DESIGN OF LOW INTERACTION HONEYPOT TO TRACK INTRUSION

Submitted in partial fulfilment
of the requirements for the degree of

BACHELOR OF TECHNOLOGY
(CSE & / IT)
By

Sujay Kumar Srivastava 071216

Ramit Kumar 071242

Sarvendra Pratap Singh 071206

Aditya Tanwar 071226

Under the Supervision of
Mr. Amol Vasudeva

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
juit
विद्या तत्व ज्योतिसम:

MAY 2011

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY- WAKNAGHAT

CONTENTS

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled **"DESIGN OF LOW INTERACTION HONEYPOT TO TRACK INTRUSION"** submitted in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Name of Supervisor: [illegible]

ACKNOWLEDGEMENT

Computer Science is fast developing subject & new dimentions are being added from time to time. During the last decade many new dimentions have been introduced and many old once have been redefined. In light of new development and recent findings, we devote the task that was asked from us during our IV year at Jaypee University Of Information Technology to DESIGN LOW INTERACTION HONEYPOT TO TRACK INTRUSION.

We would like to thank **Mr. Amol Vasudeva** who helped us during the task given to us. We thank him again for providing us with the necessary facilities to complete this task.

Sujay Kumar Srivastava	Roll No. 071216	[illegible]
Ramit Kumar	Roll No. 071242	Ramit K.
Sarvendra Pratap Singh	Roll No. 071206	[illegible]
Aditya Tanwar	Roll No. 071226	[illegible]

B.Tech(Computer Science & Engineering)

Jaypee University Of Information Technology

ABSTRACT

An Interactive Port Monitor model of Honeypot has been proposed to mitigate the effects of DDoS attacks. A network providing services through a port number will be updated and the port will be switched to some other port number to provide the same services. The organization members know that this upgrading has been done therefore no services shall be provided by the previous port. The honeypot will be deployed at the older port. Hence, any traffic coming to this port will certainly be an attack or scan by some outside the organization who still does not know about the organization up gradation and wants to break into main server thinking older part still to be the server port.

CHAPTER 1: INTRODUCTION AND PROBLEM STATEMENT

1.1 Introduction

Network security has evolved from independently deployed products such as firewalls into the realm of system-wide solutions. The reason is simple: For today's companies, especially in this era of regulatory activity, preserving the integrity, confidentiality of corporate information is critical to success. As we move further into an information-driven global economy, the value of information, and controlled access to that information, has never been greater. The goal of IT infrastructure therefore is to create systems that can detect and protect against unauthorized access while providing timely access to legitimate users. Simply denying access in the face of an attack is no longer acceptable. Today's networks must be able to respond to attacks in ways that maintain network availability and reliability and allow a business to continue to function. In many respects, the goal of security is to make networks more resilient by making them more flexible. Rather than succumb, networks must be able to absorb attacks and remain operational, much in the same way the human immune system allows us to keep functioning in the presence of viruses and related bacterial infections.

The Internet was created in 1969 to provide an open network for researchers' .In the last decade, the phenomenal growth and the success of the Internet is changing its traditional role. Unfortunately, with the growth of the Internet, the attacks to the Internet have also increased incredibly fast. The occurrence of the Morris Worm in 1988 marked the first major computer security incident on the Internet.

The widespread need and ability to connect machines across the Internet has caused the network to be more vulnerable to intrusions and has facilitated break-ins of a variety of types. Along their paths between a client and a server, network packets consume various kinds of resources including access link bandwidth, router buffers etc. Distributed-Denial-of-Service (DDoS) attacks inject maliciously-designed packets into the network to deplete some or all these sources.

Whereas service front-ends can be protected from DDoS attacks by massive replication, back-ends cannot tolerate the same level of replication, because of higher costs and tighter consistency constraints. DDoS attacks are difficult to prevent because of inevitable software vulnerabilities, which get exploited by attackers and are used to launch the attack.

There are four different ways to defend against DDoS attacks:

(1) Attack prevention aims to fix security holes, such as insecure protocols,

(2) Attack detection aims to detect DDoS in the process of an attack,

(3) Attack source identification aims to locate the attack sources and

(4) Attack reaction aims to eliminate or curtail the effects of an attack.

Countermeasures for the DDoS attacks are getting more sophisticated and widely used firewalls pop up in the nearly every company. Network intrusion detection systems start their triumphal mechanism. But do we know enough of our enemy? Or are these systems always a step behind the black hats? Gathering this kind of information about enemy is not easy but important .By knowing attack strategies, countermeasures can be taken and the vulnerabilities can be fixed. To gather as much information as possible as possible is one main goal of a Honeypot. Generally, such information gathering should be done silently, without alarming an attacker. All the gathered information leads to an advantage on the defending side and can therefore be used on the productive systems to prevent attacks.

Honeypot is an important security technology used to understand and combat DDoS attacks. It gets is name from the age old saying "you can catch more flies with honey rather than vinegar". It is primarily an instrument for information gathering and learning.

A Honeypot is set up on a network for the sole purpose of being attacked. It is designed with deliberate vulnerabilities, which is exposed to a public network. The goal is first to lure intruders away from the real system, and secondly to closely monitor the intruder to study the exploits which are used. Honeypots are not supposed to receive any legitimate traffic and thus, any traffic destined to a Honeypot is most probably an ongoing attack and can be analyzed to reveal vulnerabilities targeted by attackers' .Coupled with an Intrusion Detection System (IDS), Honeypots are effective in detecting victim hosts.

Honeypot systems can be implemented as low-interaction and high-interaction systems. The low interaction Honeypot involves a simple emulation of an operating system and network services at a very superficial level. A high-interaction Honeypot is implemented as a real operating system and with services running on real machine. This allows for data to be logged about the attacker's actions once they are on the machine, which may reveal much more about their intentions.

Honeypots can be used for two different reasons, research and production. Researchers can track the movements through these Honeypots to get a better understanding of their methodology so more advanced forms of defenses can be developed in the future. And the companies use honeypots in the defense of the intricate network systems.

1.2 Key Issues and Challenges

Honeypots have proven to be extremely successful and provide zero "false positive" results. Three main purposes that revolve around honeypots today are detection, prevention, and information gathering. Potential applications of honeypots include commercial enterprises, government agencies, surveillance and communication in battlefield and din production and research. Honeypots are useful tool, however there are some limitations associated with their use.

One problem is the issue of liability. Intentionally placing n insecure machine on a network and allowing it to get compromised can put an organization into a very awkward position if the attacker on the other organizations' or individuals' systems. Therefore, limitations have to be placed on honeypots' out-bound network connections in order to prevent abuse at the hands of attackers. Honeypots are not the solution for the computer crimes. Honeypots are hard to maintain and they need operators with deep knowledge about computer and network security.

1.3 Statement of the Problem

Design and Implementation of a low interaction HONEYPOT to track Distributed Denial of Services (DDoS) attack.

The project consists of two stages:

1. Deployment of DDoS attacks: DDoS attacks shall be deployed to a target machine either through a pre-constructed tool or would be self constructed.
2. Construction of a low interaction Honeypot to detect these DDoS attacks. The proposed Honeypot will gain the basic information about the source of attack. This would be done by gaining the information of the incoming packets from the source of attack preferably on a specific port.

CHAPTER 2: BACKGROUND STUDY

2.1 Classification of attacks

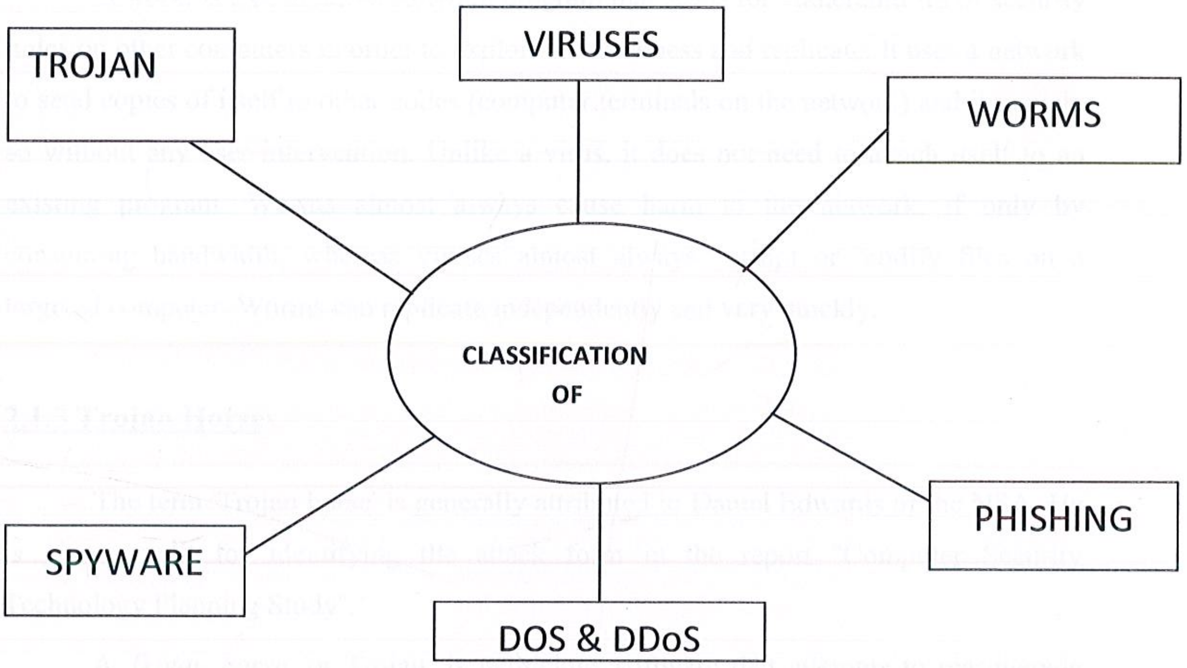


Fig. 1: Classification of Attacks

2.1.1 Virus

The term virus is credited to University of Southern California professor *Frederick Cohen* in his 1984 research paper Computer Viruses: *Theory and Experiments. A computer virus is designed to attack a computer and often to wreak havoc on other computers and network devices.* A virus can often be an attachment in an e-mail, and selecting the attachment can cause the executable code to run and replicate the virus. A virus must be executed or run in memory in order to run and search for other programs or hosts to infect and replicate. As the name implies, a virus needs a host such as a spreadsheet or e-mail in order to attach, infect, and replicate.

2.1.2 Worms

A *worm* is a destructive software program that scans for vulnerabilities or security holes on other computers in order to exploit the weakness and replicate. It uses a network to send copies of itself to other nodes (computer terminals on the network) and it may do so without any user intervention. Unlike a virus, it does not need to attach itself to an existing program. Worms almost always cause harm to the network, if only by consuming bandwidth, whereas viruses almost always corrupt or modify files on a targeted computer. Worms can replicate independently and very quickly.

2.1.3 Trojan Horse

The term 'Trojan horse' is generally attributed to Daniel Edwards of the NSA. He is given credit for identifying the attack form in the report "Computer Security Technology Planning Study".

A *Trojan horse*, or Trojan, is pernicious software that attempts to masquerade itself as a trusted application such as a game or screen saver. Once the unsuspecting user attempts to access what appears to be an innocuous game or screen saver, the Trojan can initiate damaging activities such as deleting files or reformatting a hard drive.

2.1.4 Spyware

Spyware is a class of software applications that can participate in a network attack. Spyware is an application that attempts to install and remain hidden on a target PC or laptop. Once the spyware application has been surreptitiously installed, the spyware captures information about what users are doing with their computers. Some of this captured information includes websites visited, e-mails sent, and passwords used.

Attackers can use the captured passwords and information to gain entry to a network to launch a network attack.

In addition to being used to directly participate in a network attack, Spyware can also be used to gather information that can be sold underground. This information, once purchased, can be used by another attacker that is "harvesting data" to be used in planning another network attack.

2.1.5 Phishing

Phishing is a type of network attack that typically starts by sending an e-mail to an unsuspecting user. The Phishing e-mail attempts to look like a legitimate e-mail from a known and trusted institution such as a bank or ecommerce site. This false e-mail attempts to convince users that something has happened, such as suspicious activity on their account, and that the user must follow the link in the e-mail and logon to the site to view their user information. The link in this e-mail is often a false copy of the real bank or ecommerce site and features a similar look-and-feel to the real site. The Phishing attack is designed to trick users into providing valuable information such as their username and password.

2.2 DoS and DDoS Attacks

The goal of a DoS attack is to disrupt some legitimate activity, such as browsing Web pages, listening to an online radio, transferring money from your bank account, or even docking ships communicating with a naval port. This denial-of-service effect is achieved by sending messages to the target that interfere with its operation, and make it hang, crash, reboot, or do useless work.

DoS and DDoS:

One way to interfere with a legitimate operation is to exploit vulnerability present on the target machine or inside the target application. The attacker sends a few messages crafted in a specific manner that take advantage of the given vulnerability. Another way is to send a vast number of messages that consume some key resource at the target such as bandwidth, CPU time, memory, etc. The target application, machine, or network spends all of its critical resources on handling the attack traffic and cannot attend to its legitimate clients.

Of course, to generate such a vast number of messages the attacker must control a very powerful machine—with a sufficiently fast processor and a lot of available network bandwidth. For the attack to be successful, it has to overload the target's resources. This means that an attacker's machine must be able to generate more traffic than a target, or its network infrastructure, can handle.

Now let us assume that an attacker would like to launch a DoS attack on example.com by bombarding it with numerous messages. Also assuming that example.com has abundant resources, it is then difficult for the attacker to generate a sufficient number of messages from a single machine to overload those resources. However, suppose he gains control over 100,000 machines and engages them in generating messages to example.com simultaneously. Each of the attacking machines now may be only moderately provisioned (e.g., have a slow processor and be on a modem link) but together they form a formidable attack network and, with proper use, will be able to overload a well-provisioned victim. This is a distributed denial-of-service—DDoS.

Both DoS and DDoS are a huge threat to the operation of Internet sites, but the DDoS problem is more complex and harder to solve. First, it uses a very large number of machines. This yields a powerful weapon. Any target, regardless of how well provisioned it is, can be taken offline. Gathering and engaging a large army of machines has become trivially simple, because many automated tools for DDoS can be found on hacker Web pages and in chat rooms. Such tools do not require sophistication to be used and can inflict very effective damage. A large number of machines give another advantage to an attacker. Even if the target were able to identify attacking machines (and there are effective ways of hiding this information), what action can be taken against a network of 100,000 hosts? The second characteristic of some DDoS attacks that increases their complexity is the use of seemingly legitimate traffic. Resources are consumed by a large number of legitimate-looking messages; when comparing the attack message with a legitimate one, there are frequently no telltale features to distinguish them. Since the attack misuses a legitimate activity, it is extremely hard to respond to the attack without also disturbing this legitimate activity.

A denial-of-service attack is different in goal, form, and effect than most of the attacks that are launched at networks and computers. Most attackers involved in cyber crime seek to break into a system, extract its secrets, or fool it into providing a service that they should not be allowed to use.

In DDoS attacks, breaking into a large number of computers and gaining malicious control of them is just the first step. The attacker then moves on to the DoS attack itself, which has a different goal—to prevent victim machines or networks from offering service to their legitimate users. No data is stolen, nothing is altered on the victim machines, and no unauthorized access occurs. The victim simply stops offering service to normal clients because it is preoccupied with handling the attack traffic. While no unauthorized access to the victim of the DDoS flood occurs, a large number of other hosts have previously been compromised and controlled by the attacker, who uses them as attack weapons. In most cases, this is unauthorized access, by the legal definition of that term.

2.2.1 Damaging Effects of DoS and DDoS

While the denial-of-service effect on the victim may sound relatively benign, especially when one considers that it usually lasts only as long as the attack is active, for many network users it can be devastating. Use of Internet services has become an important part of our daily lives. The Internet is increasingly being used to conduct business and even to provide some critical services. Following are some examples of the damaging effects of DoS attacks.

- Sites that offer services to users through online orders make money only when users can access those services. For example, a large book-selling site cannot sell books to its customers if they cannot browse the site's Web pages and order products online. A DoS attack on such sites means a severe loss of revenue for as long as the attack lasts. Prolonged or frequent attacks also inflict long-lasting damage to a site's reputation—customers who were unable to access the desired service are likely to take their business to the competition. Sites whose reputations were damaged may have trouble attracting new customers or investor funding in the future.
- Large news sites and search engines are paid by marketers to present their advertisements to the public. The revenue depends on the number of users that view the site's Web page. A DoS attack on such a site means a direct loss of revenue from the marketers, and may have the long-lasting effect of driving the customers to more easily accessible sites. Loss of popularity translates to a direct loss of advertisers' business.
- Some sites offer a critical free service to Internet users. For example, the Internet's Domain Name System (DNS) provides the necessary information to translate human-readable Web addresses (such as www.example.com) into Internet Protocol (IP) addresses (such as 192.0.34.166). All Web browsers and numerous other applications depend on DNS to be able to fetch information requested by the users. If DNS servers are under a DoS attack and cannot respond due to overload, many sites may become unreachable because their addresses cannot be resolved,

even though those sites are online and fully capable of handling traffic. This makes DNS a part of the critical infrastructure, and other equally important pieces of the Internet's infrastructure are also vulnerable.

- Numerous businesses have come to depend on the Internet for critical daily activities. A DoS attack may interrupt an important videoconference meeting or a large customer order. It may prevent a company from sending out an important document for a rapidly approaching deadline or interfere with its bid for a large contract.
- The Internet is increasingly being used to facilitate management of public services, such as water, power, and sewage, and to deliver critical information for important activities, such as weather and traffic reports for docking ships. A DoS attack that disrupts these critical services will directly affect even people whose activities are not related to computers or the Internet. It may even endanger human lives.

2.2.2 How do DoS and DDoS attack works?

There are two main approaches to denying a service: exploiting vulnerability present on the target or sending a vast number of seemingly legitimate messages. The first kind of an attack is usually called a vulnerability attack, while the second is called a flooding attack.

1.) Vulnerability attacks

Vulnerability attacks work by sending a few specifically crafted messages to the target application that possesses vulnerability. This vulnerability is usually a software bug in the implementation or a bug in a default configuration of a given service. Malicious messages by the attacker represent an unexpected input that the application programmer did not foresee. The messages cause the target application to go into an infinite loop; to severely slow down, crash, freeze, or reboot a machine; or to consume a vast amount of memory and deny service to legitimate users. This process is called exploiting vulnerability, and the malicious messages are called the *exploit*. In some cases, vulnerabilities of this kind can be exploited in the operating system, a common piece of middleware, or in a network protocol, as well as in application programs. While it is impossible to detect all vulnerabilities, it can also be quite hard to find new exploits. This means that each vulnerability that is detected and patched is a large gain and a sure step ahead for the defenders.

2.) Flooding attack

Flooding attacks work by sending a vast number of messages whose processing consumes some key resource at the target. For instance, complex messages may require lengthy processing that takes up CPU cycles, large messages take up bandwidth, and messages that initiate communication with new clients take up memory. Once the key resource is tied up by the attack, legitimate users cannot receive service. The crucial feature of flooding attacks is that their strength lies in the volume, rather than in content. This has two major implications:

a. The attackers can send a variety of packets. The attack traffic can be made arbitrarily similar to the legitimate traffic, which greatly hinders defense.
b. The flow of traffic must be so large as to consume the target's resources. The attacker usually has to engage more than one machine to send out the attack traffic. Flooding attacks are therefore commonly DDoS attacks.

The fact that the line between vulnerability and flooding attacks is thin and many attacks may well falls into both the vulnerability and flooding categories.

Recruiting and Controlling Attacking Machines:

DDoS attacks require engagement of multiple machines, which will be sending the attack traffic to the victim. Those machines do not belong to the attacker. They are usually poorly secured systems at universities, companies, and homes—even at government institutions. The attacker breaks into them, takes full control, and misuses them for the attack. Therefore, the attacking machines are frequently called zombies, daemons, slaves, or **Agents**.

The **Agents** are usually poorly secured machines—they do not have recent patches and software updates, they are not protected by a firewall or other security devices, or their users have easily guessed passwords. The attacker takes advantage of these well-known holes to break in. Unpatched and old software has well-known vulnerabilities with already-written exploits. These belong to a specific kind of vulnerabilities—once exploited, they allow the attacker unlimited access to the system, as if he had an administrator's account. Accounts with easily guessed passwords, such as combinations of users' names or dictionary words, allow another easy way into the machine.

Once the attacker has gained control of the host, she installs the DDoS attack agent and makes sure that all traces of the intrusion are well hidden and that the code runs even after the machine is rebooted.

DDoS attacks frequently involve hundreds or thousands of agents. It would be tedious and time consuming if the attacker had to manually break into each of them. Instead, there are automated tools that discover potential agent machines, break into them, and install the attack code upon a single command from an attacker, and report success back to her.

The attacker further hides her identity by deploying several layers of indirection between her machine and the agents. She uses one or several machines that deliver her commands to the agents. These machines are called **Handlers** or **Masters**. Figure below illustrates the **Handler/Agent** architecture:

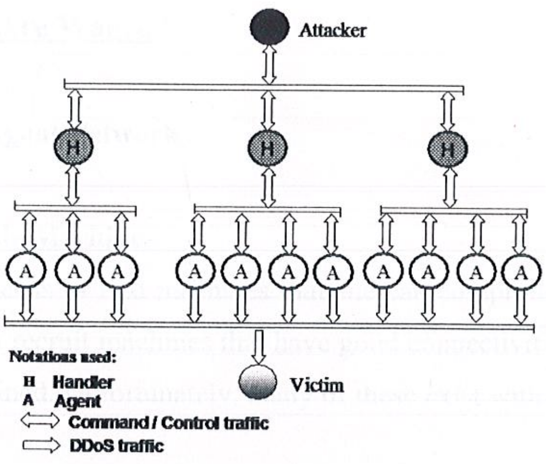
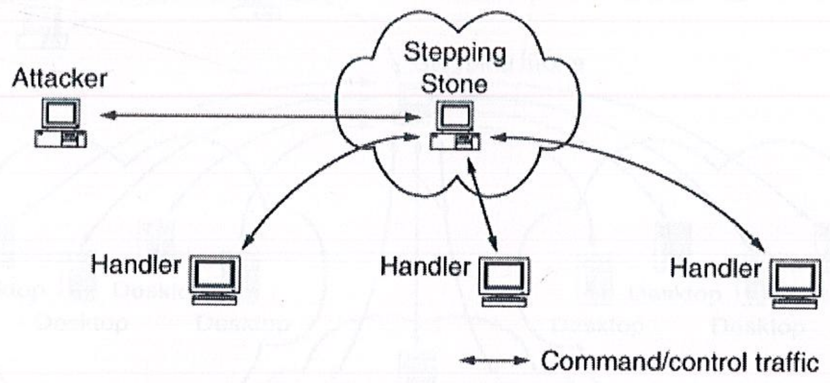


Fig. 3 DDoS attack

Another layer of indirection consists of the attacker's logging on to several machines in sequence, before accessing the handlers. These intermediary machines between the attacker's machine and the handlers are called the *stepping stones*, and are illustrated in figure below:



Both handlers and stepping stones are used to hinder investigation attempts. If authorities located and examined an agent machine, all its communication would point to one of the handlers. Further examination of the handler would point to a stepping stone and from there to another stepping stone. If stepping stones are selected from different countries and continents (and they usually are), it becomes very difficult to follow the trail back to the attacker's machine and unveil her identity.

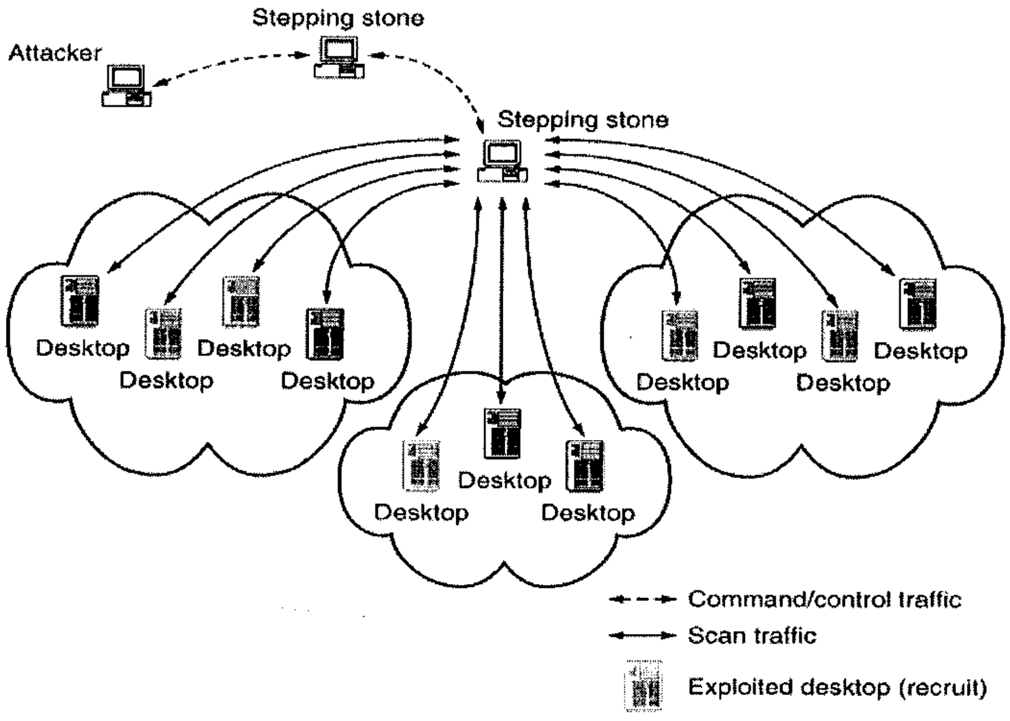
2.2.3 How Attacks Are Waged?

Recruitment of the Agent Network

a.) Finding Vulnerable Machines

The attacker needs to find machines that she can compromise. To maximize the yield, she will want to recruit machines that have good connectivity and ample resources and are poorly maintained. Unfortunately, many of these exist within the pool of millions of Internet hosts.

The process of looking for vulnerable machines is called scanning. Figure below depicts the simple scanning process. The attacker sends a few packets to the chosen target to see whether it is alive and vulnerable. If so, the attacker will attempt to break into the machine.



Tools that can be used for scanning are **blended threats** and **worms**.

Blended threats are individual program or group of programs that provide many services, in this case command and control using IRC bot and vulnerability scanning.

A **bot** (derived from "robot") is a client program that runs in the background on a compromised host, and watches for certain strings to show up in an IRC channel. These strings represent encoded commands that the bot program executes, such as inviting someone into an IRC channel, giving the user channel operator permissions, scanning a block of addresses (netblock), or performing a DoS attack. Netblock scans are initiated in certain bots, such as Power, by specifying the first few octets of the network address (e.g., 192.168 may mean to scan everything from 192.168.0.0 to 192.168.255.255). Once bots get a list of vulnerable hosts, they inform the attacker using the botnet (a network of bots that all synchronize through communication in an IRC channel). The attacker retrieves the file and adds it to her list of vulnerable hosts. Some programs automatically add these vulnerable hosts to the vulnerable host list, thereby constantly reconstituting the attack network. Network blocks for scanning are sometimes chosen randomly by attackers. *Worms* choose the addresses to scan using several methods.

- ***Completely randomly*** - Randomly choose all 32 bits of the IP address (if using IPv4) for targets, effectively scanning the entire Internet indiscriminately.
- ***Within a randomly selected address range*** - Randomly choose only the first 8 or 16 bits of the IP address, then iterate from .0.0 through .255.255 in that address range. This tends to scan single networks, or groups of networks, at a time.
- ***Using a hitlist*** - Take a small list of network blocks that are "target rich" and preferentially scan them, while ignoring any address range that appears to be empty or highly secured. This speeds things up tremendously, as well as minimizing time wasted scanning large unused address ranges.
- ***Using information found on the infected machine*** - Upon infecting a machine, the worm examines the machine's log files that detail communication activity, looking for addresses to scan. For instance, a Web browser log contains addresses

of recently visited Web sites, and a file known hosts contains addresses of destinations contacted through the SSH (Secure Shell) protocol.

b.) Breaking into Vulnerable Machines

The attacker needs to exploit vulnerability in the machines that she is intending to recruit in order to gain access to them. You will find this referred to as "owning" the machine. The vast majority of vulnerabilities provide an attacker with administrative access to the system, and she can add/delete/change files or system settings at will.

Exploits typically follow a vulnerability exploitation cycle.

1. A new vulnerability has been discovered in attacker circles and is being exploited in a limited fashion.
2. The vulnerability makes it outside of this circle and gets exploited at a wider scale.
3. Automated tools appear, and non-experts (script kiddies) are running the tools.
4. A patch for the vulnerability appears and gets applied.
5. Exploits for a given vulnerability decline

Once one or more vulnerabilities have been identified, the attacker incorporates the exploits for those vulnerabilities into his DDoS toolkit. Some DDoS tools actually take advantage of several vulnerabilities to propagate their code to as many machines as possible. These are often referred to as propagation vectors.

c.) Malware Propagation Methods:

The attacker needs to decide on a propagation model for installing his malware. A simple model is the **central repository**, or cache, approach: The attacker places the malware in a file repository (e.g., an FTP server) or a Web site, and each compromised host downloads the code from this repository. One advantage of the caching model for the defender is that such central repositories can be easily identified and removed. Attackers installing trinoo [Ditf] and Shaft [DLD00] agents used such centralized approaches in the early days.

Another model is the **back-chaining**, or pull, approach, wherein the attacker carries his tools from an initially compromised host to subsequent machines that this host compromises.

Finally, the autonomous, push, or forward propagation approach combines propagation and exploit into one process. The difference between this approach and back chaining is that the exploit itself contains the malware to be propagated to the new site, rather than performing a copy of that malware after compromising the site. The worm carries a DDoS tool as a payload, and plants it on each infected machine. Recent worms have incorporated exploit and attack code, protected by a weak encryption using linear feedback shift registers. The encryption is used to defeat the detection of well-known exploit code sequences by antivirus or personal firewall software. Once on the machine, the code self-decrypts and resumes its propagation.

2.2.4 Semantic Levels of DDoS Attacks

There are several methods of causing a denial of service. Creating a DoS effect is all about breaking things or making them fail. There are many ways to make something fail, and often multiple vulnerabilities will exist in a system and an attacker will try to exploit (or target) several of them until she gets the desired result: The target goes offline.

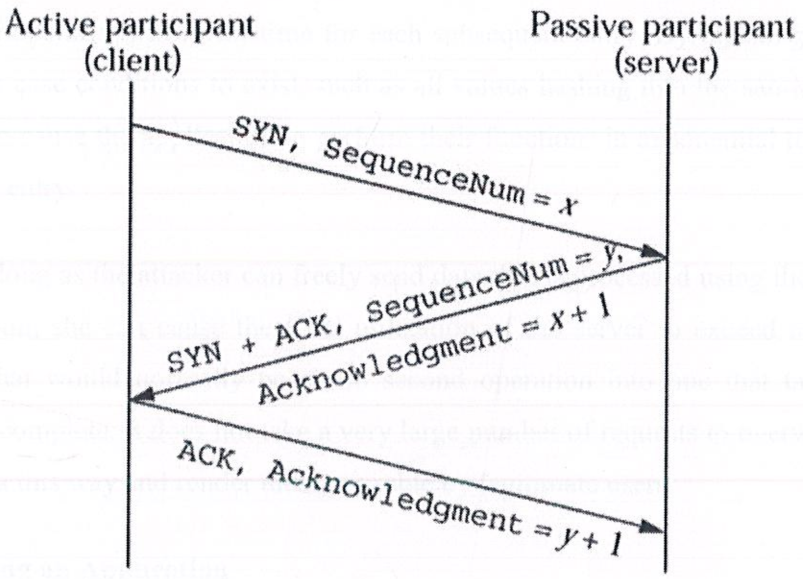
a.) Exploiting a Vulnerability

Vulnerability attacks involve sending a few well-crafted packets that take advantage of an existing vulnerability in the target machine. For example, there is a bug in Windows 95 and NT, and some Linux kernels, in handling improperly fragmented packets. Generally, when a packet is too large for a given network, it is divided into two (or more) smaller packets, and each of them is marked as fragmented. The mark indicates the order of the first and the last byte in the packet, with regard to the original. At the receiver, chunks are reassembled into the original packet before processing. The fragment marks must fit properly to facilitate reassembly. The vulnerability in the above kernels causes the machine to become unstable when improperly fragmented packets are received, causing it to hang, crash, or reboot. This vulnerability can be exploited by sending two malformed UDP packets to the victim. There were several variations of this exploit—fragments that indicate a small overlap, a negative offset that overlaps the second packet before the start of the header in the first packet, and so on. These were known as bonk, boink, teardrop, and new tear exploits.

b.) Attacking a Protocol

An ideal example of protocol attacks is a TCP SYN flood attack. We first explain this attack and then indicate general features of protocol attacks.

TCP session starts with negotiation of session parameters between a client and a server. The client sends a TCP SYN packet to the server, requesting some service. In the SYN packet header, the client provides his initial sequence number, a unique per-connection number that will be used to keep count of data sent to the server (so the server can recognize and handle missing, disordered, or repeated data). Upon SYN packet receipt, the server allocates a transmission control block (TCB), storing information about the client. It then replies with a SYN-ACK, informing the client that its service request will be granted, acknowledging the client's sequence number and sending information about the server's initial sequence number. The client, upon receipt of the SYN-ACK packet, allocates a transmission control block. The client then replies with an ACK to the server, which completes the opening of the connection. This message exchange is called a three-way handshake and is depicted in figure below:


Active participant (client)
Passive participant (server)
SYN, SequenceNum = x
SYN + ACK, SequenceNum = y, Acknowledgment = x + 1
ACK, Acknowledgment = y + 1

The potential for abuse lies in the early allocation of the server's resources. When the server allocates his TCB and replies with a SYN-ACK, the connection is said to be half-open. The server's allocated resources will be tied up until the client sends an ACK packet, closes the connection (by sending an RST packet) or until a timeout expires and the server closes the connection, releasing the buffer space. During a TCP SYN flooding attack, the attacker generates a multitude of half-open connections by using IP source spoofing. These requests quickly exhaust the server's TCB memory, and the server can accept no more incoming connection requests. Established TCP connections usually experience no degradation in service, though, as they naturally complete and are closed, the TCB records spaces they were using will be exhausted by the attack, not replaced by other legitimate connections. In rare cases, the server machine crashes, exhausts its memory, or is otherwise rendered inoperative. In order to keep buffer space occupied for the desired time, the attacker needs to generate steady stream of SYN packets toward the victim (to reserve again those resources that have been freed by timeouts or completion of normal TCP sessions).

c.) Attacking Middleware

Attacks can be made on algorithms, such as hash functions that would normally perform its operations in linear time for each subsequent entry. By injecting values that force worst-case conditions to exist, such as all values hashing into the same bucket, the attacker can cause the application to perform their functions in exponential time for each subsequent entry.

As long as the attacker can freely send data that is processed using the vulnerable hash function, she can cause the CPU utilization of the server to exceed capacity and degrade what would normally be a sub second operation into one that takes several minutes to complete. It does not take a very large number of requests to overwhelm some applications this way and render them unusable by legitimate users.

d.) Attacking an Application

The attacker may target a specific application and send packets to reach the limit of service requests this application can handle. For example, Web servers take a certain amount of time to serve normal Web page requests, and thus there will exist some finite number of maximum requests per second that they can sustain. If we assume that the Web server can process 1,000 requests per second to retrieve the files that make up a company's home page, then at most 1,000 customers' requests can be processed concurrently. For the sake of argument, let's say the normal load a Web server sees daily is 100 requests per second (one tenth of capacity).

e.) Pure Flooding

Given a sufficiently large number of agents, it is possible to simply send any type of packets as fast as possible from each machine and consume all available network bandwidth at the victim. This is a bandwidth consumption attack. The victim cannot defend against this attack on her own, since the legitimate packets get dropped on the upstream link, between the ISP and the victim network. Thus, frequently the victim requests help from its ISP to filter out offending traffic.

2.2.5 Attack Toolkits

While some attackers are sophisticated enough to create their own attack code, far more commonly they use code written by others. Such code is typically built into a general, easily used package called an attack toolkit.

Some Popular DDoS Programs:

Trinoo - uses a handler/agent architecture, wherein an attacker sends commands to the handler via TCP and handlers and agents communicate via UDP. Both handler and agents are password protected to try to prevent them from being taken over by another attacker. Trinoo generates UDP packets of a given size to random ports on one or multiple target addresses, during a specified attack interval.

Tribe Flood Network (TFN) - uses a different type of handler/agent architecture. Commands are sent from the handler to all of the agents from the command line. The attacker does not "log in" to the handler as with trinoo or Stacheldraht. Agents can wage a UDP flood, TCP SYN flood, ICMP Echo flood, and Smurf attacks at specified or random victim ports. The attacker runs commands from the handler using any of a number of connection methods (e.g., remote shell bound to a TCP port, UDP-based client/server remote shells, ICMP-based client/server shells such as LOKI, SSH terminal sessions, or normal telnet TCP terminal sessions). Remote control of TFN agents is accomplished via

ICMP Echo Reply packets. All commands sent from handler to agents through ICMP packets are coded, not clear text, which hinders detection.

Stacheldraht - (German for "barbed wire") combines features of trinoo and TFN tools and adds encrypted communication between the attacker and the handlers. Stacheldraht uses TCP for encrypted communication between the attacker and the handlers, and TCP or ICMP for communication between handler and agents. Another added feature is the ability to perform automatic updates of agent code. Available attacks are UDP flood, TCP SYN flood, ICMP Echo flood, and Smurf attacks.

Shaft - is a DDoS tool that shares a combination of features similar to those in trinoo, TFN, and Stacheldraht. Added features are the ability to switch handler and agent ports on the fly (thus hindering detection of the tool by intrusion detection systems), a "ticket" mechanism to link transactions, and a particular interest in packet statistics. Shaft uses UDP for communication between handlers and agents. Remote control is achieved via a simple telnet connection from the attacker to the handler. Shaft uses "tickets" for keeping track of its individual agents. Each command sent to the agent contains a password and a ticket. Both passwords and ticket numbers have to match for the agent to execute the request. Simple letter shifting (a Caesar cipher) is used to obscure passwords in sent commands. Agents can generate a UDP flood, TCP SYN flood, ICMP flood, or all three attack types. The flooding occurs in bursts of 100 packets per host (this number is hard-coded), with the source port and source address randomized. Handlers can issue a special command to agents to obtain statistics on malicious traffic generated by each agent. It is suspected that this is used to calculate the yield of a DDoS network.

Tribe Flood Network 2000 (TFN2K) - is an improved version of the TFN attack tool. It includes several features designed specifically to make TFN2K traffic difficult to recognize and filter; to remotely execute commands; to obfuscate the true source of the traffic, to transport TFN2K traffic over multiple transport protocols including UDP, TCP, and ICMP, and to send "decoy" packets to confuse attempts to locate other nodes in a TFN2K network. TFN2K obfuscates the true traffic source by spoofing source addresses.

Attackers can choose between random spoofing and spoofing within a specified range of addresses. In addition to flooding, TFN2K can also perform some vulnerability attacks by sending malformed or invalid packets.

Trinity - is the first DDoS tool that is controlled via IRC. Upon compromise and infection by Trinity, each machine joins a specified IRC channel and waits for commands. Use of a legitimate IRC service for communication between attacker and agents replaces the classic independent handler and elevates the level of the threat. Trinity is capable of launching several types of flooding attacks on a victim site, including UDP, IP fragment, TCP SYN, TCP RST, TCP ACK, and other floods.

2.3 Intrusion Detection System

Intrusion detection systems (IDSs) are software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems. As network attacks have increased in number and severity over the past few years, intrusion detection systems have become a necessary addition to the security infrastructure of most organizations. Intrusions are caused by attackers accessing the systems from the Internet, authorized users of the systems who attempt to gain additional privileges for which they are not authorized, and authorized users who misuse the privileges given them.

2.3.1 Need for IDS

Intrusion detection allows organizations to protect their systems from the threats that come with increasing network connectivity and reliance on information systems. Given the level and nature of modern network security threats, the question for security professionals should not be *whether* to use intrusion detection, but *which* intrusion detection features and capabilities to use. IDSs have gained acceptance as a necessary addition to every organization's security infrastructure. Despite the documented contributions intrusion detection technologies make to system security, in many organizations one must still justify the acquisition of IDSs. There are several compelling reasons to acquire and use IDSs:

1.) To prevent problem behaviors by increasing the perceived risk of discovery and punishment for those who would attack or otherwise abuse the system.

2.) To detect attacks and other security violations those are not prevented by other security measures.

3.) To detect and deal with the preambles to attacks (commonly experienced as network probes and other "doorknob rattling" activities)

4.) To document the existing threat to an organization.

5.) To provide useful information about intrusions that do take place, allowing improved diagnosis, recovery, and correction of causative factors.

2.3.2 Classification of IDS

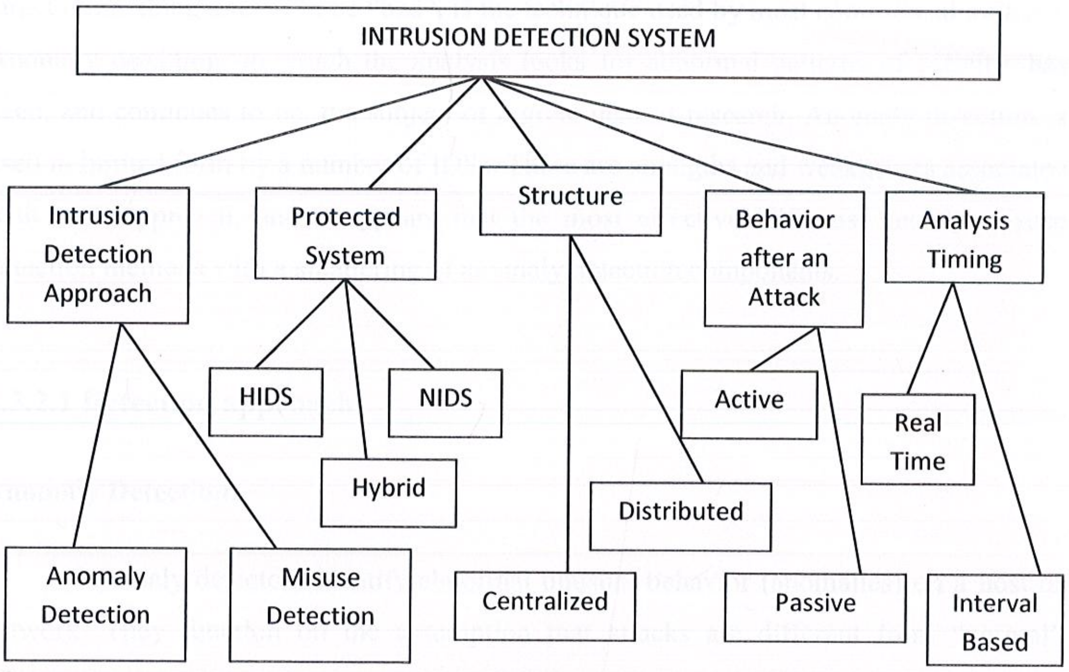


Fig. 4: Classification of IDS

Intrusions can be divided into 6 main types:

1. Attempted break-ins, which are detected by atypical behavior profiles or violations of security constraints.
2. Masquerade attacks, which are detected by atypical behavior profiles or violations of security constraints.
3. Penetration of the security control system, which are detected by monitoring for specific patterns of activity.
4. Leakage, which is detected by atypical use of system resources.
5. Denial of service, which is detected by atypical use of system resources.
6. Malicious use, which is detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

There are two primary approaches to analyzing events to detect attacks:

Misuse detection and anomaly detection. Misuse detection, in which the analysis targets something known to be "bad", is the technique used by most commercial systems. Anomaly detection, in which the analysis looks for abnormal patterns of activity, has been, and continues to be, the subject of a great deal of research. Anomaly detection is used in limited form by a number of IDSs. There are strengths and weaknesses associated with each approach, and it appears that the most effective IDSs use mostly misuse detection methods with a smattering of anomaly detection components.

2.3.2.1 Detection approach

Anomaly Detection:

Anomaly detectors identify abnormal unusual behavior (anomalies) on a host or network. They function on the assumption that attacks are different from "normal" (legitimate) activity and can therefore be detected by systems that identify these differences. Anomaly detectors construct profiles representing normal behavior of users, hosts, or network connections. These profiles are constructed from historical data collected over a period of normal operation. The detectors then collect event data and use a variety of measures to determine when monitored activity deviates from the norm.

Advantages:

- IDSs based on anomaly detection detect unusual behavior and thus have the ability to detect symptoms of attacks without specific knowledge of details.
- Anomaly detectors can produce information that can in turn be used to define signatures for misuse detectors.

Disadvantages:

- Anomaly detection approaches usually produce a large number of false alarms due to the unpredictable behaviors of users and networks.

- Anomaly detection approaches often require extensive "training sets" of system event records in order to characterize normal behavior patterns.

Misuse Detection

Misuse detectors analyze system activity, looking for events or sets of events that match a predefined pattern of events that describe a known attack. As the patterns corresponding to known attacks are called *signatures*, misuse detection is sometimes called "signature-based detection." The most common form of misuse detection used in commercial products specifies each pattern of events corresponding to an attack as a separate signature. However, there are more sophisticated approaches to doing misuse detection (called "state-based" analysis techniques) that can leverage a single signature to detect groups of attacks.

Advantages:

- Misuse detectors are very effective at detecting attacks without generating an overwhelming number of false alarms.
- Misuse detectors can quickly and reliably diagnose the use of a specific attack tool or technique. This can help security managers prioritize corrective measures.
- Misuse detectors can allow system managers, regardless of their level of security expertise, to track security problems on their systems, initiating incident handling procedures.

Disadvantages:

- Misuse detectors can only detect those attacks they know about –therefore they must be constantly updated with signatures of new attacks. □□Many misuse detectors are designed to use tightly defined signatures that prevent them from detecting variants of common attacks. State-based misuse detectors can overcome this limitation, but are not commonly used in commercial IDSs.

2.3.2.2 Information Based

The most common way to classify IDSs is to group them by information source. Some IDSs analyze network packets, captured from network backbones or LAN segments, to find attackers. Other IDSs analyze information sources generated by the operating system or application software for signs of intrusion.

Network-Based IDSs

The majority of commercial intrusion detection systems are network based. These IDSs detect attacks by capturing and analyzing network packets. Listening on a network segment or switch, one network-based IDS can monitor the network traffic affecting multiple hosts that are connected to the network segment, thereby protecting those hosts. Network-based IDSs often consist of a set of single-purpose sensors or hosts placed at various points in a network. These units monitor network traffic, performing local analysis of that traffic and reporting attacks to a central management console. As the sensors are limited to running the IDS, they can be more easily secured against attack. Many of these sensors are designed to run in "stealth" mode, in order to make it more difficult for an attacker to determine their presence and location.

Advantages:

- A few well-placed network-based IDSs can monitor a large network.
- The deployment of network-based IDSs has little impact upon an existing network. Network-based IDSs are usually passive devices that listen on a network wire without interfering with the normal operation of a network. Thus, it is usually easy to retrofit a network to include network-based IDSs with minimal effort.
- Network-based IDSs can be made very secure against attack and even made invisible to many attackers.

Disadvantages:

- Network-based IDSs may have difficulty processing all packets in a large or busy network and, therefore, may fail to recognize an attack launched during periods of high traffic. Some vendors are attempting to solve this problem by implementing IDSs completely in hardware, which is much faster. The need to analyze packets quickly also forces vendors to both detect fewer attacks and also detect attacks with as little computing resource as possible which can reduce detection effectiveness.
- □Many of the advantages of network-based IDSs don't apply to more modern switch-based networks. Switches subdivide networks into many small segments and provide dedicated links between hosts serviced by the same switch. Most switches do not provide universal monitoring ports and this limits the monitoring range of a network-based IDS sensor to a single host. Even when switches provide such monitoring ports, often the single port cannot mirror all traffic traversing the switch.
- Network-based IDSs cannot analyze encrypted information. This problem is increasing as more organizations (and attackers) use virtual private networks.
- Most network-based IDSs cannot tell whether or not an attack was successful; they can only discern that an attack was initiated. This means that after network-based IDS detects an attack, administrators must manually investigate each attacked host to determine whether it was indeed penetrated.

Some network-based IDSs have problems dealing with network based attacks that involve fragmenting packets. These malformed packets cause the IDSs to become unstable and crash.

Host-Based IDSs

Host-based IDSs operate on information collected from within an individual computer system. (Note that application-based IDSs are actually a subset of host-based IDSs.) This vantage point allows host based IDSs to analyze activities with great reliability and precision, determining exactly which processes and users are involved in a particular attack on the operating system. Furthermore, unlike network based IDSs, host-based IDSs can "see" the outcome of an attempted attack, as they can directly access and monitor the data files and system processes usually targeted by attacks. Host-based IDSs normally utilize information sources of two types, operating system audit trails, and system logs. Operating system audit trails are usually generated at the innermost (kernel) level of the operating system, and are therefore more detailed and better protected than system logs. However, system logs are much less obtuse and much smaller than audit trails, and are furthermore far easier to comprehend.

Advantages:

- Host-based IDSs, with their ability to monitor events local to a host, can detect attacks that cannot be seen by network-based IDS.
- □Host-based IDSs can often operate in an environment in which network traffic is encrypted, when the host-based information sources are generated before data is encrypted and/or after the data is decrypted at the destination host.
- □Host-based IDSs are unaffected by switched networks.
- When Host-based IDSs operate on OS audit trails, they can help detect Trojan horse or other attacks that involve software integrity breaches. These appear as inconsistencies in process execution.

Disadvantages:

- Host-based IDSs are harder to manage, as information must be configured and managed for every host monitored.
- Since at least the information sources (and sometimes part of the analysis engines) for host-based IDSs reside on the host targeted by attacks, the IDS may be attacked and disabled as part of the attack.

- Host-based IDSs are not well suited for detecting network scans or other such surveillance that targets an entire network, because the IDS only sees those network packets received by its host.
- Host-based IDSs can be disabled by certain denial-of-service attacks.
- When host-based IDSs use operating system audit trails as an information source, the amount of information can be immense, requiring additional local storage on the system.
- Host-based IDSs use the computing resources of the hosts they are monitoring, therefore inflicting a performance cost on the monitored systems.

HYBRID

There is a blend of HIDS and NIDS as a separate class of a *Network Node IDS (NNIDS*) which has its agents deployed on every host within the network being protected. In fact, a NNIDS operates very much like a hybrid per-host NIDS since a single agent usually processes the network traffic directed to the host it runs upon (an "every man for himself approach"). The main reason for introducing such hybrid IDS was the need to work online with encrypted networks and their data destined to the single host (only the source and destination can see decrypted network traffic). Most large commercially offered intrusion detection systems are shim-hybrid ones, i.e. those that merge the strengths of HIDS and NIDS in a unique concept.

2.3.2.2 Structure Based

Centralized:

Organizations must determine the flow of commands and data from agents to their respective managers. Should all IDS communications and data travel from agents to distributed managers in the field, or should a single manager (or central manager farm) govern the IDS infrastructure? Data collection is a by-product of this hierarchy in that data should flow along the same path as commands. We firmly believe that, as a function of cost and logistics, it is simply smarter operational practice to house IDS managers centrally. The following are two important benefits of doing so:

1. Centralized IDS management decreases the amount of equipment (both hardware and software) that an organization must maintain and administer, resulting in:
    - Reduced costs, due to fewer systems, minimal travel and maintenance expenses;
    - Less administrative effort/manpower; and,
    - More efficiency, as each manager can direct multiple devices over many networks for numerous IDS solutions (as opposed to one manager for each subsection of the network).
2. Centralized control of IDS managers also simplifies the overall network architecture and reduces the number of vulnerable points in an organization's security infrastructure

Fully Distributed

Monitoring and detection is done using an agent-based approach, where response decisions are made at the point of analysis. A distributed IDS (dIDS) consists of multiple Intrusion Detection Systems (IDS) over a large network, all of which communicate with each other, or with a central server that facilitates advanced

network monitoring, incident analysis, and instant attack data. By having these co-operative agents distributed across a network, incident analysts, network operations, and security personnel are able to get a broader view of what is occurring on their network as a whole.

A dIDS also allows a company to efficiently manage its incident analysis resources by centralizing its attack records and by giving the analyst a quick and easy way to spot new trends and patterns and to identify threats to the network across multiple network segments. This article will discuss distributed intrusion detection systems, including the general setup of a dIDS and a fictional case study to demonstrate the distributed analysis abilities. It will also try to give the reader some insight into the benefits of running a dIDS system, from both incident analyst and corporate views.

2.3.2.3 Behavior after attacks

Active Responses

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

Collect additional information

The most innocuous, but at times most productive, active response is to collect additional information about a suspected attack. Each of us has probably done the equivalent of this when awakened by a strange noise at night. The first thing one does in such a situation is to listen more closely, searching for additional information that allows you to decide whether you should take action.

In the IDS case, this might involve increasing the level of sensitivity of information sources. This option also allows the organization to gather information that can be used to support investigation and apprehension of the attacker, and to support criminal and civil legal remedies.

Change the Environment

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

- Injecting TCP reset packets into the attacker's connection to the victim system, thereby terminating the connection

• Reconfiguring routers and firewalls to block packets from the attacker's apparent location (IP address or site),

• Reconfiguring routers and firewalls to block the network ports, protocols, or services being used by an attacker, and

• In extreme situations, reconfiguring routers and firewalls to sever all connections that use certain network interfaces.

Take Action against the Intruder

Some who follow intrusion detection discussions, especially in information warfare circles, believe that the first option in active response is to take action against the intruder. The most aggressive form of this response involves launching attacks against or attempting to actively gain information about the attacker's host or site. However tempting it might be, this response is ill advised. Due to legal ambiguities about civil liability, this option can represent a greater risk than the attack it is intended to block. The first reason for approaching this option with a great deal of caution is that it may be illegal. Furthermore, as many attackers use false network addresses when attacking systems, it carries with it a high risk of causing damage to innocent Internet sites and users. Finally, strike back can escalate the attack, provoking an attacker who originally intended only to browse a site to take more aggressive action. Should an active intervention and trace back of this sort be warranted (as in the case of a critical system) human control and supervision of the process is advisable. We strongly recommend that you obtain legal advice before pursuing any of these "strike-back" options.

Passive Responses

Passive IDS responses provide information to system users, relying on humans to take subsequent action based on that information. Many commercial IDSs rely solely on passive responses.

Alarms and Notifications:

Alarms and notifications are generated by IDSs to inform users when attacks are detected. Most commercial IDSs allow users a great deal of latitude in determining how and when alarms are generated and to whom they are displayed.

The most common form of alarm is an onscreen alert or popup window. This is displayed on the IDS console or on other systems as specified by the user during the configuration of the IDS. The information provided in the alarm message varies widely, ranging from a notification that an intrusion has taken place to extremely detailed messages outlining the IP addresses of the source and target of the attack, the specific attack tool used to gain access, and the outcome of the attack.

2.3.2.4 Analysis Timing

Timing refers to the elapsed time between the events that are monitored and the analysis of those events.

Interval-Based (Batch Mode)

In interval-based IDSs, the information flow from monitoring points to analysis engines is not continuous. In effect, the information is handled in a fashion similar to "store and forward" communications schemes.

Many early host-based IDSs used this timing scheme, as they relied on operating system audit trails, which were generated as files. Interval based IDSs are precluded from performing active responses.

With on the fly processing, an IDS performs online verification of system events. Generally, a stream of network packets is constantly monitored constantly. With this type of processing, intrusion detection uses the knowledge of current activities over the network to sense possible attack attempts (it does not look for successful attacks in the past).

Given the computation complexity, the algorithms that are used here are limited to quick and efficient procedures that are often algorithmically simple. This is due to a compromise between the main requisite – attack detection capability and the complexity of data processing mechanisms used in the detection itself.

At the same time, construction of an on-the-fly processing IDS tool requires a large amount of RAM (buffers) since no data storage is used. Therefore, such an IDS may sometime miss packets, because realistic processing of too many packets is not available.

The amount of data collected by the detector is small since it views only buffer contents. Hence, only small portions of information can be analyzed for searching certain values or sequences.

Real-Time (Continuous)

Real-time IDSs operate on continuous information feeds from information sources. This is the predominant timing scheme for network based IDSs, which gather information from network traffic streams. In this document, we use the term "real-time" as it is used in process control situations. This means that detection performed by a "real-time" IDS yields results quickly enough to allow the IDS to take action that affects the progress of the detected attack.

Advantages:

- they excel at detecting attacks in progress and even responding to (blocking) them;
- the ability to cover network-inherent security holes associated with vulnerability to many types of attacks, particularly DoS, which cannot be detected using a common audit trail analysis approach—network traffic analysis is needed here;
- the system resources are less consumed than in the case of audit trail processing.

Disadvantages:

- Source identification is accomplished based on the network address derived from the packet .The source address may be spoofed, making attacks harder to trace and respond to automatically.
- That they cannot handle encrypted packets thereby not providing essential information required for intrusion detection.
- Since the analytical module uses a limited portion of source information (buffer content only), its detection capability is limited.
- A continuous scanning of network traffic reduces the network throughout the segment on which the IDS sits. This is of particular importance when an IDS tool is deployed near the firewall.

<u>2.3.3 Tools that complement IDS</u>

Several tools exist that complement IDSs and are often labeled as intrusion detection products by vendors since they perform similar functions. This section discusses four of these tools, Vulnerability Analysis Systems, File Integrity Checkers, Honey Pots, and describes how they can enhance an organization's intrusion detection capability.

2.3.3.1 Vulnerability analysis and assessment systems

Vulnerability analysis (also known as vulnerability assessment) tools test to determine whether a network or host is vulnerable to known attacks. Vulnerability assessment represents a special case of the intrusion detection process. The information sources used are system state attributes and outcomes of attempted attacks. The information sources are collected by a part of the assessment engine. The timing of analysis is interval-based or batch-mode and the type of analysis is misuse detection. This means that vulnerability assessment systems are essentially batch mode misuse detectors that operate on system state information and results of specified test routines.

Vulnerability analysis is a very powerful security management technique, but is suitable as a complement to using IDS, not as a replacement. Should an organization rely solely on vulnerability analysis tools to monitor systems, a knowledgeable attacker may monitor the vulnerability analysis system, note when the information source is collected, and time the attack to fit between collection times.

Advantages

- Vulnerability Analysis is of significant value as a part of a security monitoring system, allowing the detection of problems on systems that cannot support an IDS.
- Vulnerability Analysis Systems provide security-specific testing capabilities for documenting the security state of systems at the start of a security program and for reestablishing the security baseline whenever major changes occur.

Disadvantages and Issues

- Host-based vulnerability analyzers are tightly bound to specific operating systems and applications; they are therefore often more costly to build, maintain, and manage.
- Network-based vulnerability analyzers are platform-independent, but less accurate and subject to more false alarms.
- Some network-based checks, especially those for denial-of service attacks, can crash the systems they're testing.
- When conducting vulnerability assessment of networks on which intrusion detection systems are running, the IDSs can block subsequent assessments. Worse yet, repeated network-based assessments can "train" certain anomaly-detection-based IDSs to ignore real attacks.
- Organizations that use vulnerability assessment systems must take care to assure that their testing is limited to systems within their political or management control

boundaries. Privacy issues must be taken into account, especially when employee or customer personal data is included in information sources.

2.3.3.2 File Integrity Checkers

File Integrity Checkers are another class of security tools that complement IDSs. They utilize message digest or other cryptographic checksums for critical files and objects, comparing them to reference values, and flagging differences or changes.

The use of cryptographic checksums is important, as attackers often alter system files, at three stages of the attack. First, they alter system files as the goal of the attack (e.g., Trojan Horse placement), second, they attempt to leave back doors in the system through which they can reenter the system at a later time, and finally, they attempt to cover their tracks so that system owners will be unaware of the attack.

Although File Integrity Checkers are most often used to determine whether attackers have altered system files or executables, they can also help determine whether vendor-supplied bug patches or other desired changes have been applied to system binaries. They are extremely valuable to those conducting a forensic examination of systems that have been attacked, as they allow quick and reliable diagnosis of the footprint of an attack. This enables system managers to optimize the restoration of service after incidents occur.

2.3.3.3 Honey Pot Systems

Several novel additions to the intrusion detection product line are under development and may soon become available. It is important to understand how these products differ from traditional IDSs and to realize that they are not yet widely used.

Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems. Honey pots are designed to:

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

These systems are filled with fabricated information designed to appear valuable but that a legitimate user of the system wouldn't access. Thus, any access to the honey pot is suspect. The system is instrumented with sensitive monitors and event loggers that detect these accesses and collect information about the attacker's activities.

- An expert attacker, once diverted into a decoy system, may become angry and launch a more hostile attack against an organization's systems.
- A high level of expertise is needed for administrators and security managers in order to use these systems.

2.4 FIREWALLS

Internet connectivity is no longer optional for organizations. The information and services available are essential to the organization. Moreover, individual users within the organization want and need Internet access, and if this is not provided via their LAN, they will use dial-up capability from their PC to an Internet service provider (ISP). However, while Internet access provides benefits to the organization, it enables the outside world to reach and interact with local network assets. This creates a threat to the organization. While it is possible to equip each workstation and server on the premises network with strong security features, such as intrusion protection, this is not a practical approach. Consider a network with hundreds or even thousands of systems, running a mix of various versions of UNIX, plus Windows. When a security flaw is discovered, each potentially affected system must be upgraded to fix that flaw. The alternative, increasingly accepted, is the firewall. The firewall is inserted between the premises network and the Internet to establish a controlled link and to erect an outer security wall or perimeter. The aim of this perimeter is to protect the premises network from Internet-based attacks and to provide a single choke point where security and audit can be imposed. The firewall may be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

2.4.2 Applications of Firewall

A firewall is primarily used to provide security to the computer network of an organization. Since a firewall may be used to prevent a large class of network attacks, we may classify the applications of firewall as follows:

1.) ***Filtering:*** Firewalls can filter a data packet at various layers of the Internet protocol hierarchy. A firewall is generally deployed at the perimeter of the computer network of an organization. It can, therefore, examine every data packet that goes in or comes out of the computer network. This feature allows a firewall to deny the malicious data packets from entering the computer network of the organization.

2.) ***DOS prevention*:** A firewall is often used to prevent *Denial-of-Service* attacks launched against the hosts in the computer network of the organization. Such a firewall enforces a maximum on the number of simultaneous connections that can be opened on any host at a given point of time. At this stage, if a user in the outside Internet desires to open a new connection on the host, the firewall simply rejects the data packet from entering the computer network.

3.) ***Detection of IP address spoofing*:** A firewall can be configured to detect and filter a data packet that has an incorrect source IP address. In order to check the legitimacy of the source IP address, the firewall connects to the DHCP server of the source host and verifies that the hardware address i.e. the MAC address of the source host corresponds to its IP address

2.4.3 Capabilities of Firewall

The following capabilities are within the scope of a firewall:

1. A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or

leaving the network, and provides protection from various kinds of IP spoofing and routing attacks. The use of a single choke point simplifies security management because security capabilities are consolidated on a single system or set of systems.

2. A firewall provides a location for monitoring security-related events. Audits and alarms can be implemented on the firewall system.
3. A firewall is a convenient platform for several Internet functions that are not security related. These include a network address translator, which maps local addresses to Internet addresses, and a network management function that audits or logs Internet usage.

2.4.4 Limitations

Firewalls have their limitations, including the following:

1. The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.
2. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.
3. The firewall cannot protect against the transfer of virus-infected programs or files. Because of the variety of operating systems and applications supported inside the perimeter, it would be impractical and perhaps impossible for the firewall to scan all incoming files, e-mail, and messages for viruses.

2.5 Honeypots

A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource. The strategy behind honeypots is to shut intruders safely from production systems and to obtain information about the intruders by logging their actions. Honeypots do not replace firewalls and IDS but are used in conjunction with them.

2.5.1 Purposes of Honeypot

1) The first purpose of Honeypots is to enable the controller to view the hacker and see where and how the hacker is breaking in.
2) The second purpose of Honeypots is to trap and stop the hacker while he is breaking into the system.
3) The third purpose of Honeypots is to enable the controllers to make better and more secure systems.

2.5.2 Classification of Honeypots

Honeypots are broadly classified via two methods:

1. Based on usage

Production honeypots are used to protect organizations. Production honeypots can significantly reduce the risk of intrusion by uncovering vulnerabilities and alert administrators of attacks. They add value to security measures of an organization. Commercial organizations used production honeypots to help protect their network.

Research honeypots are used to research the threats organizations face, and how to better protect against those threats. These types of honeypots are more focused on researching the actions of intruder by using a number of different configurations to lure them in. The Honeypot Project, for example is a volunteer, non-profit security research organization that uses honeypots to collect information to cyber threats.

2. Based on level of involvement

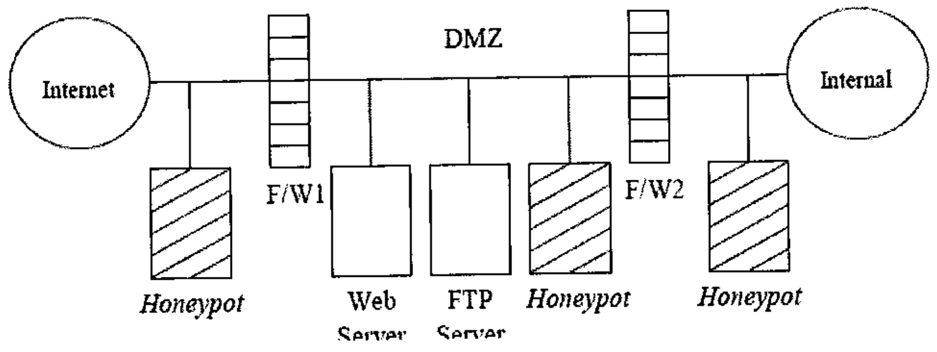
Low Interaction Honeypots would be characterized by its minimal interaction with the hacker. These types of honeypots typically emulate a specific service like ftp or http. They are much simpler to deploy and maintain, but bog only a limited amount of information regarding the hacker's activities

High Interaction Honeypot gives a real operating system to attack upon. This exposes the system to ample of risk and complexity. At the same time the possibility to accumulate information about the attack as well as the attractiveness of the honeypot increases a lot, so they are specially used for research purposes.

Low interaction	**High interaction**
• Solution emulates operation systems and services. • Minimal risk, as the emulated services control what attackers can and cannot do. • Captures limited amounts of information, mainly transactional data and some limited interaction	• No emulation, real operating systems and services are provided • Can capture far more information, including new tools, communications, or attacker keystrokes. • Can be complex to install or deploy (commercial versions tend to be much simpler). • Increased risk, as attackers are provided real operating systems to interact with

2.5.3 Placement of Honeypots

Honeypots can be placed externally as well as internally. Conceptually they can be placed at three main locations in an organization:



A honeypot does not need a certain surrounding environment as it is a standard server with no special needs. A honeypot can be placed anywhere a server could be placed. But certainly, some places are better for certain approaches as others. A honeypot can be used on the Internet as well as the intranet, based on the needed service.

Placing a honeypot on the intranet can be useful if detecting some bad guys inside a private network is wished. It is especially important to set the internal thrust for a honeypot as low as possible as this system could be compromised, probably without immediate knowledge. A honeypot can be placed at following locations:

- In front of the firewall (Internet)
- DMZ
- Behind the firewall (intranet)

Each approach has its advantages as well as disadvantages.

Sometimes it is even impossible to choose freely as placing a server in front of a firewall is simply not possible or not wished. By placing the honeypot in front of a firewall, the risk for the internal network does not increase. The danger of having a compromised system behind the firewall is eliminated. A honeypot will attract and generate a lot of unwished traffic like port scans or attack patterns. By placing a honeypot outside the firewall, such events do not get logged by the firewall and an internal IDS system will not generate alerts. Otherwise, a lot of alerts would be generated on the firewall or IDS. Probably the biggest advantage is that the firewall or IDS, as well as any other resources, have not to be adjusted as the honeypot is outside the firewall and viewed as any other machine on the external network. Running a honeypot does therefore not increase the dangers for the internal network nor does it introduce new risks.

The disadvantage of placing a honeypot in front of the firewall is that internal attackers can not be located or trapped that easy, especially if the firewall limits outbound traffic and therefore limits the traffic to the honeypot.

Placing a honeypot inside a DMZ honeypot (B) seems a good solution as long as the other systems inside the DMZ can be secured against the honeypot. Most DMZs are not fully accessible as only needed services are allowed to pass the firewall. In such a case, placing the honeypot in front of the firewall should be favored as opening all corresponding ports on the firewall is too time consuming and risky.

A honeypot behind a firewall (honeypot (C)) can introduce new security risks to the internal network, especially if the internal network is not secured against the honeypot through additional firewalls. This could be a special problem if the IPs are used for authentication. It is important to distinguish between a setup where the firewall enables access to the honeypot or where access from the Internet is denied. A honeypot does often provide a lot of services. Probably most of them are not used as exported services to the Internet and are therefore not forwarded to the honeypot by the firewall. By placing the honeypot behind a firewall, it is inevitable to adjust the firewall rules if access from the Internet should be permitted. The biggest problem arises as soon as the internal honeypot is compromised by an external attacker. He gains the possibility to access the internal network through the honeypot. This traffic will be unstopped by the firewall as it is regarded as traffic to the honeypot only, which in turn is granted. Securing an internal honeypot is therefore mandatory, especially if it is a high-involvement honeypot. With an internal honeypot it is also possible to detect a misconfigured firewall which forwards unwanted traffic from the Internet to the internal network. The main reason for placing a honeypot behind a firewall could be to detect internal attackers. The best solution would be to run a honeypot in its own DMZ, therefore with a preliminary firewall. The firewall could be connected directly to the Internet or intranet, depending on the goal. This attempt enables tight control as well as a flexible environment with maximal security.

2.5.4 Current Honeypot Technology

Several honeypot systems have been selected to show the status of honeypot products. Each is a sample of one kind. They are investigated concerning security value, interaction and virtualization.

Back officer Friendly (BOF)

It is a lightweight honeypot and free to distribute. It represents an accurate distillation of the ideas and insights of honeypot. BOF emulates several common services such as http, ftp, telnet, mail etc. BOF logs, alerts and responses a fake reply whenever someone connects to such ports. BOF user can have clear view of the attacking process.

Specter

Specter is a commercial production honeypot whose value lies in detection. Specter can stimulate 13 different operating systems in application level including Windows, Linux, Aix, Solaris, MacOS etc. Its windows based software which offers 14 different network services and traps. The other character is actively gathering attackers' information such as Whois and DNS lookup. Specter is a low interactive honeypot which fakes the reply of attacker's request.

Honeyd

Honeyd is a powerful production honeypot, which can be used for attacks detection and reaction. It represents today's level of production honeypot in many fields.

First, it can emulate over 400 kinds of OS at IP stack level. This hides the guest OS before attacker. Second, it emulates hundreds of computers at a single machine. Third, Honeyd is Open Source honeypot system. It is free to use and easy to modify for particular requirement. Honeyd still use the simulated service reply to attacker's request, but administrator can customize the reply script to provide attacker more flexibility.

Homemade

Homemade honeypots are honeypots created by individuals or organizations to suit a specific need. Since no two are alike, it is possible to have low-interaction or high-interaction homemade honeypots. However, in general, homemade honeypots tend to medium interaction. We will cover several possible homemade honeypot solutions and how they can best be used. Homemade honeypots can be designed for production or research purposes, depending on how they are built, deployed, and used.

Honeynet

Honeynet represents the highest level of research honeypot. It is a high interaction honeypot which is primarily used for research. It can also be modified to production honeypot for attack detection and reaction. New methods of data capture and data control proposed by Honeynet Project show greater flexibility and higher access control ability, which can be applied to both research honeypot and production honeypot.

2.5.5 Advantages and Disadvantages of Honeypots

If knowledge is power to the attacker, so is it to the security practitioner. Knowing both the advantages and the disadvantages of honeypots is a must-know. By knowing the inherent risks in honeypots, we can use this knowledge to mitigate these risks and circumvent the disadvantages. We highlight some of these disadvantages and advantages below:

Advantages

- Simplicity of the honeypot.
- They do not include any actual production services, so no one in the organization will need to access the honeypot. For this reason, any connection to the honeypot is likely scan or attack.
- Unlike intrusion detection systems, which only identify when and how an attacker got into the network, a honeypot is a distraction as well.
- Small data sets of high value: Honeypots reduce 'noise' by collecting only small data sets, but information of high value, as it is only the bad guys. This means its much easier (and cheaper) to analyze the data a honeypot collects an derive value from it. They provide a lot of useful information on attacker's movement within the system.
- A honeypot can also be used as an early warning system, alerting administrators of any hostile intent before the real system is compromised.
- Minimal resources: Honeypots require minimal resources, they only capture bad activity.
- Information: Honeypots can collect in-depth information that few, if any other technologies can match.

Disadvantages

- Honeypots are worthless if no one attacks them.
- Honeypots can be used as a launching platform to attack other machines.
- Honeypots encourage an aggressive atmosphere and add risk to a network.
- If a honeypot is successfully broken into, the attacker can use this to begin other attacks.
- If the honeypot is attacked, the administrator should prevent major changes to the honeypot, because any noticeable changes will make the attacker more suspicious.

CHAPTER 3: HONEYPOT DESIGNS AND RELATED RESEARCH WORKS

3.1 Concept of Real and Virtual Honeypots

There are several different types of honeypots. Honeypots can mimic just about anything on a network. Most of the time honeypots are configured to look like a server, but a honeypot can also pose as a workstation or even as a Cisco router for that matter.

Regardless of what a honeypot is posing as, honeypots have two main kinds based on concept of deception and implementation: **Real** and **Virtual**

Both types of honeypots have their advantages and disadvantages.

Virtual Honeypot

A virtual honeypot is basically an emulated server. There are both hardware and software implementations of virtual honeypots. For example, if a network administrator was concerned that someone might try to exploit an FTP server, the administrator might deploy a honeypot appliance that emulates an FTP server.

Virtual honeypots have both their good and bad points. The primary advantages to virtual honeypots are:

1. They are cheaper
2. Easier to deploy
3. They are more secure than real honeypots.
4. Because of a virtual emulation, there's no operating system and therefore no way that a hacker could use the honeypot to compromise the rest of your network.

The disadvantages are:

1. Virtual honeypots will not fool a skilled hacker for long. Virtual honeypots don't have an underlying operating system (aside from perhaps a very limited embedded version of Windows or Linux). Because of this, many of the commands that a more experienced hacker might issue simply won't work. This instant tip-off tells hackers that they're accessing a honeypot rather than an actual server.
2. They have limited information gathering capabilities.
3. The other problem with virtual honeypots is the way that they gather information. For example, if a honeypot is emulating an FTP server, then the honeypot is probably only programmed to gather information related to known attack patterns against FTP servers. If a hacker tries something new or encrypts the attack packets with IPv6, then there is a good chance that the honeypot may not know how to react to the exploit.

Real Honeypots

Real honeypots run on real operating systems (usually Windows or Linux) and real server software i.e. they work on real components rather than emulating an environment.

The main advantages of a Real honeypot are:

1. Because a real operating system is involved, the honeypot will react to a hack attempt in exactly the same way that a production server would.
2. Because a real honeypot is not limited to the constraints of an emulator, it can log any type of attempt to breach security, even if the attack uses a previously unknown technique.

There are several negative issues associated with real honeypots though

1. They are expensive. A real honeypot runs a real operating system. This means that in addition to purchasing the honeypot software, server hardware and an operating system license must also be purchased
2. Real honeypots are also more difficult to deploy than virtual honeypots are. This is because every effort must be made to secure the honeypot's operating system. That leads to the most serious negative aspect of real honeypots. If a hacker does manage to take control of a honeypot machine, the honeypot can be used as a staging area from which to attack the rest of the network.

3.2 Common Designs of the Honeypot

Several different types of honeypots are employed in the security industry today. Each varies in complexity and functionality, which are closely linked. These range from the simplest (port monitors) to the most complex (full systems plus network IDS) and are briefly discussed below. This section will outline the types, as well as provide an example.

3.2.1 Port Monitors

These are the most common and simplest type of honeypot. The reason is due to the simplicity of the programming. A port monitor is simply a "sockets" based program that opens up to a listening port. A "socket" is defined as the minimum amount of information necessary for communication on the network, and originated from TCP/IP. A socket contains the source/destination IP address, the source/destination port, and the transport protocol (UDP or TCP).

If an attacker discovers the open port, which can often be associated with a defined service, the honeypot can collect the attacker's traffic and log it for future analysis by listening for traffic ports typically scanned by attackers. However, this will alert an attacker that the port is monitored, because the port monitoring system will first accept, and then drop the connection. When a connection is suddenly dropped, it alerts the attacker of the possibility of an IDS running on the port.

3.2.2 Interactive Port Monitor

A port monitor is simply a passive listening device that may actually alert attackers, as discussed above. The next step up in complexity is a deception system that interacts with the attacker. In contrast to a port monitoring system, a deception system will respond to port intrusions as if it is an actual server. For example, if port 23 was opened on a host, an attacker would associate this will the popular program 'telnet'. Using this knowledge to its benefit, the honeypot could reply with a 'login:' string, which could encourage the attacker to start trying to gain access to the system. This will not only distract the attacker from other real systems, but would also provide valuable information about the attacker and their method of operation.

3.2.3 Multi-Protocol Deception Systems

A multi-protocol system is simply a deception system having multi-protocols and banners to emulate packages for different operating systems. Examples of such a system include commercially available systems like the Deception Toolkit (DTK) and SPECTER. Both of these packages simulate multiple operating systems and network services.

3.2.4 Full Systems (with and without IDS)

A full system goes beyond a honeypot (which is implemented strictly for deception). A full system is fully functional and operational, and is usually set to alert on exceptional conditions. A full system with IDS includes a full intrusion detection system to supplement the internal logging of the full system.

Related Research works

1) "Experiences with Honeypot Systems: Development, Deployment and Analysis" by Robert McGrew, Rayford B. Vaughn. It states various experiments with high interaction honeypot and their use to detect threats. High interaction honeypots are real honeypot. They are useful to capture details of vulnerabilities or exploits that are not yet known to the public, but are being used by a small number of attackers. They are useful in detecting new ways of attack as they can dissipate more information than virtual or low interaction honeypot. If an organization is to deploy a high-interaction honeypot for the purposes of identifying threats, it is important to devote resources to analyse the data generated by these honeypots.

2) "Research on network security of defense based on Honeypot " by Jian Bao , Chang-peng Ji and Mo Gao. It consist of experiment with honeypot and various other defence techniques. They consist of Honeypot, firewall, IDS and Honeypot server, integrated use of encryption, access control and other means to provide security protection for system, through the security checks, such as traffic statistics, exception analysis, hole detection, pattern matching, and IDS based on host and network.

3) “Data Analyzer based on data mining for Honeypot Router” by Abdallah Ghourabi, Tarek Abbes and Adel Bouhoula. This paper explore the techniques to analyse the data of the log files created by the honeypot. This paper presents a data analysis tool based on clustering method of data mining. These data will be clustered by using the DBSCAN clustering algorithm in order to classify the captured packets and extract those that are suspicious. Suspicious packets will be then verified by a human expert.

4) “An Integrated Honeypot Framework for Proactive Detection, Characterization and Redirection of DDoS Attacks at ISP level “by Anjali Sardana and R. C. Joshi. This paper provides an end to end solution for defense against both diluted degrading and high rate concentrated flooding DDoS attacks in ISP domain. Presented dynamic honeypot based mitigationscheme where dynamic honeypot engine in honeypot controller automatically triggers the generation of appropriate number of honeypots and servers after each eon. The proposed framework mitigates DDoS attacks meeting the difficult challenges of keeping collateral damage and overheads in terms of memory and computation. The results are promising and show that the framework has the potential to provide stable network functionality even in the presence of high rate attacks.

5) “Honeypot Scheme for Distributed Denial-of-Service Attack” by Vinu V Das This paper has analyzed existing honeypot system to identify some of problems, such as *Legitimate Attacker and Link Unreachable problem. Link Unreachable problem* has been dealt with by opening a temporary communication channel through the honeypot, by virtually making it to act as AS; so other attackers and unauthorized nodes will not be able to intrude into the network and no DoS because it still acts as honeypot to other nodes and

ASs. Apart from perfectly addressing the conventional issues, proposed system is efficient and secure in DoS.

6) Honeyware: a web-based low interaction client honeypotYaser Alosefer, Omer Rana. A low interaction client honeypot uses a lightweight simulation to simulate a client when interacting with a server. The aim of this is to detect malicious code present on a server by interacting with the server and examining its responses to determine if there is any malicious code present. The reply is examined by checking the string for the page, known as the malicious code string. One disadvantage of this is that it cannot detect unknown malware because it uses a signature database to match them with the responses from the server. However, the low interaction client honeypot also has some advantages such as its ease of use and deployment and its speed in reporting results as compared to the high interaction honeypot.

CHAPTER 4: DESIGN AND IMPLEMENTATION

4.1 Proposed Design of the Project

The proposed design is based on the Interactive Port Monitor model of honeypot and is shown as given below.

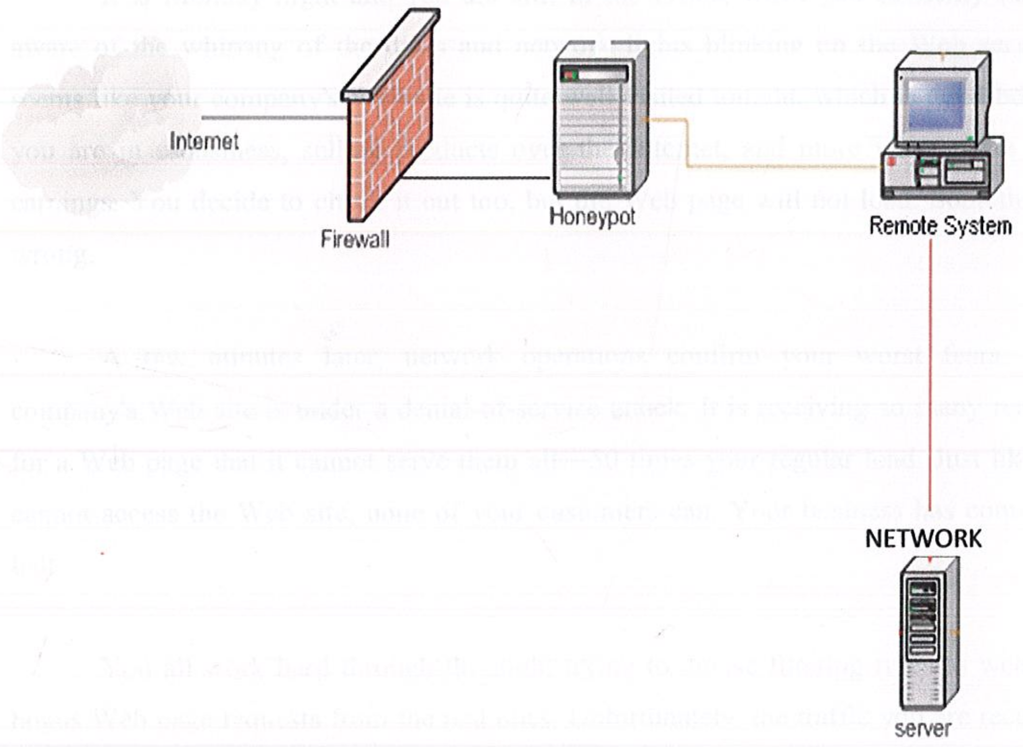


Fig. 6: Proposed design of Honeypot system for detection of DDoS attacks

4.2 Design Components

Our design consists of the following components:

4.2.1 DDoS Attack Tool

4.2.1.1 Scenario

It is Monday night and you are still in the office, when you suddenly become aware of the whirring of the disks and network lights blinking on the Web server. It seems like your company's Web site is quite well visited tonight, which is good because you are in e-business, selling products over the Internet, and more visits mean more earnings. You decide to check it out too, but the Web page will not load. Something is wrong.

A few minutes later, network operations confirm your worst fears. Your company's Web site is under a denial-of-service attack. It is receiving so many requests for a Web page that it cannot serve them all—50 times your regular load. Just like you cannot access the Web site, none of your customers can. Your business has come to a halt.

You all work hard through the night trying to devise filtering rules to weed out bogus Web page requests from the real ones. Unfortunately, the traffic you are receiving is very diverse and you cannot find a common feature that would make the attack packets stand out. You next try to identify the sources that send you a lot of traffic and blacklist them in your firewall. But there seem to be hundreds of thousands of them and they keep changing. You spend the next day bringing up backup servers and watching them overload as your earnings settle around zero.

All you are left with are questions: Why are you being attacked? Is it for competitive advantage? Is an ex-employee trying to get back at you? Is this a very upset customer? How long can your business be offline and remain viable? How did you get into this situation, and how will you get out of it? Or is this just a bug in your own Web applications, swamping your servers accidentally?

4.2.1.2 DoS and DDoS

The goal of a DoS attack is to disrupt some legitimate activity, such as browsing Web pages, listening to an online radio, transferring money from your bank account, or even docking ships communicating with a naval port. This denial-of-service effect is achieved by sending messages to the target that interfere with its operation, and make it hang, crash, reboot, or do useless work.

One way to interfere with a legitimate operation is to exploit vulnerability present on the target machine or inside the target application. The attacker sends a few messages crafted in a specific manner that take advantage of the given vulnerability. Another way is to send a vast number of messages that consume some key resource at the target such as bandwidth, CPU time, memory, etc. The target application, machine, or network spends all of its critical resources on handling the attack traffic and cannot attend to its legitimate clients.

Of course, to generate such a vast number of messages the attacker must control a very powerful machine—with a sufficiently fast processor and a lot of available network bandwidth. For the attack to be successful, it has to overload the target's resources. This means that an attacker's machine must be able to generate more traffic than a target, or its network infrastructure, can handle.

Now let us assume that an attacker would like to launch a DoS attack on example.com by bombarding it with numerous messages. Also assuming that example.com has abundant resources, it is then difficult for the attacker to generate a sufficient number of messages from a single machine to overload those resources. However, suppose he gains control over 100,000 machines and engages them in generating messages to example.com simultaneously. Each of the attacking machines now may be only moderately provisioned (e.g., have a slow processor and be on a modem link) but together they form a formidable attack network and, with proper use, will be able to overload a well-provisioned victim. This is a distributed denial-of-service—DDoS.

Both DoS and DDoS are a huge threat to the operation of Internet sites, but the DDoS problem is more complex and harder to solve. First, it uses a very large number of machines. This yields a powerful weapon. Any target, regardless of how well provisioned it is, can be taken offline. Gathering and engaging a large army of machines has become trivially simple, because many automated tools for DDoS can be found on hacker Web pages and in chat rooms. Such tools do not require sophistication to be used and can inflict very effective damage. A large number of machines give another advantage to an attacker. Even if the target were able to identify attacking machines (and there are effective ways of hiding this information), what action can be taken against a network of 100,000 hosts? The second characteristic of some DDoS attacks that increases their complexity is the use of seemingly legitimate traffic. Resources are consumed by a large number of legitimate-looking messages; when comparing the attack message with a legitimate one, there are frequently no telltale features to distinguish them. Since the attack misuses a legitimate activity, it is extremely hard to respond to the attack without also disturbing this legitimate activity.

Different Types of DDoS attack

DDoS Attack can be broadly classified into 2 types:

1. Vulnerability Attack
2. Flooding Attack

4.2.1.3 Type of Attack used in the Project

Flooding attack - Flooding attacks work by sending a vast number of messages whose processing consumes some key resource at the target. For instance, complex messages may require lengthy processing that takes up CPU cycles, large messages take up bandwidth, and messages that initiate communication with new clients take up memory. Once the key resource is tied up by the attack, legitimate users cannot receive service. The crucial feature of flooding attacks is that their strength lies in the volume, instead of content.

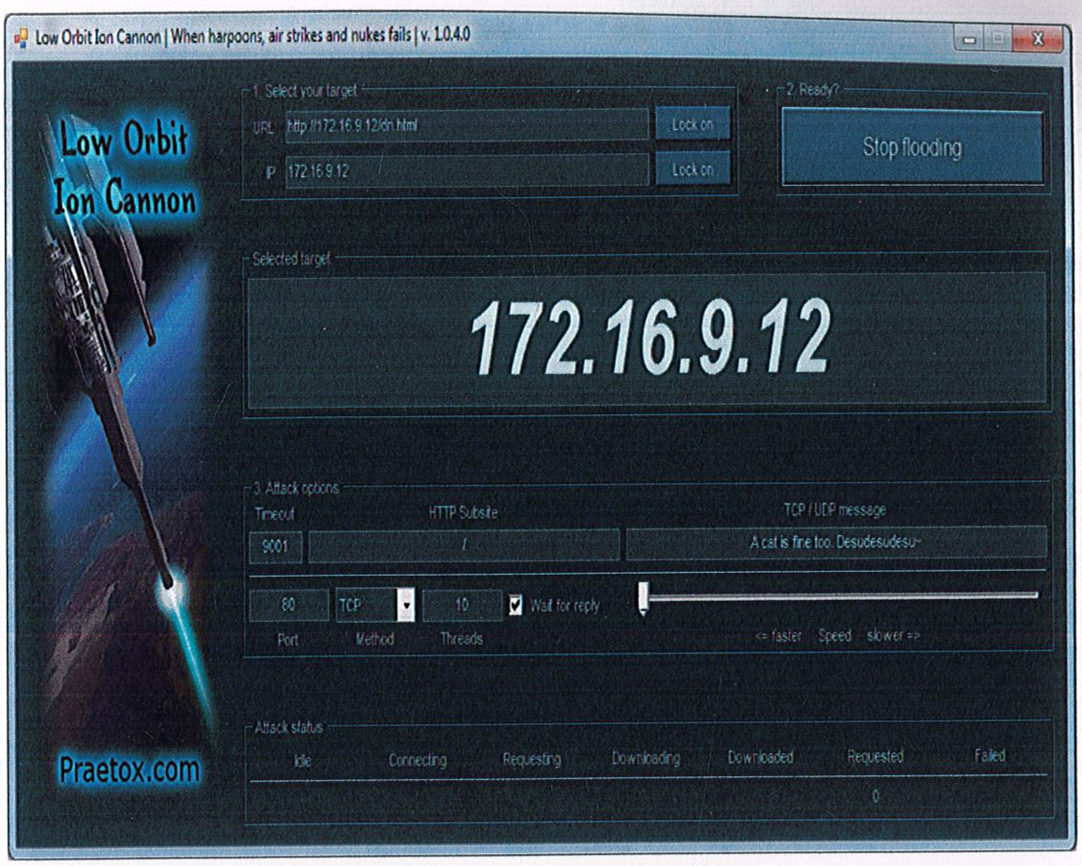
This has two major implications:

a.) The attackers can send a variety of packets. The attack traffic can be made arbitrarily similar to the legitimate traffic, which greatly hinders defense.

b.) The flow of traffic must be very large in order to consume the target's resources. Thus, the attacker usually has to engage more than one machine to send out the attack traffic. Flooding attacks are therefore usually DDoS attacks.

4.2.1.4 Software Used for Launching Flooding Attack

The software used in the project for launching DDoS attack is Low Orbit Ion Cannon (LOIC) v 1.0.4.0 which was taken up from the internet, often used by script kiddies for launching attacks.



URL: http://172.16.9.12/dn.html
IP: 172.16.9.12
Selected target: 172.16.9.12
Timeout: 9001
HTTP Subsite: /
TCP / UDP message: A cat is fine too. Desudesudesu~
Port: 80
Method: TCP
Threads: 10
[✓ Wait for reply]
Requested: 0

Fig. 7: LOIC DDoS attack tool kit

LOIC Flooder v1.0.4.0

This software is used to flood the victim site with large number of TCP and UDP packets thereby slowing down the speed of the victim site i.e. it exhausts the network bandwidth of the victim. Hence other legitimate users who want to have access at the time of attack being launched would be deprived of the service which would result in loss of revenue for the organization hosting the web site. For the attack to be more powerful, it would be simultaneously launched from multiple machines, hence a DDoS attack.

The other reasons for which this tool used is:

1.) It is easy to use. Does not require complex knowledge of computer networks and can be used by a layman

2.) It provides a simple Graphic User Interface (G.U.I).

4.2.1.5 How to launch attack using this tool

This tool takes 3 inputs, which are:

1.) **URL** –it takes the URL address of the site on which the attack is to be launched.

For example, http://www.172.16.9.12/dn.html then lock on.

2.) **IP**- It takes ip address of the system on which the attack is to be launched.

For example, 172.16.9.12 then lock on.

3.) **Port** – it takes the port number as input on which the web service is provided. It is generally 80.

4.) **Threads** - it is the number of request packet which would be sent to target machine. The more the number of these threads more powerful an attack it would be.

5.) **Timeout** – it is the maximum time to wait for a response.

6.) **Method** –it is the type of attack that we are launching.

For example, TCP or UDP.

4.2.2 HONEYPOT

The honeypot shall contain three softwares

4.2.2.1 Packet Sniffer

A **packet sniffer** is computer software or computer hardware that can intercept and log traffic passing over a digital network or part of a network. As data streams flow across the network, the sniffer captures each packet and eventually decodes and analyzes its content

All information that travels across a network is sent in "packets." For example, when an email is sent from one computer to another, it is first broken up into smaller segments. Each segment has the destination address attached, the source address, and other information such as the number of packets and reassembly order. Once they arrive at the destination, the packet's headers and footers are stripped away, and the packets reconstituted.

On wired broadcast LANs, depending on the network structure, one can capture traffic on all or just parts of the traffic from a single machine within the network. In a *switched* Ethernet network, the switch acts like a central switchboard. It receives packets directly from the originating computer, and sends them directly to the machine to which they are addressed. In this scenario, if Computer A sends an email to Computer B, and Computer D is in promiscuous mode, it still won't see the packets. Therefore, some people mistakenly assume a packet sniffer cannot be used on a switched network.

The packet sniffers are versatile. The versatility of packet sniffers means they can be used to:

- Analyze network problems.
- Detect network intrusion attempts.
- Gain information for effecting a network intrusion.
- Monitor network usage.
- Gather and report network statistics.
- Filter suspect content from network traffic.
- Spy on other network users and collect sensitive information such as passwords.
- Debug client/server communications.
- Debug network protocol implementations.

General Description of the Required Product:

The need of our project is to create a simple network sniffer which can parse IP, TCP and UDP packets. This packet sniffer will be able to create a log file of all the packets that are incoming or outgoing from the network node, on which it is installed. The resulting work will be a tool that can be used for the analysis and detection of the network traffic.

Overview of Function Requirements:

The product can run as a stand-alone application on a network. It must be flexible enough to run easily with a variety of different windows and linux applications that are required for our project and should be simple to install and use.

The product requires the use of a pointing device (mouse) and a keyboard to interface with the user. Its user interface uses simple form that does not require use of high end graphics. The product needs the machine on which it is installed to be on a switched Ethernet network.

The product can also make a log of all the incoming and outgoing packets and store it according to the time, so as to ease the analysis process.

Overview of Data Requirements:

The data input by the user will include simple mouse clicks on buttons to select the interface that is the IP address of the host machine and to start and stop the sniffing process. There is storage of data in the database table. The only data that is returned to the user at the run time are packet information in form of a log where each node can be used to know the details of each of the packet that is listed.

General Constraints, Assumptions, Dependencies and Guidelines:

- The machine on which the product is used should be on a network.
- There are no memory requirements.
- The product must be stored in such a way that allows the user easy access to it.
- The product must have a user-friendly interface that is simple to understand.
- Response time for loading the product should take no longer than five minutes.
- A general knowledge of basic computer skills is required to use the product.

User View of Product Use:

After selecting the interface or the IP address of the machine and clicking on START button, the sniffer will start capturing the data packets. STOP button that can be used anytime to stop the sniffer from capturing the packets. There will be a log file that will show the captured packets at the run time. In the middle of the sniffing process, the packets can be viewed along with the details of the same. Later on, the details of the packets can be obtained from the database log file.

Platform:

Taking a look at the requirement specification, we come to know that we need to design a packet sniffer that should have an interactive user interface and easy to understand and use. The designing should be done considering the need of the network and the database to support the purpose of the product. Considering the above points we choose C Programming for coding and GCC compiler to run our program.

Concepts for Developing Sniffer:

TCP Protocol: The Transmission Control Protocol is a connection- and stream-oriented, reliable Point-to-point protocol. TCP communication is analogous to a phone call. TCP uses IP as its network protocol. IP is datagram-oriented and a best-effort protocol. *Best-effort* means that datagrams are sent without the guarantee of delivery and correct order.

As we know, TCP is stream-oriented. TCP must simulate the streaming of data. Therefore, it is necessary that TCP controls the order and correct occurrence of the datagrams. If a datagram is corrupt or lost, it must be resent. If this does not function, an error is reported. The advantage of TCP is its reliability. The disadvantage to TCP is the loss of performance due to the administration overhead for handling the reliability.

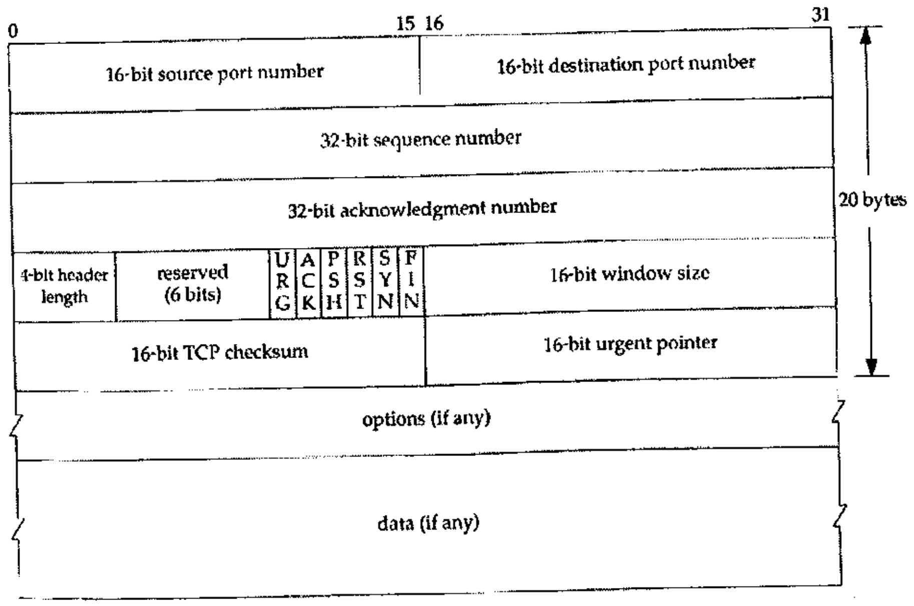


Fig. 8: TCP HEADER

The **Source port** and **Destination port** fields identify the local end points of the connection. The well-known ports are defined at www.iana.org but each host can allocate the others as it wishes. A port plus its host's IP address forms a 48-bit unique end point. The source and destination end points together identify the connection.

The **Sequence number** and **Acknowledgement number** fields perform their usual functions. Note that the latter specifies the next byte expected, not the last byte correctly received. Both are 32 bits long because every byte of data is numbered in a TCP stream.

The **TCP header length** tells how many 32-bit words are contained in the TCP header. This information is needed because the Options field is of variable length, so the header is, too.

A **Checksum** is also provided for extra reliability. It checksums the header, the data, and the conceptual pseudo header.

UDP Protocol: The User Datagram Protocol is a connection-less and datagram-oriented best-effort protocol. A UDP-communication is analogous to sending a letter. UDP provides a way for applications to send encapsulated IP datagrams and send them without having to establish a connection.

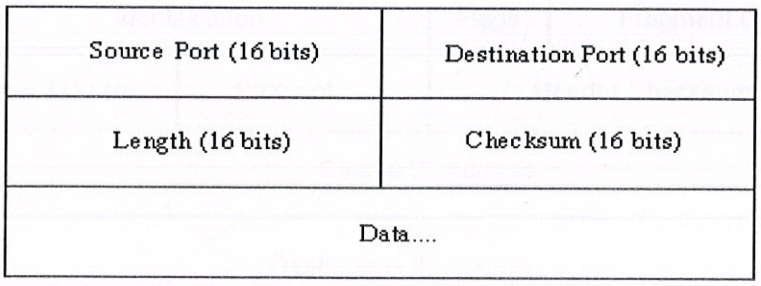


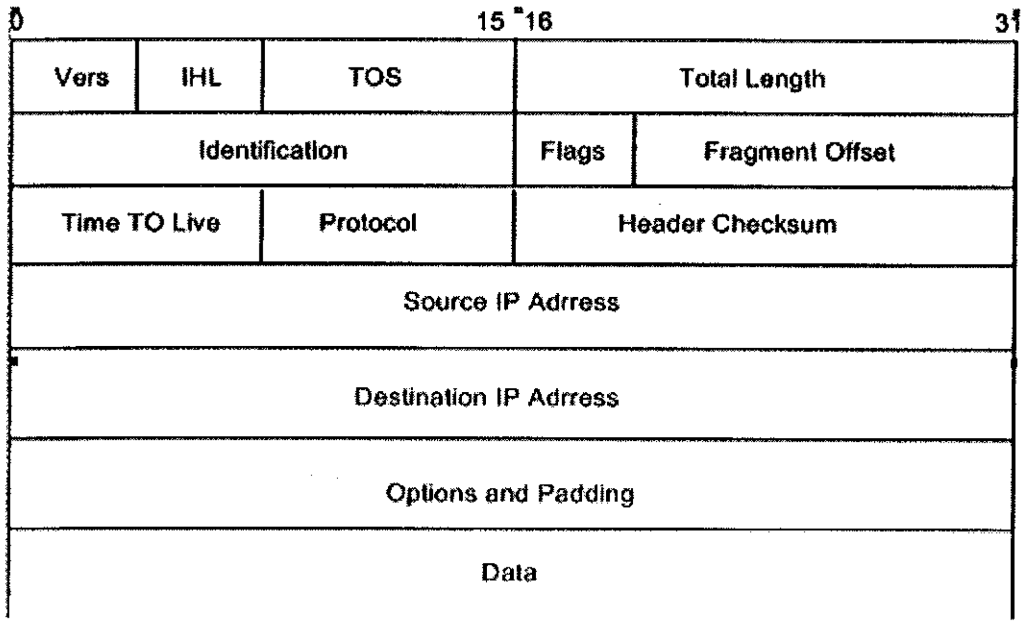
Fig. 9: UDP HEADER

The **source port** is primarily needed when a reply must be sent back to the source. By copying the source port field from the incoming segment into the **destination port** field of the outgoing segment, the process sending the reply can specify which process on the sending machine is to get it.

The **UDP length** field includes the 8-byte header and the data. The **UDP checksum** is optional and stored as 0 if not computed (a true computed 0 is stored as all 1s). Turning it off is foolish unless the quality of the data does not matter

IP Protocol: An IP datagram consists of a header part and a text part. The header has a 20-byte fixed part and a variable length optional part.

The **Version** field keeps track of which version of the protocol the datagram belongs to. By including the version in each datagram, it becomes possible to have the transition between versions take years, with some machines running the old version and others running the new one.



<u>**Fig. 10: TCP Header**</u>

Since the header length is not constant, a field in the header, **IHL**, is provided to tell how long the header is, in 32-bit words.

The **Type of service** field is one of the few fields that have changed its meaning (slightly) over the years. It was and is still intended to distinguish between different classes of service. Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery. For file transfer, error-free transmission is more important than fast transmission.

The **Total length** includes everything in the datagram—both header and data. The maximum length is 65,535 bytes.

The **Identification** field is needed to allow the destination host to determine which datagram a newly arrived fragment belongs to. All the fragments of a datagram contain the same Identification value.

The **Time to live** field is a counter used to limit packet lifetimes. It is supposed to count time in seconds, allowing a maximum lifetime of 255 sec.

The **Source address** and **Destination address** indicate the network number and host number.

Ports:

Generally, a computer has a single connection to the network. If all data arrives through one connection, how can it be determined which application running on the computer receives the data? The answer is through the use of ports.

A port is a 16-bit number in the range or 0 to 65535.The port numbers 0 to 1023 are reserved for special services such as HTTP (port 80), Mail (port 25), and Telnet (port 23).

A connected application must be bound to at least one port. *Binding* means that a port is assigned to a socket used by an application. The application is registered with the system. All incoming packets that contain the port number of the application in the packet header are given to the application socket. If a socket is waiting on a port for an incoming connection, normally the port is blocked for other applications.

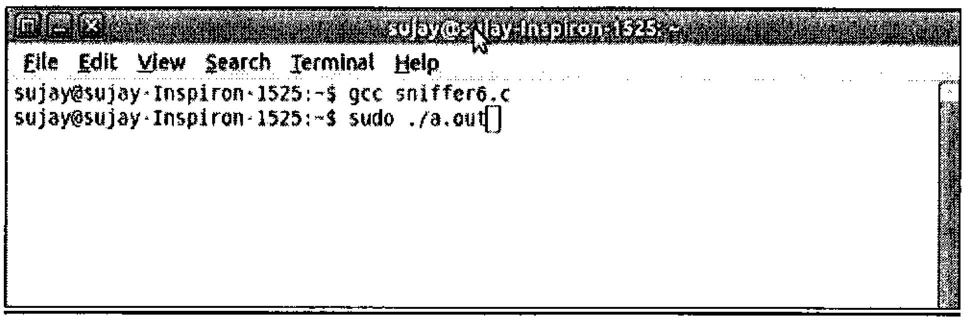
Sockets:

To access IP-based networks from an application, we need *sockets*. A socket is a programming interface and communication endpoint that can be used for connection to other computers, sending and receiving data from them.

Generally, three types of sockets exist:

1) **A raw socket:** This type is implemented on the network layer .An example for a protocol on this layer is IP.
2) **Datagram sockets:** Datagrams are packets of data. This type of sockets is implemented on the transport layer. However, the assignment to a layer is not strict, because, for instance, IP is also datagram-oriented.
3) **Stream sockets:** In contrast to datagram sockets, these sockets provide a stream of data.

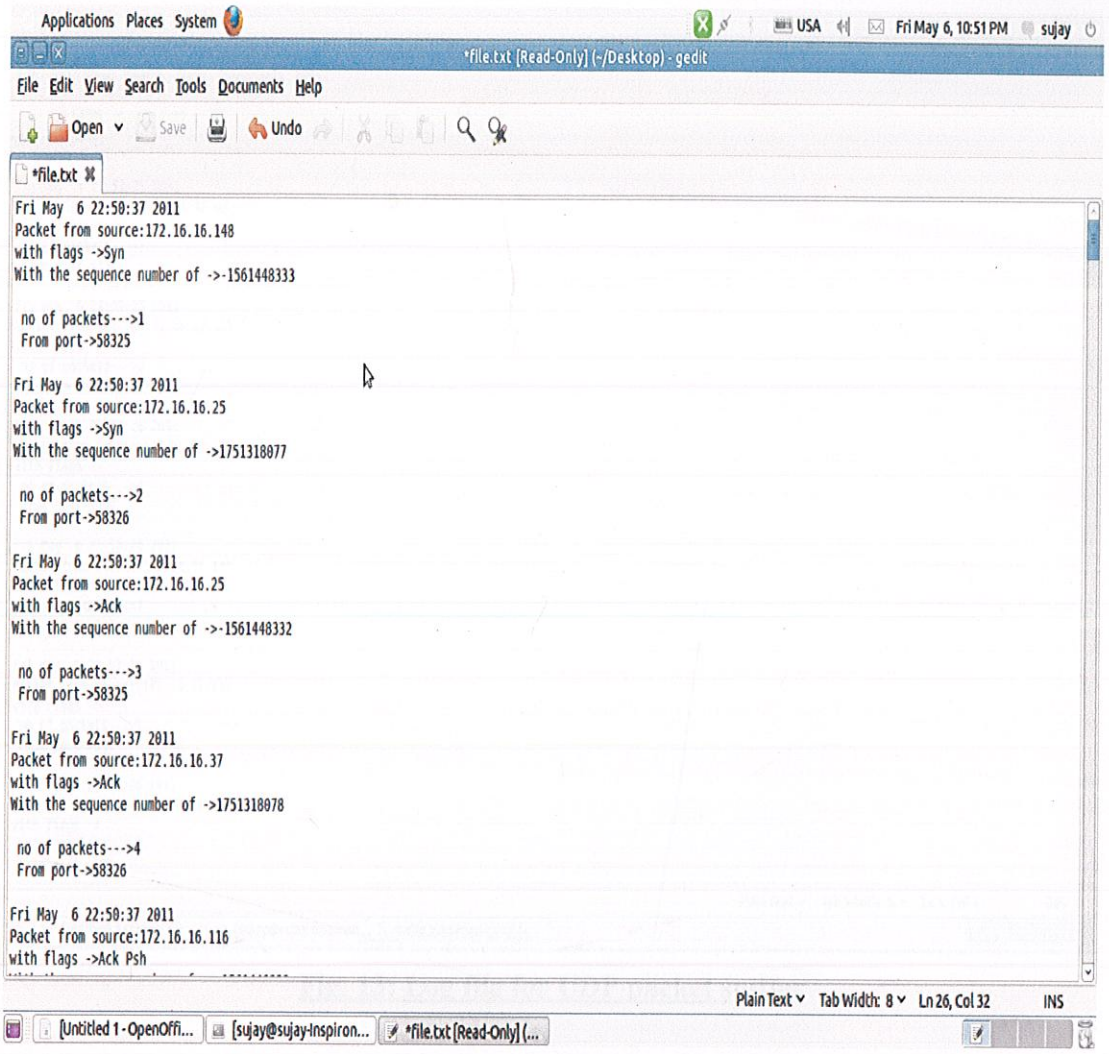
<u>How to use the Packet sniffer software</u>


sujay@sujay-Inspiron-1525: ~
File Edit View Search Terminal Help

```
sujay@sujay-Inspiron-1525:~$ gcc sniffer6.c
sujay@sujay-Inspiron-1525:~$ sudo ./a.out
```

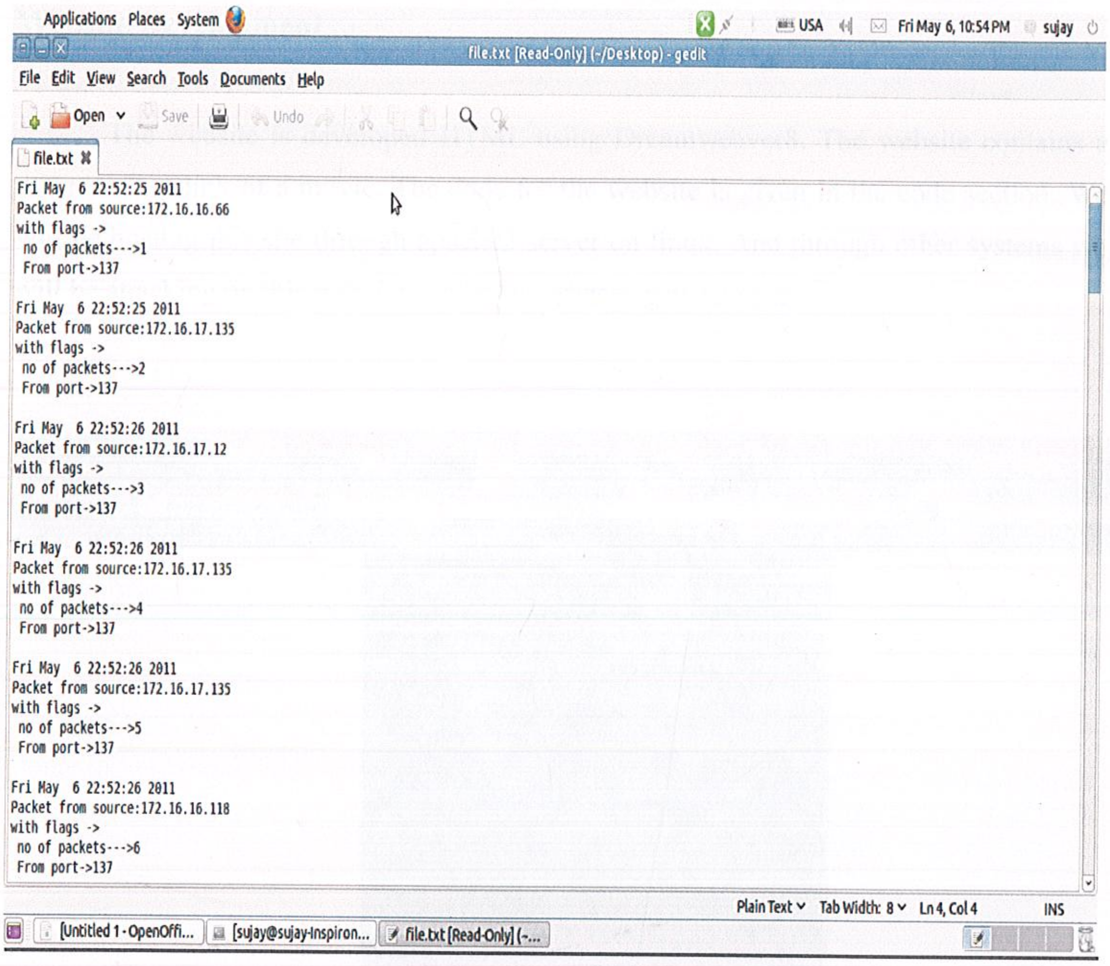
<u>Fig. 11: Linux Terminal</u>

We run the packet sniffer code in gcc compiler with the help of linux terminal. After the program has been executed, then the text file is created with the results so that monitoring of the network traffic on the system be initiated. The screen shot of the files are shown below.


Applications Places System
USA
Fri May 6, 10:51 PM
sujay
*file.txt [Read-Only] (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
*file.txt
Plain Text
Tab Width: 8
Ln 26, Col 32
INS
[Untitled 1 - OpenOffi...
[sujay@sujay-Inspiron...
*file.txt [Read-Only] (...

```
Fri May  6 22:50:37 2011
Packet from source:172.16.16.148
with flags ->Syn
With the sequence number of ->-1561448333

 no of packets--->1
 From port->58325

Fri May  6 22:50:37 2011
Packet from source:172.16.16.25
with flags ->Syn
With the sequence number of ->1751318077

 no of packets--->2
 From port->58326

Fri May  6 22:50:37 2011
Packet from source:172.16.16.25
with flags ->Ack
With the sequence number of ->-1561448332

 no of packets--->3
 From port->58325

Fri May  6 22:50:37 2011
Packet from source:172.16.16.37
with flags ->Ack
With the sequence number of ->1751318078

 no of packets--->4
 From port->58326

Fri May  6 22:50:37 2011
Packet from source:172.16.16.116
with flags ->Ack Psh
```

Fig. 12: Log file for TCP packet sniffer

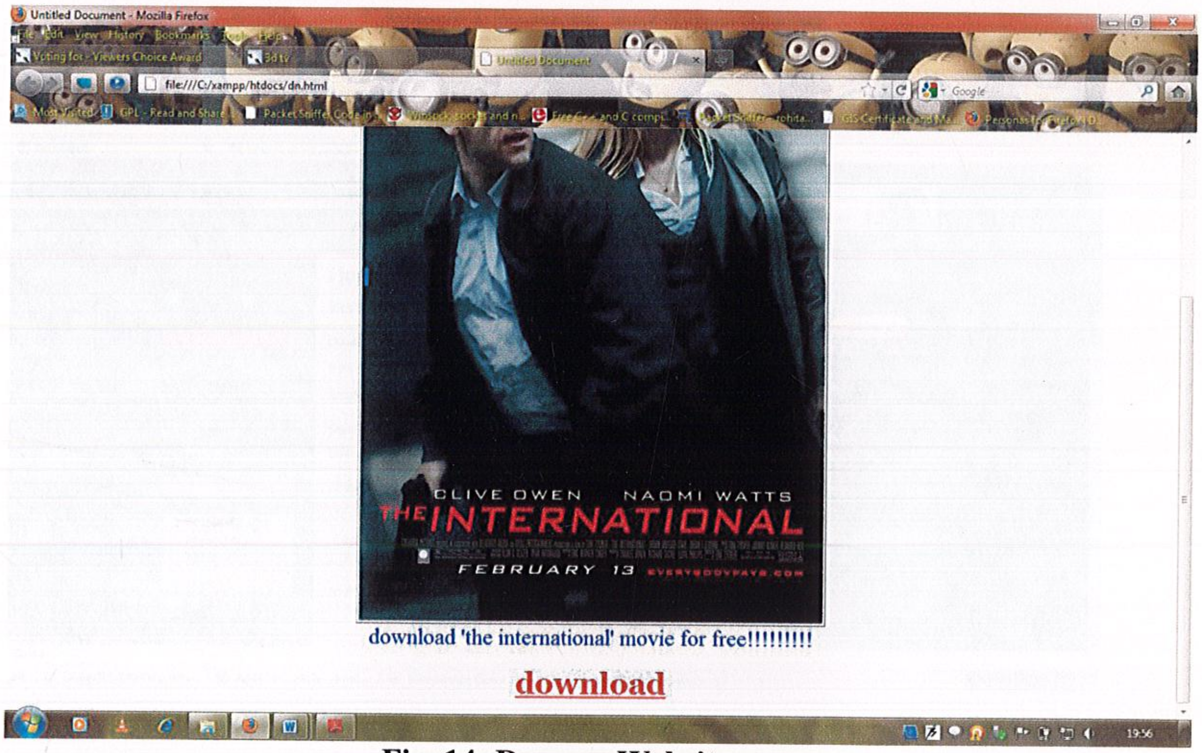

Applications Places System
USA Fri May 6, 10:54 PM sujay
file.txt [Read-Only] (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
file.txt
Plain Text Tab Width: 8 Ln 4, Col 4 INS
[Untitled 1 - OpenOffi... [sujay@sujay-Inspiron... file.txt [Read-Only] (-...

```
Fri May  6 22:52:25 2011
Packet from source:172.16.16.66
with flags ->
 no of packets--->1
 From port->137

Fri May  6 22:52:25 2011
Packet from source:172.16.17.135
with flags ->
 no of packets--->2
 From port->137

Fri May  6 22:52:26 2011
Packet from source:172.16.17.12
with flags ->
 no of packets--->3
 From port->137

Fri May  6 22:52:26 2011
Packet from source:172.16.17.135
with flags ->
 no of packets--->4
 From port->137

Fri May  6 22:52:26 2011
Packet from source:172.16.17.135
with flags ->
 no of packets--->5
 From port->137

Fri May  6 22:52:26 2011
Packet from source:172.16.16.118
with flags ->
 no of packets--->6
 From port->137
```

Fig. 13: Log file for UDP packet sniffer

Website Development

The website is developed HTML using Dreamweaver8. The website contains a downloading link of a movie. The code for the website is given in the code section. We will be hosting this site through apache2 server on linux. And through other systems we will be attacking on this website so that its services gets stopped.



Untitled Document - Mozilla Firefox
file:///C:/xampp/htdocs/dn.html
CLIVE OWEN NAOMI WATTS
THE INTERNATIONAL
FEBRUARY 13
download 'the international' movie for free!!!!!!!!!
download

Fig. 14: Dummy Website

4.2.2.2 Website Hosting

We use Apache Server to host website on Local Area Network . To download apache on linux we run the command given below in terminal.

$ sudo apt-get install apache2

This command will install apache on internet on the system.

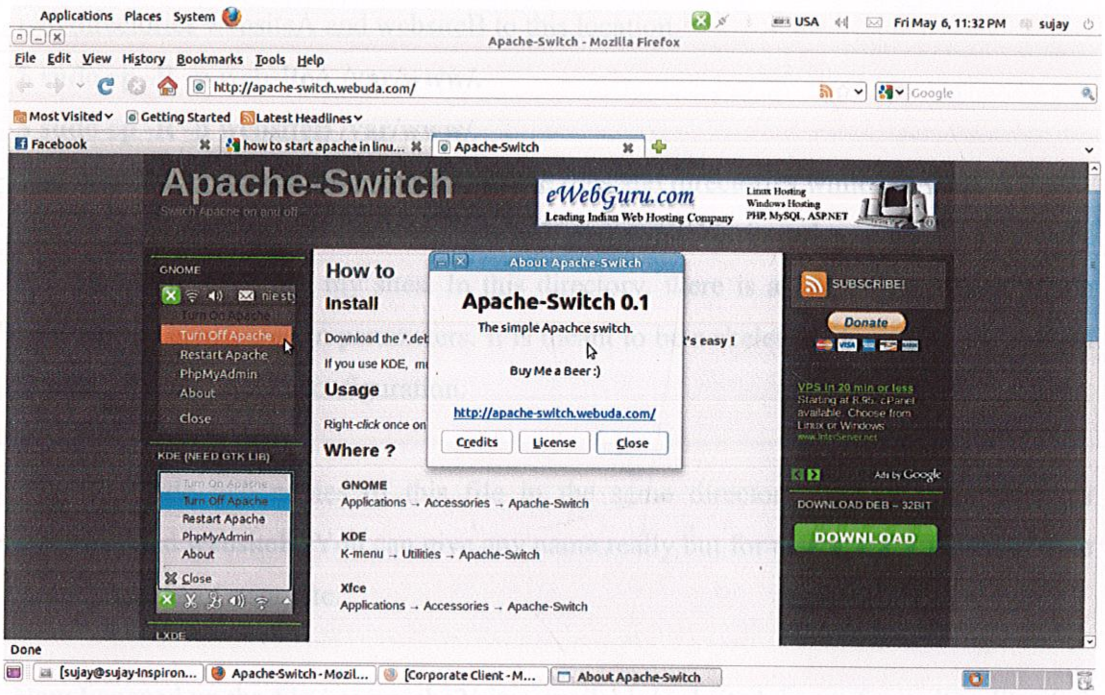

Applications Places System
Fri May 6, 11:32 PM sujay
Apache-Switch - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://apache-switch.webuda.com/
Most Visited Getting Started Latest Headlines
Facebook
how to start apache in linu...
Apache-Switch
Apache-Switch
eWebGuru.com
Leading Indian Web Hosting Company
GNOME
Turn On Apache
Turn Off Apache
Restart Apache
PhpMyAdmin
About
Close
KDE (NEED GTK LIB)
How to Install
Usage
Where ?
About Apache-Switch
Apache-Switch 0.1
The simple Apachce switch.
Buy Me a Beer :)
http://apache-switch.webuda.com/
Credits
License
Close
GNOME
Applications → Accessories → Apache-Switch
KDE
K-menu → Utilities → Apache-Switch
Xfce
Applications → Accessories → Apache-Switch
SUBSCRIBE!
Donate
DOWNLOAD DEB – 32BIT
DOWNLOAD
Done
[sujay@sujay-Inspiron...
Apache-Switch - Mozil...
[Corporate Client - M...
About Apache-Switch

Fig. 14: Apache Switch 0.1

We install apache switch 0.1 from the website as shown in the figure above. After installing this software we can easily start and atop the apache.

Configuring Apache and hosting website

Let us assume for the sake of this tutorial that I have decided to host two websites on my local machine. All the files related to the two websites have already been created and saved in two separate directories by name websiteA and websiteB . The default location for serving the files in apache is usually in the /var/www location. So I move the two directories websiteA and websiteB to this location.

```
$ sudo cp -R -p websiteA /var/www/.
$ sudo cp -R -p websiteB /var/www/.
```

The -p option preserves the ownership of the files and directories while copying.

Next I move into the directory /etc/apache2/sites-available in order to configure apache web server to recognize my sites. In this directory, there is a file called default which contains the configuration parameters. It is meant to be a skeleton file which can be used as a base for additional configuration.

I made two copies of this file in the same directory and renamed them as websiteA and websiteB. You can give any name really but for clarity it is prudent to give them the name of your site.

Now I opened up the file /etc/apache2/sites-available/websiteA in my favourite editor (vi) and made changes to the following portions (shown in bold):

```
#FILE: /etc/apache2/sites-available/websiteA
NameVirtualHost websiteA:80
<virtualhost websiteA:80>
ServerAdmin ravi@localhost
ServerName websiteA
DocumentRoot /var/www/websiteA/
<directory>
Options FollowSymLinks
```

```
AllowOverride None
</directory>
<directory /var/www/websiteA/>
Options Indexes FollowSymLinks MultiViews
AllowOverride None
Order allow,deny
allow from all
...
</directory>
...
...
</virtualhost>
```

In the above listing, ServerName indicates the name (or alias) with which your site has to be recognised. Usually a webserver will be hosting multiple sites with different domain names on the same machine. At such times, the webserver distinguishes between different websites through the ServerName directive. Since this is a local machine, I have given it the name websiteA. But in actual hosting where you want to make the website available on the net, you will be giving it your domain name say like www.mysite.com or something.

The DocumentRoot option denotes which files to be served when the server name is entered in the web browser. This usually points to the place where you have stored your website files.

You can have multiple Directory tags but one of them should point to the location of the website files.

I made similar changes to the file websiteB where the ServerName was given a unique name websiteB instead of websiteA as given above. Also the Directory tag contained the path /var/www/websiteB/ instead of websiteA.

Finally, because I was hosting this on my local machine and since I had not configured DNS, I had to edit the /etc/hosts file and include the name given in the ServerName portion of the configuration file. After the inclusion, my machine's /etc/hosts file looked as follows:

```
#FILE : /etc/hosts
127.0.0.1 localhost.localdomain localhost websiteA websiteB
```

That was it. Now in order to enable the websites, all I had to do was create a symbolic link to these files in the /etc/apache2/sites-enabled/ directory and restart the web server. It is quite clear that to disable a website, all you need to do is remove the symlink in the sites-enabled directory.

```
$ cd /etc/apache2/sites-enabled
$ sudo ln -s /etc/apache2/sites-available/websiteA .
$ sudo ln -s /etc/apache2/sites-available/websiteB .
```

Restart the apache web server

```
$ sudo apache2ctl restart
OR
$ sudo /etc/init.d/apache2 restart
```

Now I tested it by opening up the web browser and typing http://websiteA to get the first website and http://websiteB to get the second website.

4.3 Hardware and Software Requirements

Hardware Requirements

1. 6 PC's or workstations with Pentium IV processor or above
2. Each PC having atleast 1 Gb of RAM.
3. 6 LAN wires – CAT 5 cables.

Software Requirements

1. **LOIC v1.0.4.0** software– to launch the DDoS attack.
2. OS - Windows XP SP2 and Linux.
3. Packet Sniffer
4. Dreamweaver 8
5. Apache2

4.4 Working Mechanism and Data Capture Levels

Consider a scenario where a server of a network had been providing services through some port say 56600. The organization now plans to update its network structure and now switches over to port 56601 to provide the same services. The organization members now know that this upgrading has been done and so no services shall be provided by port 56600.

We now deploy a honeypot at the older port 56600. Now any traffic coming to this port will certainly be an attack or scan by someone outside the organization who still does not know about the organization's upgradation and wants to break into the main server thinking older port still to be the server port.

1. Now let us suppose a DDoS attack has been launched to the old port by an attacker who assumes this port to be the server port.
2. First of all the firewall will be configured in such a manner that it will allow all the traffic into the honeypot system.
3. The installed sniffer will then capture the detailed information of all the packets that flood the honeypot. This will form the most important level of interaction and will provide most of the information about the hackers operations.
4. All these logs will be copied to the remote system using a remote logging software from time to time.
5. Flush technique will be used where all the incoming traffic will be allowed and blocked from time to time, logs copied to the remote system and honeypot resources is released.

The remote system will send these logs to the main server for analysis and taking appropriate action to add security to its network.

4.5 Software Development Life Cycle

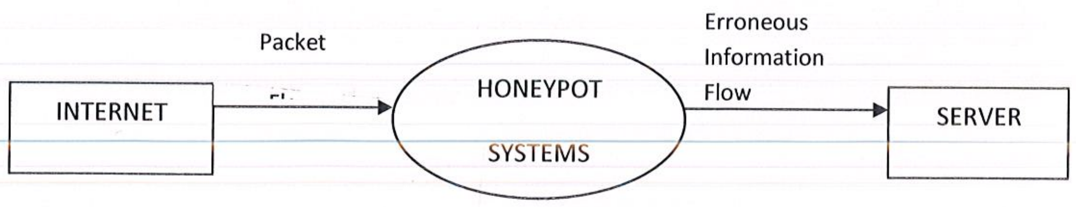
In the analysis phase of the SDLC, we start with the analysis of the problem statement and then the detailed understanding of the objective and the goals of the problem was carried out. For understanding of the problem, the basic requirement was the detailed study of the major topics related to our project. The topics which required such deep study were DDoS, IDS and of course Honeypot systems. The issues that were equally dealt with were virus, worms, spy ware, phishing and firewalls.

In the designing phase of the life cycle, a number of available designs were studied and hence with proper modifications in some the best of them, we came out with the most feasible one.

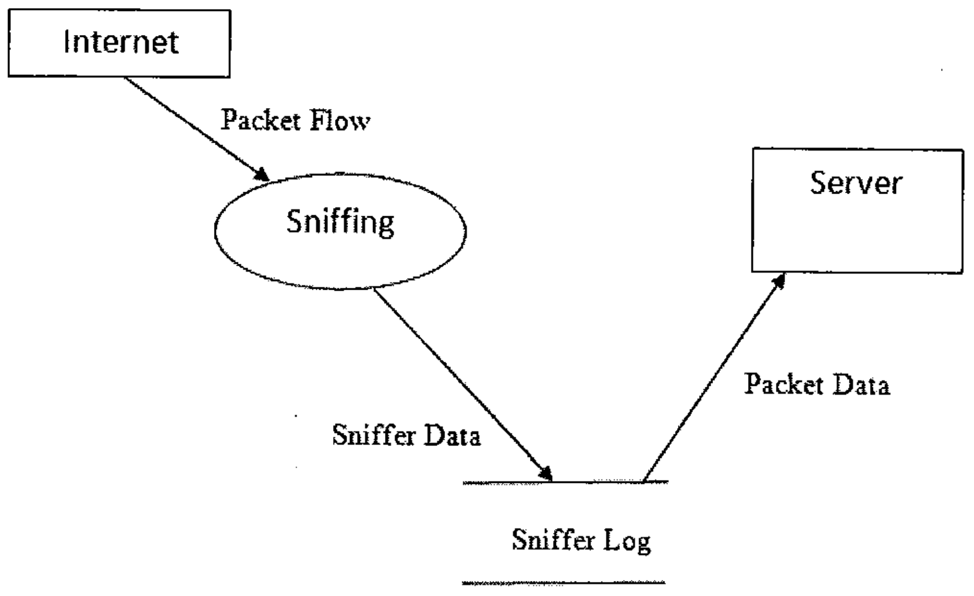
In the coding phase, packet sniffer has been coded using C language which is in accordance with the design developed in the previous phase. In the testing phase, packet sniffer software is tested using white box testing.

4.6 Data Flow Diagrams

LEVEL 0 DFD

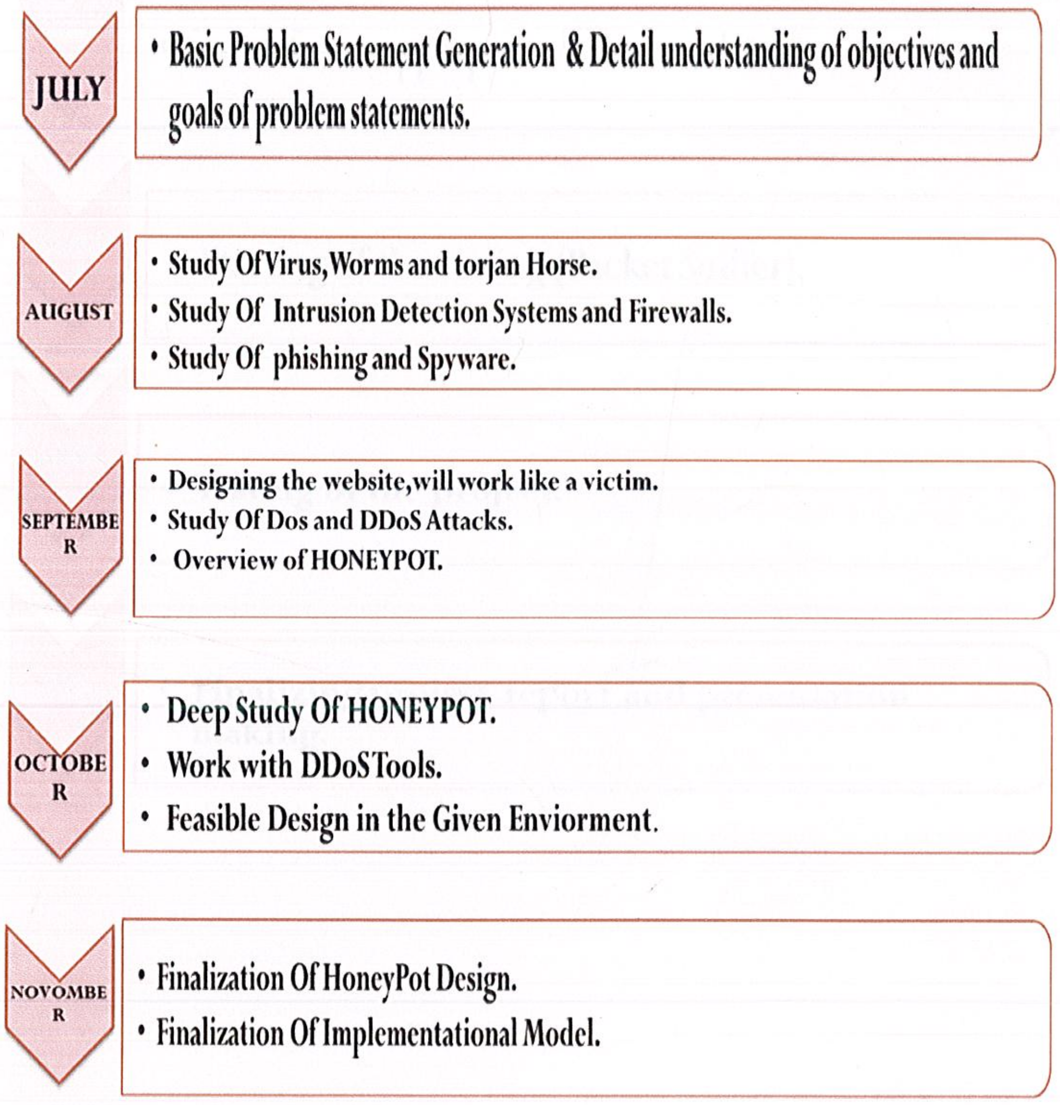


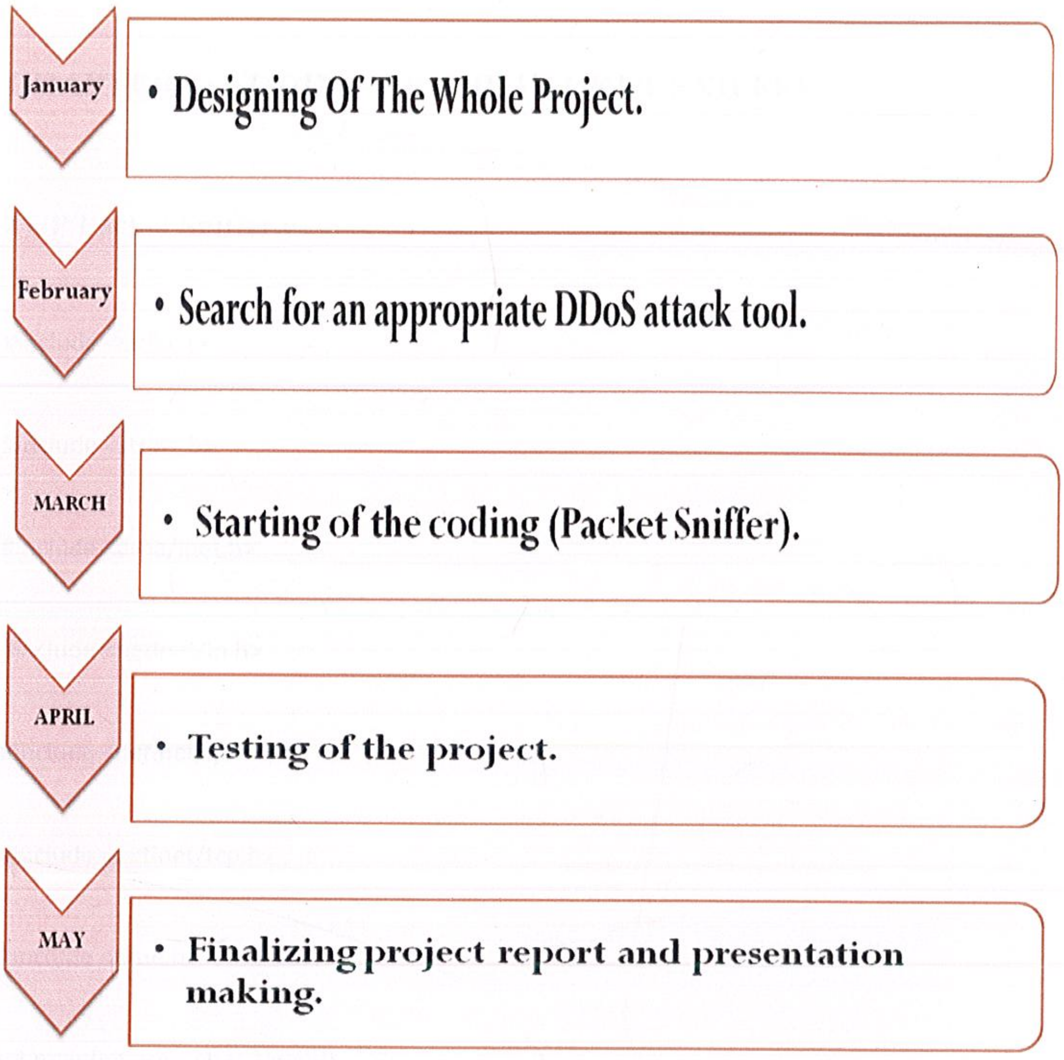
Level 1 DFD:


Internet
Packet Flow
Sniffing
Server
Packet Data
Sniffer Data
Sniffer Log

4.7 Project Development Chart

ROADMAP


JULY
• Basic Problem Statement Generation & Detail understanding of objectives and goals of problem statements.
AUGUST
• Study Of Virus, Worms and torjan Horse.
• Study Of Intrusion Detection Systems and Firewalls.
• Study Of phishing and Spyware.
SEPTEMBE R
• Designing the website,will work like a victim.
• Study Of Dos and DDoS Attacks.
• Overview of HONEYPOT.
OCTOBE R
• Deep Study Of HONEYPOT.
• Work with DDoSTools.
• Feasible Design in the Given Enviorment.
NOVOMBE R
• Finalization Of HoneyPot Design.
• Finalization Of Implementational Model.



January
- Designing Of The Whole Project.

February
- Search for an appropriate DDoS attack tool.

MARCH
- Starting of the coding (Packet Sniffer).

APRIL
- Testing of the project.

MAY
- Finalizing project report and presentation making.

CHAPTER 5: CODING OF THE PACKET SNIFFER

TCP Packet Sniffer.c

```
#include <stdio.h>

#include <string.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#include <netinet/ip.h>

#include <netinet/tcp.h>

#include <time.h>

int main(int argc, char *argv[])

{

int s, bytes,a=0,sy=0,f=0,r=0,u=0,p=0,dell=0;

int ac=0,syc=0,fc=0,rc=0;

time_t t;

struct tcphdr *tcp;
struct iphdr*ip;

struct in_addr addr;

char buffer[4000];

s = socket(AF_INET, SOCK_RAW, IPPROTO_TCP);

if (s == -1)

{

perror("socket() failed");

return 1;

}

ip = (struct iphdr*) buffer;

tcp = (struct tcphdr*) (buffer + sizeof(struct iphdr));

while( (bytes = recv(s, buffer, sizeof(buffer), 0)) > 0)

{

FILE *file=fopen("/home/sujay/Desktop/file.txt","a");

if(file)
```

```
{

time(&t);

fprintf(file,"%s",ctime(&t));

addr.s_addr = ip->saddr;

a=ntohs(tcp->ack);

sy=ntohs(tcp->syn);

r=ntohs(tcp->rst);

f=ntohs(tcp->fin);

p=ntohs(tcp->psh);

u=ntohs(tcp->urg);

if (ip->saddr!=inet_addr("192.168.0.113"))

{

fprintf(file,"Packet from source:%s\nwith flags ->",inet_ntoa(addr));

if(a==256)

{
```

```
fprintf(file,"Ack ");

}

if ( sy==256)

{

fprintf(file,"Syn ");

;}

if (f==256)

{

fprintf(file,"Fin ");

}

if (r==256)

{

fprintf(file,"Rst ");

}

if (p==256)
```

```
{
fprintf(file,"Psh ");
}
if (u==256)
{
fprintf(file,"Urg");
}
fprintf(file,"\n");
fprintf(file,"With the sequence number of ->%i\n",ntohl(tcp->seq));
fprintf(file,"\n no of packets--->%d",++dell);
fprintf(file,"\n From port->%i\n\n",ntohs(tcp->source));
}
fclose(file);
}
}
```

```
if (bytes == -1)

{

perror("recv() failed");

return 2;

}

return 0;

}
```

UDP Packet Sniffer.c

```
#include <stdio.h>

#include <string.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#include <netinet/ip.h>

#include <netinet/tcp.h>

#include <netinet/udp.h>
```

```
#include <time.h>

int main(int argc, char *argv[])

{

int s, bytes,a=0,sy=0,f=0,r=0,u=0,p=0,dell=0;

int ac=0,syc=0,fc=0,rc=0;

time_t t;

struct tcphdr *tcp;

struct udphdr *udp;

struct iphdr*ip;

struct in_addr addr;

char buffer[4000];

s = socket(AF_INET, SOCK_RAW, IPPROTO_UDP);

if (s == -1)

{

perror("socket() failed");
```

```
return 1;

}

ip = (struct iphdr*) buffer;

udp = (struct udphdr*) (buffer + sizeof(struct iphdr));

while( (bytes = recv(s, buffer, sizeof(buffer), 0)) > 0)

{

FILE *file=fopen("/home/sujay/Desktop/file.txt","a");

if(file)

{

time(&t);

fprintf(file,"%s",ctime(&t));

addr.s_addr = ip->saddr;

if (ip->saddr!=inet_addr("192.168.0.113"))

{

fprintf(file,"Packet from source:%s\nwith flags ->",inet_ntoa(addr));
fprintf(file,"\n no of packets--->%d",++dell);

fprintf(file,"\n From port->%i\n\n",ntohs(udp->source));

}

fclose(file);

}

}

if (bytes == -1)

{

perror("recv() failed");

return 2;

}

return 0;

}
```

Website Code .html

Website has been created in dreamweaver and the code generated is given below:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
<style type="text/css">
<!--
.style1 {
	font-size: xx-large;
	font-weight: bold;
	font-family: Georgia, "Times New Roman", Times, serif;
	color: #FF0000;
}
.style2 {
	color: #0000FF;
	font-size: 24px;
}
-->
</style>
</head>
```

```
<body>
<div align="center">
  <table width="507" height="757" border="1">
    <tr>
      <td background="international_ver3.jpg" bgcolor="#000000">&nbsp;</td>
    </tr>
  </table>
  <span class="style2">download 'the
  international' movie for free!!!!!!!!!  </span>
  <p><a href="The International.avi" class="style1">download</a> </p>
</div>
</body>
</html>
```

CHAPTER 6: TESTING

The design that has been proposed for our honeypot system consists of a tool for deployment of DDoS attack, a firewall to monitor the incoming traffic, a sniffer for the packet information logs and a machine that acts as a remote system containing the remote logger.

White box testing has been implemented to test the design of the system which includes a close examination of procedural details i.e. the collaboration and compatibility between components using various test cases. Hence in this project it has been seen that whether all the modules that have been mentioned above are working as desired. The basic purpose was to test the correctness of the module placement and data flow between them.

First a switched Ethernet LAN was created which consisted of 5 machines in total. An efficient DDoS tool called **Low Orbit Ion Cannon** ver. 1.0.4.0 was installed on 6 machines to simulate a DDoS attack on the target machine running a Http based Web Service. For this purpose **Apache2** software to create a local server which will host the local web service for a LAN and hence the target of the DDoS attack. The local service was created using html and css applications.

On this machine, which was our honeypot, a sniffer was also installed which monitored the network traffic and a firewall which would stop and start the network traffic as and when required. After all the modules were assembled, the input to packet sniffer was given which was the IP address of the host machine. The firewall was configured to allow all the traffic to pass through.

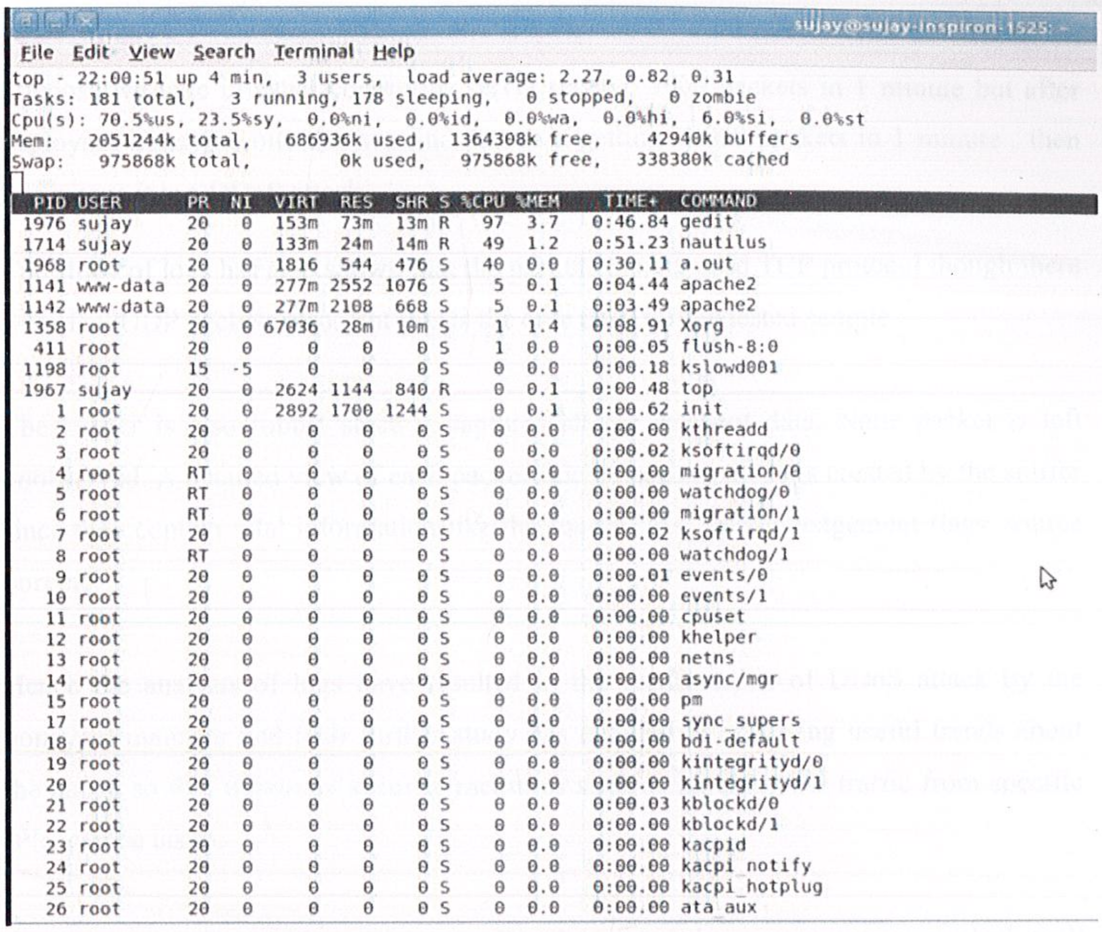
After all the software were installed and configured on all the machines, attack was launched on the host machine using the other three machines with 200 threads each requesting connection to the web service. But the CPU usage of the victim machine was not affected to large extent as expected. So, to increase the CPU usage a more powerful attack was launched consisting of 400 connection requests by each attacking machine. But still result was not satisfactory, so an attack of 600 connection request thread was launched which was powerful enough to exhaust the network bandwidth and increased the CPU usage of victim to 60-70%. Since the network traffic was almost congested the web service provided by target machine was not accessible on the client machine which showed that the web service on host machine was under Distributed Denial of Service attack. The data was then analyzed to obtain important information about the DDoS attack that was launched.

CHAPTER 7: RESULT AND ANALYSIS

The motive of this project was to design a real time honeypot to track one of the most dangerous network attacks – the Distributed Denial of Services. A lot of qualitative study has been done to fulfill the desired objective. This also includes carrying out a research paper suggesting a completely new design of the honeypot. The results that have been obtained are simple yet impressive and overall satisfactory.

The attack to the target machine or honeypot was launched by 6 machines using different no. of server request threads like 200, 300, 400 and 600. But result was obtained with 600 request connections only from each pc.

The target machine's CPU usage showed 70% of processor usage. During this period no service from this machine, which acted as a local server, could be granted to the clients since it became busy answering to a flood of requests by attacking machines thereby giving a quality proof of a **successful DDoS attack**. The snapshot of CPU performance of server under DDoS attack is shown below:



sujay@sujay-inspiron-1525: ~
File Edit View Search Terminal Help

```
top - 22:00:51 up 4 min,  3 users,  load average: 2.27, 0.82, 0.31
Tasks: 181 total,   3 running, 178 sleeping,   0 stopped,   0 zombie
Cpu(s): 70.5%us, 23.5%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  6.0%si,  0.0%st
Mem:   2051244k total,   686936k used,  1364308k free,    42940k buffers
Swap:   975868k total,        0k used,   975868k free,   338380k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1976 sujay     20   0  153m  73m  13m R   97  3.7   0:46.84 gedit
 1714 sujay     20   0  133m  24m  14m R   49  1.2   0:51.23 nautilus
 1968 root      20   0  1816  544  476 S   40  0.0   0:30.11 a.out
 1141 www-data  20   0  277m 2552 1076 S    5  0.1   0:04.44 apache2
 1142 www-data  20   0  277m 2108  668 S    5  0.1   0:03.49 apache2
 1358 root      20   0 67036  28m  10m S    1  1.4   0:08.91 Xorg
  411 root      20   0     0    0    0 S    1  0.0   0:00.05 flush-8:0
 1198 root      15  -5     0    0    0 S    0  0.0   0:00.18 kslowd001
 1967 sujay     20   0  2624 1144  840 R    0  0.1   0:00.48 top
    1 root      20   0  2892 1700 1244 S    0  0.1   0:00.62 init
    2 root      20   0     0    0    0 S    0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S    0  0.0   0:00.02 ksoftirqd/0
    4 root      RT   0     0    0    0 S    0  0.0   0:00.00 migration/0
    5 root      RT   0     0    0    0 S    0  0.0   0:00.00 watchdog/0
    6 root      RT   0     0    0    0 S    0  0.0   0:00.00 migration/1
    7 root      20   0     0    0    0 S    0  0.0   0:00.02 ksoftirqd/1
    8 root      RT   0     0    0    0 S    0  0.0   0:00.00 watchdog/1
    9 root      20   0     0    0    0 S    0  0.0   0:00.01 events/0
   10 root      20   0     0    0    0 S    0  0.0   0:00.00 events/1
   11 root      20   0     0    0    0 S    0  0.0   0:00.00 cpuset
   12 root      20   0     0    0    0 S    0  0.0   0:00.00 khelper
   13 root      20   0     0    0    0 S    0  0.0   0:00.00 netns
   14 root      20   0     0    0    0 S    0  0.0   0:00.00 async/mgr
   15 root      20   0     0    0    0 S    0  0.0   0:00.00 pm
   17 root      20   0     0    0    0 S    0  0.0   0:00.00 sync_supers
   18 root      20   0     0    0    0 S    0  0.0   0:00.00 bdi-default
   19 root      20   0     0    0    0 S    0  0.0   0:00.00 kintegrityd/0
   20 root      20   0     0    0    0 S    0  0.0   0:00.00 kintegrityd/1
   21 root      20   0     0    0    0 S    0  0.0   0:00.03 kblockd/0
   22 root      20   0     0    0    0 S    0  0.0   0:00.00 kblockd/1
   23 root      20   0     0    0    0 S    0  0.0   0:00.00 kacpid
   24 root      20   0     0    0    0 S    0  0.0   0:00.00 kacpi_notify
   25 root      20   0     0    0    0 S    0  0.0   0:00.00 kacpi_hotplug
   26 root      20   0     0    0    0 S    0  0.0   0:00.00 ata_aux
```

Fig: CPU performance under DDoS attack

It can be clearly seen that under a successful DDoS attack the CPU usage attained a very high value. Hence even when the no. of processes running in the server was a normal , a successful DDoS attack was launched which congested or flooded the server thereby denying all its services to legitimate users.

The log files were created at ease and the data about different attackers is gained. The log file contains time stamp for each entry so that the number of packets received in small interval of time can be calculated. If the number of packets received is much more than the usual then it could be DDoS attack.

Suppose we take in usual conditions server receive 3000 packets in 1 minute but after deploying honeypot one day we find server is getting 20000 packets in 1 minute , then this means it is a DDoS attack.

The study of logs has also shown that the packets mostly used TCP protocol though there were few UDP packets also. But this is the case only for the tested sample.

The sniffer is also robust since it captures all the relevant data. None packet is left unobserved. A detailed view of each packet can be seen in the logs created by the sniffer since they contain vital information like the source IP's, acknowledgement flags, source port etc.

Hence the analysis of logs have resulted in the confirmation of DDoS attack by the honeypot manager and their further study has resulted in observing useful trends about the attack so that important security measures such as blocking the traffic from specific IP's, can be taken.

Finally it can be said that the complete integrated honeypot system worked well providing enough data for a detailed analysis.

CHAPTER 8: FUTURE SCOPE

The working and results of honeypot and associated modules have been quite satisfactory. The sniffer gives very detailed and efficient logs. Also the DDoS tool generates a strong DDoS attack. The honeypot can be used in any real situation to track DDoS and other kind of attacks.

As the next step we plan to:

- Formalize the design and verify its correctness under specific types of attacks.
- Analyze the performance of honeypot in terms of overhead introduced into the network. This includes the scalability and optimizing performance using different mechanisms.
- Analyze the effectiveness of honeypot in terms of detecting DDoS attacks.

The proposed design can be further improved to obtain a better, useful and more informative results by increasing the level of interaction or data capture by installing a software code to monitor the keystrokes and login information as provided by the hacker. The design can also be made innovative by bringing out slight changes in its organization like keeping the remote system in a different network from that of a server, which may have few advantages over the proposed design. Honeypots are our best shot for staying equal with the malicious hacker community and perhaps even moving ahead. A honeypot is often the best computer security defense tool for the job. As an early warning system, it can alert you when all the other security defenses have failed.

CHAPTER 9: APPENDIX

9.1 Appendix 1: List of Abbreviations

1. DoS: Denial of Service
2. DDoS: Distributed Denial of Service
3. IDS: Intrusion Detection System
4. DNS: Domain Name System
5. IRC: Internet Relay Chat
6. FTP: File Transfer Protocol
7. URL: Uniform Resource Locator
8. HTTP: Hyper Text Transfer Protocol
9. IP: Internet Protocol
10. UDP: User Datagram Protocol
11. TCP: Transmission Control Protocol
12. SYN: Synchronous
13. ACK: Acknowledge
14. TCB: Transmission Control Block
15. ISP: Internet Service Provider
16. TFN: Tribe Flood Network
17. ICMP: Internet Control Message Protocol
18. TFN2K: Tribe Flood Network 2000
19. HIDS: Host Based Intrusion Detection System
20. NIDS: Network Based Intrusion Detection System
21. NNIDS: Network Node Intrusion Detection System
22. dIDS: distributed Intrusion Detection System
23. MAC: Media Access Control
24. DHCP: Dynamic Host Configuration Protocol
25. DFD: Data Flow Diagram

CHAPTER 10: REFERENCES

[1] Internet Denial of Service: Attack and Defense Mechanisms
By Jelena Mirkovic, Sven Dietrich, David Dittrich, Peter Reiher

[2] Honeypots: Tracking Hackers
By Lance Spitzner

[3] Know Your Enemy
http://project.honeynet.org

[4] Snort is an Open Source Intrusion Detection System,
http://www.snort.org

[5] CERT
http://www.cert.org

[6] A website fully dedicated on DDoS
http://www.staff.washington.edu/dittrich/misc/ddos/

[7] The Joys of DDoS
By Barrett Lyon and Jay Adelson

[8] Honeypots for DDoS
By NathalieWeiler

[9] Defending Against DDoS
By Tao Peng

[10] An Integrated Honeypot Framework for Proactive Detection, Characterization and Redirection of DDoS Attacks at ISP level
Anjali Sardana and R.´C. Joshi

[11] Honeypot Scheme for Distributed Denial-of-Service Attack
Vinu V Das

[12] Lance Spitzner Definitions and Value of Honeypots.
http://www.trackinghackers.com/papers/honeypots.html

[13] http://ieeexplrore.ieee.org

[14] Honeynet tutorial-Honeypots –Advantages and disadvantages
http://homes.cerias.purdue.edu/~kaw/research/honeynet/honeynettutorial/honeypots