The objective of correlation is to recognize the intrusion plan that is currently executed by the intruder [CM02]. The role

of the Resolver is to determine the likelihood of an attack based on the correlated information. Those information are evidence for and against an attack based on the available intrusion scenario models that are stored in the knowledge base [LTGJ92]. The Resolver would be activated when an intrusion is detected by either the signature-based detector or the anomaly detector engine or both. The knowledge base used by the Resolver is updated by the Management Server periodically or on demand bases depending on the configuration. The correlation of information from various sources improves the detection accuracy. It also eliminates the possibility of multiple alarms due to the suspicious packets.

Attack Validator: It gathers information about the target in local network and builds a table to see if that target is vulnerable to the detected attack [GMAL03]. If the target is vulnerable it validates the attack, then prevents it, alerts the security manager and logs it to the management console giving high priority [Cisco04]. The Management Server updates the information about the target hosts periodically or on-demand bases depending on the configuration. This engine will reduce the number of event records; therefore the security managers will not spend hours examining logs.

Response Engine: It is the component that applies the configured response to an attack. Every packet that passes through the NIPS Sensor is analyzed by the detection component and if they are defined as malicious the Response Engine is responsible to take the preconfigured action against it. Depending on the configuration of the Response Engine, NIPS Sensor can passively monitor attacks or proactively prevent them. The responses to attacks can range from real-time notifications of the security managers to complete blocking of attacks in progress. The responses include the following options:

- System logging,
- Sending alert to security manager by paging, sending SMS, sending email, using SNMP traps or other mechanisms,
- Mirroring the flow,
- Saving forensic data,
- Sending bogus response,
- Forwarding the traffic up to a certain bandwidth or certain number of connections,

- Sending TCP reset,
- Changing firewall or router configuration,
- Dropping malicious packet, connection or simply blocking the traffic.

Response Engine communicates with the Management Server in order to save forensic data or send alert to security mangers through pager, SMS, email or SNMP traps. In other cases it directly takes the configured action against an attack.

4.2.2 NIPS Management Server

NIPS Management Server consists of the software resources that are used to configure and manage the NIPS Sensors. The Management Server software can be installed on the NIPS Sensor for single Sensor deployments, leading to a two-tier architecture, on small networks. In all other situations, the Management Server must be installed on a separate computer.

The Management Server centralizes the logging, reporting, and security policy management for the system. All logs, attack signatures and security policies are stored in a database and are managed using the User Interface by one or more administrators. The Management Server consists of the following components (Figure 4.3):

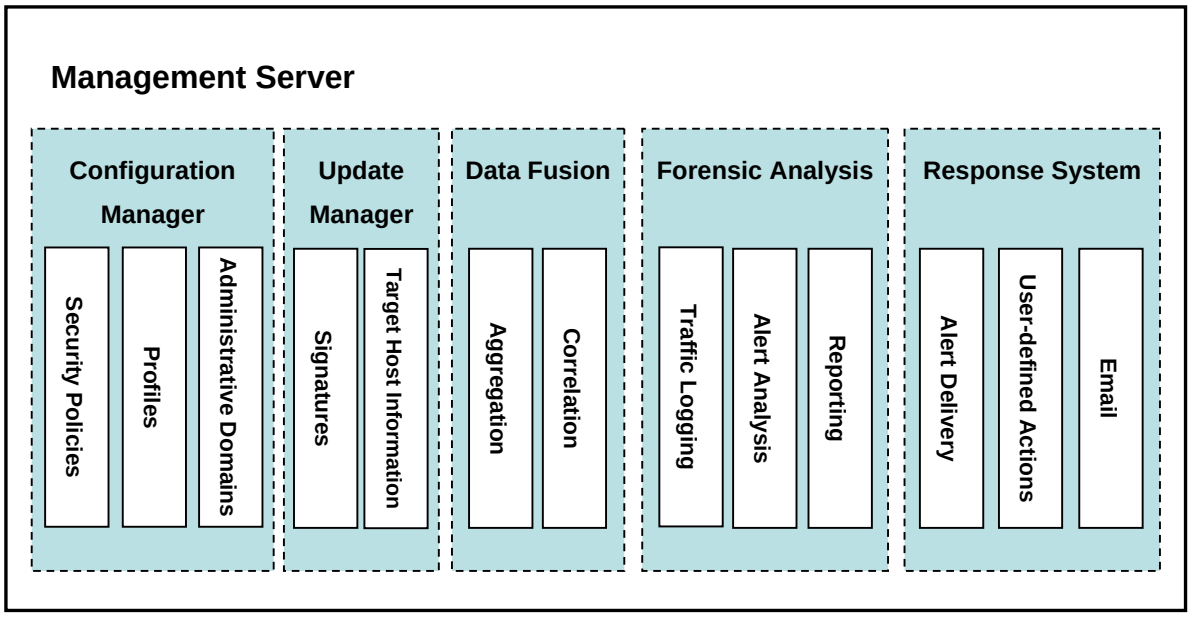

Figure 4.3: NIPS Management Server

- **Configuration Manager:** It manages the configuration of the security policies that are local to a part of the network or global to the whole enterprise network. Those policies are list of rules that will be applied or are being applied to a part of the network traffic or to all enterprise-wide traffic. Configuration Manager also controls configuration of multiple NIPS Sensors at various points in the network. Configuration Manager deploys the configuration information changed through the User Interface to the NIPS Sensors. In order to make the task of management and control of the multiple NIPS Sensors easier, Sensors are grouped into one or more administrative domains that can be administered and monitored by one or more users. Administrative domains may contain one or more NIPS Sensors. Configuration Manager controls the assignment of Management Server users to one or more administrative domains with a different administrative privilege for each domain. The privileges of the users and their User Interface configurations are also managed by the Configuration Manager.

- **Update Manager:** It manages the update status of the signature definitions and target host information on NIPS Sensors. It is responsible to keep the signature databases of NIPS Sensors updated. It checks for the signature definition updates, downloads new signatures and distributes them to the NIPS Sensors on the network. It is also responsible to gather information about the target hosts, such as the operating system, patch status, user status through various scan techniques or manual configuration and distribute them to the NIPS Sensors on the network. Update Manager may be configured to update the target host information periodically or based on a packet to or from a previously non-existing internal host.

- **Data Fusion:** It is responsible to aggregate alerts and logs by consolidating them from multiple NIPS Sensors into a single, central repository. The aggregated repository can contain interrelated logs and alerts that are associated to the same intrusion. Data Fusion aggregates interrelated records to single meaningful incident for an effective summarization. Data Fusion can also link alerts and events over time through correlation. This would provide meaningful attack summaries to managers and save time while analyzing attacks.

- **Forensic Analysis:** It provides the intelligence to inspect events and to extract summarization alerts for effective analysis including drill-down, filtering, sorting, and grouping. This information is used to system hardening or criminal prosecution. It is responsible for the logging of suspicious traffic in a safe repository. It provides periodic and on-demand customizable summary reports. It also has the necessary engines to analyze summary reports and drill down to traffic logs for a specific incident. It can also provide a query manager for individual analysis.

- **Response System:** It is the complement of the Response Engines on NIPS Sensors. It delivers alerts from the Response Engines on multiple NIPS Sensors on the network and forwards them through the configured mechanisms. Those mechanisms are the ones that would be much more effective to be centralized such as, SMS, email, SNMP trap. It also provides users to define and store their own actions that are mixtures of existing response actions and forwards those definitions to the NIPS Sensors on the network.

Management Server communication with the other two tiers of the NIPS system (the Sensor and User Interface) is encrypted and authenticated to provide an additional level of security.

The Management Server gathers log records for security events using UDP. Alerts are stored in a MySQL or a commercial database. The data fusion capability allows aggregation and correlation of log data. Consolidation of logs helps to reduce the total number of unique events that the Sensors generate by logically combining large numbers of identical events into a single alert. This summarized data are used for reporting. Security managers can easily analyze these summary reports and, also, drill down to perform detail analysis. The Management Server also provides a flexible query manager.

4.2.3 NIPS User Interface

The NIPS User Interface (UI) is software that provides a graphical environment for centrally managing NIPS Sensors. The UI is a web-based software application that

can be accessed from multiple computers through a web browser that supports SSL (Secure Socket Layer).

Although the UI supports multiple users, only one user at a time can take control of the Management Server – this eliminates concerns about synchronization or data loss. Each administrator can configure the UI with his own preferences – Management Server stores user preferences and custom Log Viewer views in the central database so that users do not have to reconfigure the views when accessed from different machines [NetScreen04].

The UI profiles two main views: the dashboard and main user interface. The Dashboard provides a quick overview of attacks detected and blocked (both as graphs and "Top 5" reports), device performance, and system statistics while the main user interface provides the functions and windows to manage the system.

The web-based UI is consisting of three panes:

- **Banner** – displayed at the top of the browser window, providing links to access the help table of contents and fast access to user-preferred screens.

- **Navigation Tree** – displayed down the left side of the browser window, containing folders that provide access to management functions such as events, reports, packages, devices, and roles.

- **Information and Configuration Area** – displayed in the center of the browser window, containing several separate windows that enable the administrator to access and configure the filters, examine connection set-up rates, and examine information about attack events.

As the client connects to the Management Server, a prompt for the user name and password is displayed. A Super User account is included by default, but it is possible to set up multiple administrator accounts, each with different roles. There are three main accounts:

- **Super User** – has authority to view all screens and administer all NIPS Sensors.

- **Administrator** – has NIPS Sensor administration authority. This role has the ability to administer NIPS Sensors which he has been granted access and view the related screens.

- **Operator** – has view access to all screens, but may not perform any of NIPS Sensor management or administration functions.

It is also possible to apply more granular access controls by restricting individual users to specific devices and/or segment groups. Disallowing device access, for example, would prevent a user from seeing and making changes to the device configuration while allowing them to manage and deploy policies. Restricting a user to specific segments or segment groups will ensure that each administrator can only deploy policies to that segment or group of segments (which can span multiple sensors). This means that if a user is restricted to the DMZ segment group he can deploy policies only to the segments which make up that group, making this perfect for large corporate or managed service environments.

CHAPTER 5

NIPS IMPLEMENTATION

In this section, an explanation of the implementation of the NIPS appliance will be given. The NIPS appliance is a modified version of Snort [Snort04], an open source, lightweight network intrusion detection system. The appliance receives a network packet from one interface through WinPcap [WinPcap04] driver, forwards it to the Snort [Snort04] detection engine and based on the result of the detection engine it either forwards the packet to the other interface through WinPcap [WinPcap04] driver or drops the packet.

"Snort is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture" [Snort04].

The Snort [Snort04] intrusion detection system was written using the C programming language as an application intended for use on a Unix operating system, relying mostly on a low-level independent communications driver referred to as Pcap. The effort by a number of supporters, Snort [Snort04] has been ported to the recent Microsoft Windows operating systems including Windows NT, 2000, XP and others. Snort [Snort04] requires the same low-level drivers, WinPcap [WinPcap04], in order to function on the Windows operating system.

WinPcap [WinPcap04] is a free, public system for direct network access on Windows operating systems. Snort [Snort04] needs a low level view in order to directly handle the network traffic. Therefore, it needs raw access to the network which is provided by WinPcap [WinPcap04].

Before explaining the implementation, it would be better to explain WinPcap [WinPcap04] and Snort [Snort04] architectures.

5.1 WinPcap Architecture

Most applications access the network through widely used system primitives, like sockets. By using sockets, operating system copes with low level details (protocol handling, flow reassembly, etc.). WinPcap [WinPcap04] provides access to raw network data without protocol handling, or flow reassembly.

The purpose of WinPcap is to provide direct network access to Windows applications; it provides facilities to [WinPcap04]:

- capture raw packets, both the ones destined to the machine where it's running and the ones exchanged by other hosts (on shared media),
- filter packets according to user-specified rules before dispatching them to the application,
- transmit raw packets to the network,
- gather statistical values on the network traffic.

This set of capabilities is obtained by means of a device driver, and a couple of DLLs. WinPcap architecture includes a kernel-level packet filter, a low-level dynamic link library (packet.dll), and a high-level and system-independent library (wpcap.dll). Figure 5.1 shows the components of WinPcap architecture [WinPcap04].

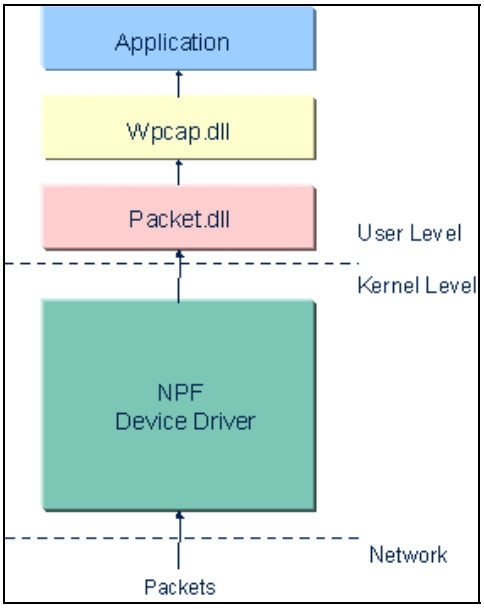

Figure 5.1: Components of WinPcap architecture [WinPcap04]

First, a capture system needs to bypass the protocol stack in order to access the raw data transiting on the network. This requires a portion running inside the kernel of operating system, interacting directly with the network interface drivers. This portion is much system dependent, and in WinPcap it is conceived as a device driver; called Netgroup Packet Filter (NPF), and currently different versions of the driver for Windows 95, Windows 98, Windows ME, Windows NT 4, Windows 2000 and Windows XP are provided. These drivers offer both basic features like packet capture and injection, as well as more advanced ones like a programmable filtering system and a monitoring engine. Thus, it can be used to restrict a capture session to a subset of the network traffic (e.g. it is possible to capture only the FTP traffic generated by a particular host), and the monitoring engine provides a powerful but simple to use mechanism to obtain statistics on the traffic (e.g. it is possible to obtain the network load or the amount of data exchanged between two hosts) [WinPcap04].

Second, the capture system must export an interface that user-level applications will use to take advantage of the features provided by the kernel driver. WinPcap provides two different libraries: packet.dll and wpcap.dll. The first one offers a low-level API that can be used to directly access the functions of the driver, with a programming interface independent from the Microsoft operating system. The second one exports a more powerful, system independent set of high level capture functions that are compatible with libpcap [TCPDump04], the well known Unix capture library [WinPcap04]. These functions allow capturing packets in a way independent from the underlying network hardware and operating system. Besides that, libpcap [TCPDump04] compatibility allows writing portable network tools that will work on the whole Windows operating system family and on all the major Unix flavors. One major example of this kind of programs is Snort [Snort04].

5.2 Snort Architecture

Snort [Snort04] has modular plug-in architecture. There are three types of plug-in available in Snort [Snort04]: detection plug-ins, preprocessors, output plug-ins. Detection plug-ins check a single aspect of a packet for a value defined within a rule

and determine if the packet data meets their acceptance criteria. For example, the TCP flags detection plug-in checks the flags section of TCP packets for matches with flag combinations defined in a particular rule. Detection plug-ins may be called multiple times per packet with different arguments.

Preprocessors are only called a single time per packet and may perform highly complex functions like port scan detection, TCP stream reassembly, IP defragmentation, http request normalization, or Telnet decode. They can directly manipulate packet data and even call the detection engine directly with their modified data. They can perform less complex tasks like statistics gathering or threshold monitoring as well.

Output plug-ins are also called once per packet after the preprocessor and detection engine. They provides real-time alerting capability, incorporating alerting mechanisms for syslog, user specified files, or a UNIX socket. They also provide logging of packets in many formats, including tcpdump [TCPDump04] binary format or decoded ASCII format to a hierarchical set of directories that are named based on the IP address of the remote host. Database or XML logging plug-ins exists, as well.

Snort [Snort04] has three primary functional modes: sniffer, packet logger, and network intrusion detection system. Sniffer mode simply reads the packets off of the network and displays them for you in a continuous stream on the console. Packet logger mode logs the packets to the disk. Network intrusion detection mode is the most complex and configurable configuration, allowing Snort [Snort04] to analyze network traffic for matches against a user defined rule set, and perform several actions based upon what it sees.

The rules contain the information that defines the who, where, and what of a packet, as well as what to do in the event that a packet with all the attributes indicated in the rule should show up. The first item in a rule is the rule action. The rule action tells Snort [Snort04] what to do when it finds a packet that matches the rule criteria. There are five available default actions in Snort [Snort04]: alert, log, pass, activate, and dynamic.

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

One can also define his own rule types and associate one or more output plug-ins with them. He can then use the rule types as actions in Snort [Snort04] rules.

Snort [Snort04] is developed with the performance, simplicity, and flexibility in mind. Snort is logically divided into multiple components. These components work together to detect various attacks and to generate output in a required format. These components ride on top of the libpcap [TCPDump04] or WinPcap [WinPcap04] promiscuous packet capturing library, which provides a portable packet sniffing and filtering capability. Snort consists of the following major components [RR03]:

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

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

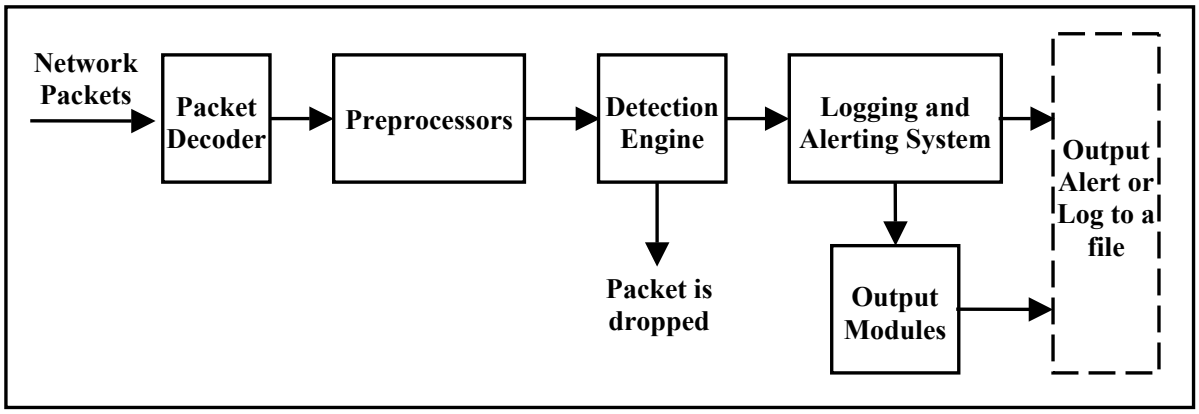

Figure 5.2: Components of Snort [RR03]

5.2.1 Packet Decoder

The packet decoder takes packets from different types of network interfaces and prepares the packets to be preprocessed or to be sent to the detection engine. The interfaces may be Ethernet, SLIP, PPP, WiFi and so on [RR03]. It parses the packet and decodes the string of bytes into a packet structure that is formed of protocol fields and flags. Each subroutine in the decoder imposes order on the packet data by overlaying data structures on the raw network traffic. These decoding routines are called in order through the protocol stack, from the data link layer up through the transport layer, finally ending at the application layer. During this decoding process, it validates the length and checksum fields. It then forwards the valid packets to the preprocessors.

5.2.2 Preprocessors

When a packet is received by Snort [Snort04], it may not be ready for processing by the main Snort [Snort04] detection engine and application of Snort [Snort04] rules. For example, a packet may be fragmented. Before searching a string within the packet or determine its exact size, defragmentation is required by assembling all fragments of the data packet. On IDS, before applying any rules or try to find a signature, the packets have to be reassembled [RR03]. The job of a preprocessor is to make a packet suitable for the detection engine to apply different rules to it. In addition, some preprocessors are used for other tasks such as detection of anomalies and obvious errors in data packets, decoding of HTTP URI. All enabled preprocessors operate on each packet. There is no way to bypass some of the preprocessors based upon some criteria.

5.2.3 The Detection Engine

The detection engine is the most important part of Snort [Snort04]. Its responsibility is to detect if any intrusion activity exists in a packet. The detection engine employs Snort [Snort04] rules for this purpose. The rules are read into internal data structures or chains where they are matched against all packets. Snort [Snort04] organizes parts of packets to make the job of matching rules against them faster. It maintains detection rules in a two dimensional linked list of what are termed Chain

Headers and Chain Options. The commonalities are condensed into a single Chain Header and individual detection signatures are kept in Chain Option structures. If a packet matches any rule, appropriate action is taken; otherwise the packet is dropped. Appropriate actions may be logging the packet or generating alerts.

5.2.4 Logging and Alerting System

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

5.2.5 Output Modules

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

- The database module is used to store Snort [Snort04] output data in databases, such as MySQL, MSSQL or Oracle,
- The SNMP module can be used to send Snort [Snort04] alerts in the form of traps to a management server,
- The Sending Server Message Block (SMB) alerts module can send alerts to Microsoft Windows machines in the form of pop-up SMB alert windows,
- The syslog module logs messages to the syslog utility (using this module you can log messages to a centralized logging server.),
- XML or CSV modules can be used to save data in XML or comma separated files. The CSV files can then be imported into databases or spreadsheet software for further processing or analysis.

5.2.6 Snort_inline

Snort_inline [SnortInline04] is basically a modified version of Snort [Snort04]. It accepts packets from iptables [IPtables04], instead of libpcap [TCPDump04]. It then uses new rule types to tell iptables if the packet should be dropped or allowed to pass based on the Snort [Snort04] rule set. You can think of Snort_inline [SnortInline04] as a NIPS that uses existing IDS signatures to make decisions on packets that traverse Snort_inline [SnortInline04].

Snort_inline [SnortInline04] is also an open source project like Snort [Snort04]. It is actually Snort itself with an addition, inline operation capability. It is being developed for Unix platforms and no Windows version exists at the time this thesis is being written. When it is operating in inline mode, it accepts packets from iptables [IPtables04], and then forwards them to the packet decoder after converting them to libpcap [TCPDump04] packet format. The packet flows toward the logging and alerting system as it is in the Snort architecture [Snort04]. In the logging and alerting subsystem, the appropriate output module for the matching rule is called. Unlike Snort [Snort04], there are three additional rule types in Snort_inline [SnortInline04]:

- **drop** - The drop rule type will tell iptables [IPtables04] to drop the packet and log it via usual Snort [Snort04] means,
- **reject** - The reject rule type will tell iptables [IPtables04] to drop the packet, log it via usual Snort [Snort04] means, and send a TCP reset if the protocol is TCP or an ICMP port unreachable if the protocol is UDP,
- **sdrop** - The sdrop rule type will tell iptables [IPtables04] to drop the packet. Nothing is logged.

Depending on the matching rule type, the packet is either forwarded using iptables [IPtables04], or dropped. Besides taking one of these actions, logging and/or alerting can also be done.

5.3 A NIPS Implementation

The NIPS appliance of implementation is a modified version of Snort running inline on Windows platform. To achieve this, Snort_inline, the open source, inline NIDS running on Unix platform, has been modified to be run on Windows platform in order to use the already defined and implemented new rule types (drop, sdrop and reject). As stated in the previous section, Snort_inline is a version of Snort with inline operation capability and has been developed to use iptables [IPtables04] when it runs in inline mode. Thus, it accepts packets from iptables [IPtables04], instead of libpcap [TCPDump04], and either forward or drops them depending on the result of the detection process.

Therefore, in order to implement a NIPS – that fits the requirements stated in the previous chapter – by modifying the Snort, an API that can

- receive network packets,
- provide raw data access to the application, and
- send packets to the network

is required. As acknowledged before, WinPcap [WinPcap04] has these capabilities and it is compatible with Snort [Snort04]. These are the reasons beyond the choice of WinPcap to be used in the implementation of this NIPS appliance.

The deployment of the NIPS appliance is shown in Figure 5.3. It has two interfaces and both in promiscuous mode. This allows transparent deployment of the appliance. Since it does not have an IP address, it does not require configuration in the router and hosts. And since it is in promiscuous mode, it receives all packets transmitting on the line. The packets are captured in both interfaces by the WinPcap [WinPcap04] driver and, forwarded to the packet decoder component of the Snort after setting the interface number, which defines where the packet came from. The packet is examined by the preprocessors and checked against the rules by detection engine as it is done in the original Snort as shown in Figure 5.2. After the detection process, it is checked to see whether the packet has matched with a drop, reject or sdrop rule or, one of the preprocessors has detected a malicious activity or not. If the packet has been

identified as acceptable, it is sent through the other interface; otherwise, it is dropped (not forwarded).

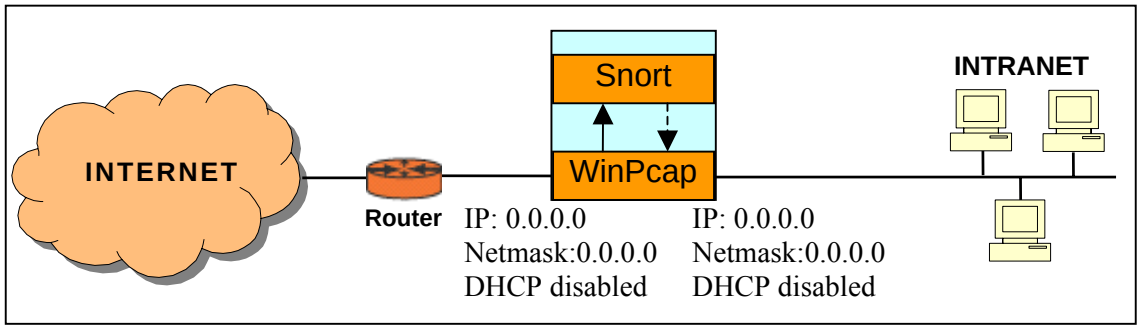

Figure 5.3: Snort/WinPcap deployment

The deployment shown in Figure 5.3 requires the following changes to be made on Snort_inline:

1. Snort_inline has been developed to either capture the traffic in one interface or accept packets from iptables. Thus, Snort_inline is modified to use WinPcap on two interfaces. This is done by defining a second "pcap_descriptor" variable and initializing it with the first one. Also a new command line switch is defined for the assignment of the interfaces when there are more than two interfaces on the device.

2. The packet structure is extended to include a new parameter that is required for two "pcap_descriptors" to work together. This parameter is used to identify the interface that the packet captured from. And it helps to determine the interface that the packet will be forwarded.

3. Code for forwarding of packet based on the result of the detection process is added. Since the appliance has been developed by modifying the Snort_inline [SnortInline04], the drop, reject and sdrop rule types have been already defined but the appliance, currently, only drops the packets for all those rule types. If the packet is flagged to be dropped, it is discarded (not forwarded). If it is not flagged to be dropped, it is send to the network through the interface other than the one it had come from.

After these changes, the NIPS appliance is capable to accept packets from WinPcap [WinPcap04], instead of iptables [IPtables04] and sending them through the

other interface using WinPcap [WinPcap04] unless the detection process expresses otherwise. However, this appliance has the problem of having, also, a copy of the packets that it sends. That is why, a mechanism to discard sent packets in order to avoid the handling of packets again and again.

The problem originates from WinPcap packet capturing capability. WinPcap buffers copies of all packets that are received or "send" from the interface it is configured to work on. To get around this problem, a linked list (PacketQueue) that holds the packets send from an interface is used. Each node of the PacketQueue holds the information that can be used to identify a packet uniquely to eliminate wrong matches. The structure of PacketQueue is given in Table 5.1.

When a packet that is sent from an interface is captured again (capture of an outgoing packet), it will match one of the packets in the PacketQueue. The matching packets are removed from the PacketQueue and the captured packet is dropped at that point to eliminate unnecessary, further processing. This structure has solved the problem of reprocessing of the same packets.

Table 5.1 : Structure of the PacketQueue

```
typedef struct _NODE_DATA{
      int type;
      u_int8_t  ether_dst[6]; /* destination ethernet address */
      u_int8_t  ether_src[6]; /* source ethernet address */
      u_int16_t ar_pro;       /* format of protocol address */
      u_int16_t ip_csum;      /* checksum */
      u_int16_t ip_id;        /* identification */
      u_int8_t  ip_proto;     /* datagram protocol */
      } nDATA;

//-- definition of a node
typedef struct NODE_STRUCT* PNODE;
typedef struct NODE_STRUCT {
      nDATA  p;
      PNODE  pRightLink;
      } NODE;

//-- definition of a list
typedef struct LIST_STRUCT* PLIST;
typedef struct LIST_STRUCT {
      PNODE pFirstNode;
      PNODE pLastNode;
      int   ListError;
      } LIST;
```

The appliance is tested by using two Windows hosts, which are connected to different interfaces of the appliance. Before deploying the appliance, DHCP service is enabled for both hosts and they are connected through a cross-cable. Then the connection between them is checked by “ping” and “telnet” programs. After observing the proper communication between hosts, the NIPS appliance is deployed between these two hosts to examine the communication and apply its one and only rule:

```
drop tcp any any <> any 23 (msg: "Dropping Telnet connection";)
```

This rule should silently drop any connection to port 23 (Telnet) and log the connection attempt with the message “Dropping Telnet connection.”

At the beginning of the test, the connection between two hosts is checked using “ping” program on both hosts (pinging each other). Both of them reached to the other. That is, the appliance is transparently deployed between these hosts. Then, “telnet” program is used to see whether the rules are being applied or not. Both hosts could not make a telnet connection with the other as expected. This shows that the appliance has

successfully decoded the packet, identified it as a telnet attempt and applied its only rule by dropping it. That means the NIPS appliance is working for the given rule.

CHAPTER 6

DISCUSSION AND CONCLUSION

The objective of this thesis is to analyze the various methodologies applied in development of an IPS. To accomplish this objective, first, the stages of an attack are analyzed to understand at which stages an attack can be detected and at which stages can be prevented. Then firewall, IDS, network-based antivirus system and honeypot technologies are examined to determine when, how and why they are used. Also, the limitations of those technologies is defined which are the cause to the new IPS technology. Later, vendors' definitions of IPS technology are gathered and in order to eliminate future confusions it is redefined in a way that would not lead to misunderstanding. Then, different types of existing so-called IPS products are examined to clarify what they can do and can not do. Afterward, the current challenging areas of the NIPS technology are defined and various detection methods applied by IPS vendors to get around these problems are examined. Later on, the requirements of a NIPS is listed and a scalable, three-tiered architecture is proposed that meets those requirements. Lastly, an example NIPS implementation is done to make the difference between the current security technologies and IPS clearer. The NIPS implementation in this thesis may not provide an effective protection for a real network, but it gives guidance to people who want to develop a NIPS.

This thesis includes the various approaches to the already existing problems of the IPS technology. The proposed architecture would get around most of the problems but, there may be other approaches which may be better. Also, the NIPS implementation is not efficient to be used in high performance networks of today. Thus, future work on IPSs is required to perform more detailed analysis on the existing and upcoming problems.

A future work may be to develop appropriate packet filter architecture for the NIPS developed in this thesis which may be different than the WinPcap [WinPcap04] architecture and similar to the packet filter architecture in firewalls, or a mixture of

both. Another future work may be to test the NIPS developed in this thesis with in a real network to find out deficiencies. Thus, new researches to eliminate them can be performed. Also, testing of the NIPS appliance in this thesis with different rule set to determine the effect of rules on performance can be another future work.

Another interesting future work may be the analysis of management systems of NIPSs to define its requirement, component and their functions in details and then develop a system that fits in the requirements. Later, this system and the NIPS implementation developed in this thesis can be combined to form a true NIPS architecture.

SUMMARY

Firewall, IDS, honeypot, and network-based antivirus system have their place in the perimeter security, each with its unique features. Each organization, depending on their business needs, budget constraints, and organizational requirements, needs to draw up a security policy and that policy will determine the mix of components that need to be installed, to meet security goals.

Intrusion prevention is a generic term that defines systems having attack detection and proactive prevention capability built-in. Proactive capabilities of IPS will help to keep networks safer from sophisticated, fast-spreading attacks. IPS is the evolved and integrated state of the current technologies. Its integrated structure is believed to be heart of the next generation network security systems.

In this thesis, various methods applied on different components of the NIPS have been analyzed. Also, the requirements of NIPS are defined and a NIPS architecture is proposed. Finally, an implementation of NIPS Sensor is defined using the existing open source applications. With these contents, this thesis defines the current state and new trends in network security, and provides guidance for future researches.

REFERENCES

[AM03] Ashley, Mitchell. “Layered Network Security: A best-practices approach”. StillSecure. January 2003

[AS99] Axelsson, S. “The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection”. In Proceedings of the 6th ACM Conference on Computer and Communications Security. pages 1-7. November 2-4, 1999

[BM01] Bace, Rebecca and Mell, Peter. “Intrusion Detection Systems”. NIST Special Publication. August 2001

[BR00] Bace, R. “Intrusion Detection”. Macmillan Technical Publishing. 2000

[Cisco04] Cisco Systems, Inc. January 2004. [http://www.cisco.com/]

[CM02] Cuppens, F., Miege, A. “Alert Correlation In A Cooperative Intrusion Detection Framework”. Proceedings of the IEEE Symposium on Security and Privacy. Oakland, May 2002.

[CP00] Check Point Software Technologies Ltd. “Stateful Firewall Technology - Products and Solutions”. 2000. [http://www.checkpoint.com/products/technology]

[DN03] Desai, Neil. “Intrusion Prevention Systems: the Next Step in the Evolution of IDS”. Feburary 2003. [http://www.securityfocus.com/infocus/1670]

[DT02a] Ted Doty, Director of Product Management, OKENA. “New Approach To Intrusion Detection: Intrusion Prevention”. Date published in TECS: 23 January, 2002. [http://www.itsecurity.com/papers/doty1.htm]

[DT02b] Doty, Ted. Director of Product Management, OKENA. “Technology Best Practices for Intrusion Prevention”. Date posted in ITsecurity.com: 9 July, 2002. [http://www.itsecurity.com/papers/doty2.htm]

[EC94] Espasa Calpe. “Dictionary of the Spanish tongue”. 1994

[EC99] Entera Corporation. “How to cash in on caching”. June 1999 [http://www.entera.com/]

[eEye03] eEye Digital Security Whitepaper. “The Art of Protecting Your Network From Unknown Vulnerabilities”. May 2003. [http://www.eeye.com]

[ForeScout02] ForeScout Technologies. “Beyond Detection: Neutralizing Attacks Before They Reach the Firewall”. Summer 2002

[ForeScout04] ForeScout Technologies, Inc. January 2004. [http://www.forescout.com/]

[Fortinet02] Fortinet Inc. “Improving Network Protection and Performance with Network-Based Antivirus Technology”. October 2002

[GDG03] Gómez, Diego González. “Sistemas de Detección de Intrusiones: Capítulo 4”. July 2003. [http://www.dggomez.arrakis.es/secinf/ids/html/cap01.htm]

[GF02] Gong, Fengmin. Chief Scientist, McAfee Network Security Technologies Group. “Next generation intrusion detection systems (IDS)”. Network Associates 2003: 9-14. March 2002

[GMAL03] Gustavo Martins Arruda, Luiz. “Around network intrusion prevention systems”. SANS Institute 2003: 7-8. August 2003

[Honeypots03] Honeypots.net. “Honeypots, Honeynets”. Page last modified on 21 November 2003. [http://www.honeypots.org/]

[II02] Intoto Inc. “Inline intrusion protection white paper”. March 2002. [http://intotoinc.com]

[IPtables04] IPtables.org. [http://www.iptables.org/]. January 2004

[JK01] Johansson, Karsten. “Offensive Operations Model”. KSAJ Inc. August 2001. [http://www.penetrationtest.com/]

[Jupitermedia02] Jupitermedia. “Intrusion Detection System”. Last modified: December 13, 2002. [http://www.webopedia.com/tem/I/intrusion_detection_system.html]

[LTGJ92] Lunt, Teresa F., Tamaru A., Gilham F., Jagannathan R., Jalali C., Neumann P., Javitz H., Valdes A., Garvey T. “A real-time intrusion-detection expert system (IDES)”. SRI International. SRI Project 6784: 98-102. February 1992

[MCA03] Murray-Conry, Andrew. TechWeb. “Emerging Technology: Detection vs. Prevention - Evolution or Revolution?”. May 05, 2003. [http://www.networkmagazine.com/shared/article/showArticle.jhtml?articleId=9400017]

[MH02] Mousa, Hussam O. “A Survey and Analysis of Neural Network Approaches to Intrusion Detection”. November 12, 2002

[NA03] Network Associates, “Intrusion Prevention: Myths, Challenges, and Requirements”, April 2003. [http://www.networkassociates.com/]

[NetScreen02] NetScreen Technologies Inc. “Intrusion Detection and Prevention Protecting Your Network From Attacks”. October 2002. [http://www.netscreen.com/]

[NetScreen04] NetScreen Technologies Inc. January 2004. [http://www.netscreen.com/]

[NIST02] National Institute of Standards and Technology (NIST). “Guidelines on Firewalls and Firewall Policy”. January 2002

[OLMS97] Oliver J., Leahy Dermot M., Tynan J., Mark Smith, Sean G. Doherty, “Firewall technology”, Digital Technical Journal, (2), 1997

[PD03] Poynter, Ian. Doctor, Brad. “Beyond The Firewall: The next level of network security”. StillSecure. January 2003

[PJP03] Planquart, Jean-Philippe. “Application of Neural Networks to Intrusion Detection”. February 2003

[PPC97] Pfleeger, P. Charles. “Security in Computing”. Prentice Hall PTR. Second Edition. p 3. 1997

[RB03] Robinson, Brian. “Intrusion prevention-How it works”. Federal Computer Week. March, 2003. [http://www.fcw.com/supplements/homeland/2003/sup1/hom-prevent1-03-10-03.asp]

[RM03] Michael Reed. “Technology update: Intrusion-prevention systems”. VP Business Development. Top Layer Networks. July 2003. [http://searchsecurity.techtarget.com/tip/1,289483,sid14_gci913275,00.html]

[RP01] Ryan, Permeh. “The use of application specific security measures in a modern computing environment”. March 2001. [http://www.eeye.com/html/Research/Papers/DS20010322.html]

[RR03] Rehman, Rafeeq Ur. “Intrusion Detection Systems with Snort”. Prentice Hall PTR. ISBN 0-13-140733-3. 2003

[SL03] Spitzner, Lance. “Honeypots: Definitions and Value of Honeypots”. Last Modified: 29 May, 2003. [http://www.tracking-hackers.com/papers/honeypots.html]

[Snort04] Snort.org. January 2004. [http://www.snort.org/]

[SnortInline04] Snort Inline. January 2004. [http://snort-inline.sourceforge.net/]

[SW00] Shay, William A. “Firewall”. University of Wisconsin-Green Bay. 2000

[TCPDump04] Tcpdump.org. January 2004. [http://www.tcpdump.org/]

[TechTarget03] TechTarget. “Honeypot”. Last updated on: September 24, 2003. [http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci551721,00.html]

[TopLayer02] Top Layer. “Beyond IDS: Essentials of Network Intrusion Prevention”. November 2002

[UY02] Ultes-Nitsche, Ulrich and Yoo, InSeon. University of Southampton, United Kingdom. “An Integrated Network Security Approach”. July 2002

[WinPcap04] NetGroup, Politecnico di Torino. January 2004. [http://winpcap.polito.it/docs/]

[YY00] Yakomba Yavwa. “The Firewall Technology”. May 2, 2000