Detection of Distributed Denial of Service Attack (DDoS) Against a Web Server

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

COMPUTER SCIENCE & ENGINEERING

Under the Supervision of

MR. AMOL VASUDEVA

by

MUDIT SRIVASTAVA (091274)
UJJWAL THAKUR (091284)
UTKARSH AGNIHOTRI (091290)

to

Jaypee University of Information Technology

Waknaghat, Solan – 173234, Himachal Pradesh

CERTIFICATE

This is to certify that the work titled "**Detection of Distributed Denial of Service Attack (DDoS) Against a Web Server"** submitted by "**Mudit Srivastava, Ujjwal Thakur and Utkarsh Agnihotri**" in partial fulfillment for the award of degree of B.Tech in CSE from Jaypee University of Information Technology, Waknaghat 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.

Signature of Supervisor	....
Name of Supervisor	Mr. Amol Vasudeva
Designation	Sr. Lecturer
Date	….........................

ACKNOWLEDGEMENT

With candor and pleasure we take this opportunity to express our sincere gratitude to our esteemed guide **Mr. Amol Vasudeva.** Without his exemplary guidance and co-operation it would not have been possible for us to reach the completion of this project.

It is our pleasant duty to thank all the staff members of the **Department of C.S.E. & I.C.T.**, especially our **H.O.D.,Prof. Dr. S.P. Ghrera** and our supervisor and project coordinator **Dr. Nitin** who never hesitated from helping us during the course of this project.

Finally, we gratefully acknowledge the support, encouragement & patience of our families and friends.

Date:

Mudit Srivastava (091274)

Ujjwal Thakur (091284)

Utkarsh Agnihotri (091290)

LIST OF FIGURES

TABLE OF CONTENTS

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.

1.2 Key Issues and Challenges

In order to develop an affective DDoS detection mechanism we must have a firewall in place. The issue at hand is the development of a rule based packet filtering proxy gateway. It also has to be configured correctly to gather relevant data about the incoming traffic to allow the detection mechanism to work correctly. The design of the firewall is a major challenge as the entire project is coded in Java and there are no native APIs in Java which allow for efficient capturing of data packets.

One problem with the use of honeypot systems is the issue of liability. Intentionally placing an 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.

1.3 Statement of the Problem

Design of a proxy gateway and implementation of a Source IP Address Monitoring Algorithm to detect a 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 proxy gateway and implementation of a detection algorithm to detect the DDoS attack while it is taking place.
3. Use of Honeypots to study how intruders try to compromise a system.

CHAPTER 2: BACKGROUND STUDY

2.1 Classification of attacks

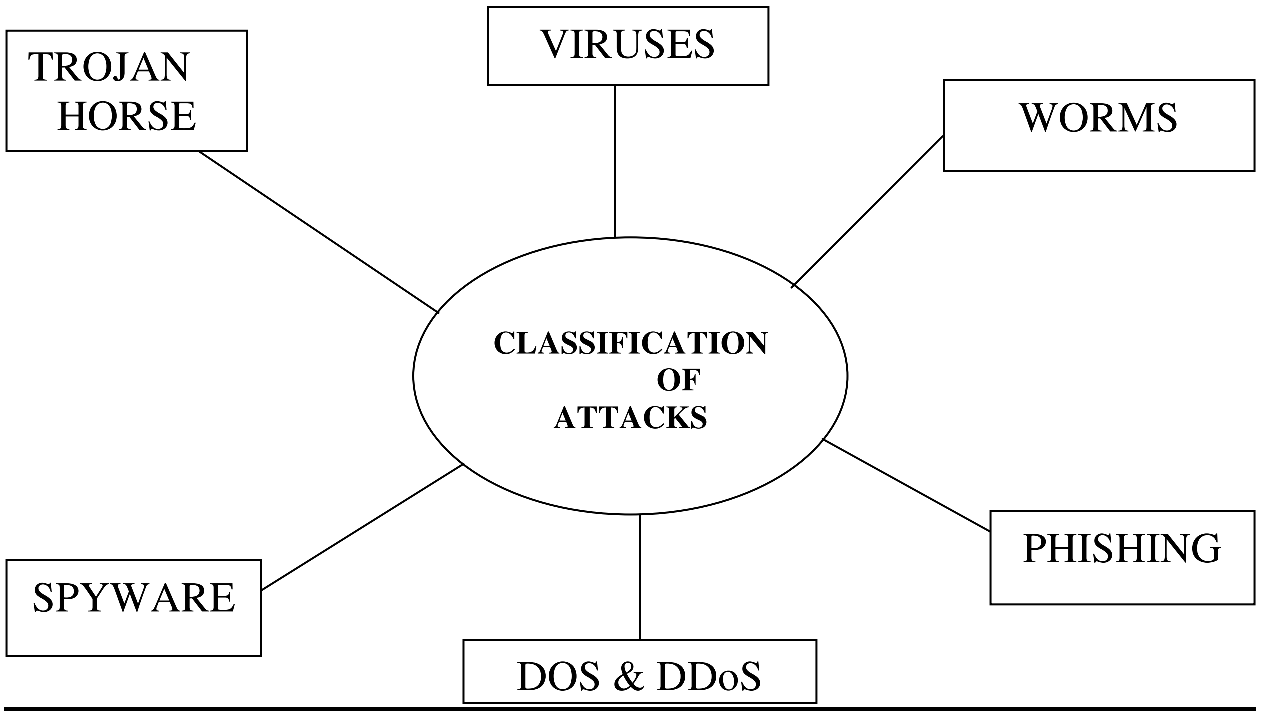


Fig2- 1Classification of Attcaks

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.

Protecting against dangerous computer worms

Worms spread by exploiting vulnerabilities in operating systems. All vendors supply regular security updates and if these are installed to a machine then the majority of worms are unable to spread to it. If a vendor acknowledges vulnerability but has yet to release a security update to patch it, a zero day exploit is possible. However, these are relatively rare.

Users need to be wary of opening unexpected email, and should not run attached files or programs, or visit web sites that are linked to such emails. However, as with the ILOVEYOU worm, and with the increased growth and efficiency of phishing attacks, it remains possible to trick the end-user into running a malicious code. Anti-virus and anti-spyware software are helpful, but must be kept up-to-date with new pattern files at least every few days. The use of a firewall is also recommended.

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.

2.2.1DoS 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.

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.

. 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.

2.2.2.1Recruiting 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 2.2 illustrates the **Handler/Agent** architecture:

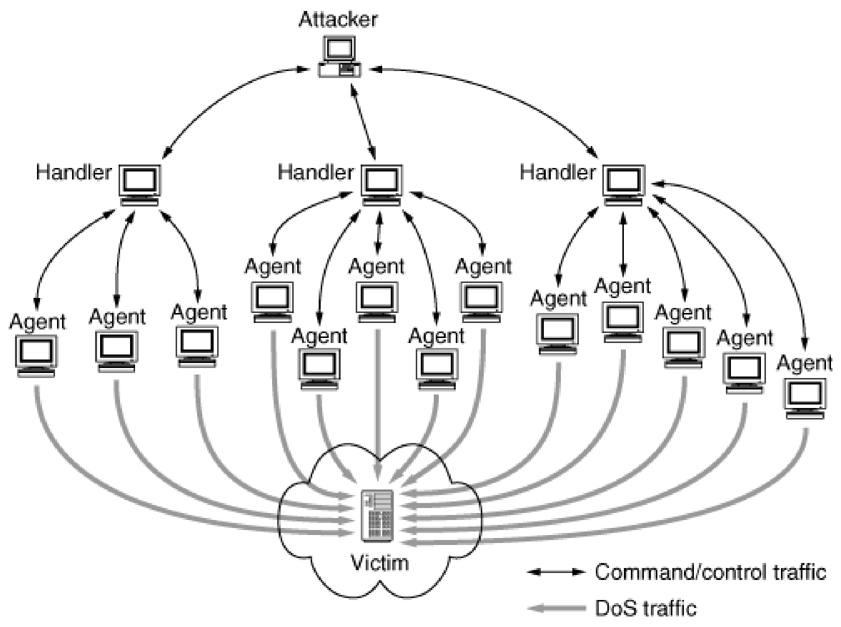


Fig2- 2 Handler/Agent architecture

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 figure2.3:

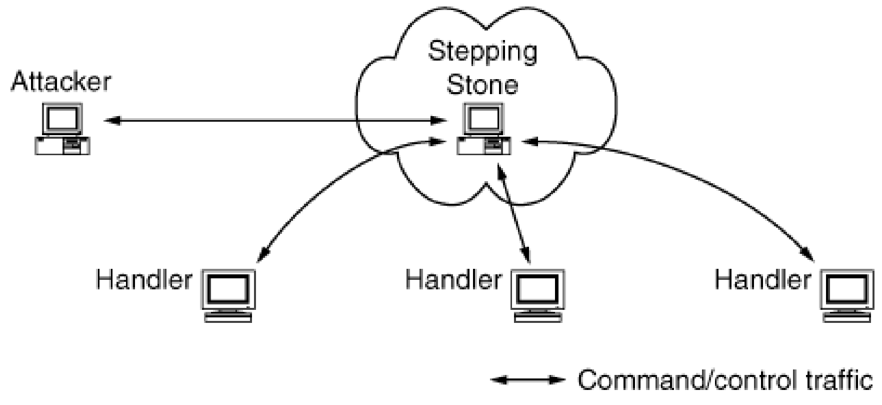


Fig2- 3Stepping stone

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/he 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 2.4 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.

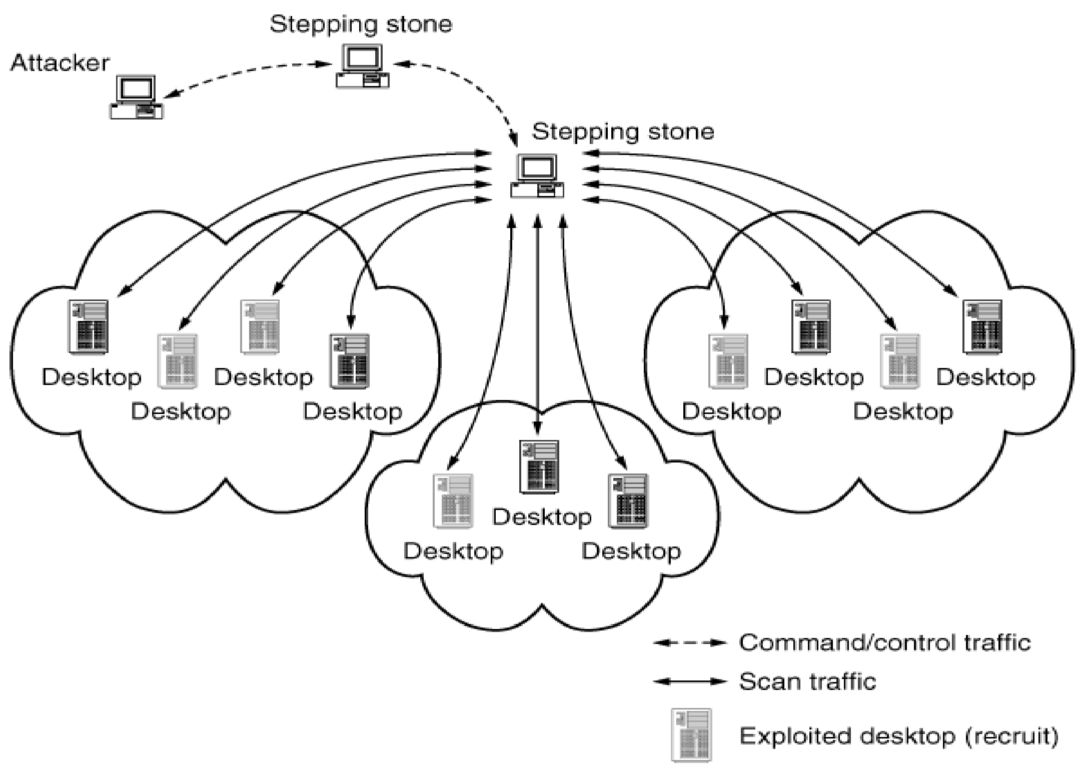


Fig2- 4 Simple Scanning Process

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. Figure 2.5 illustrates the worm propagation:

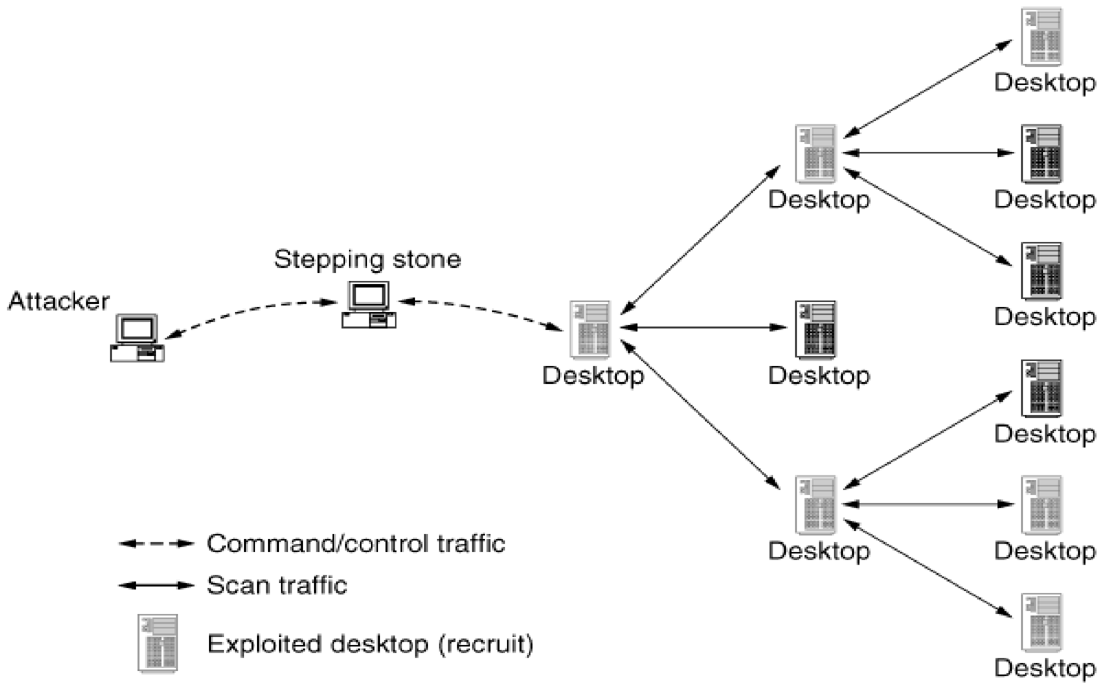


Fig2- 5 Worm Propagation

- **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.

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. Figure below illustrates propagation with central repository:

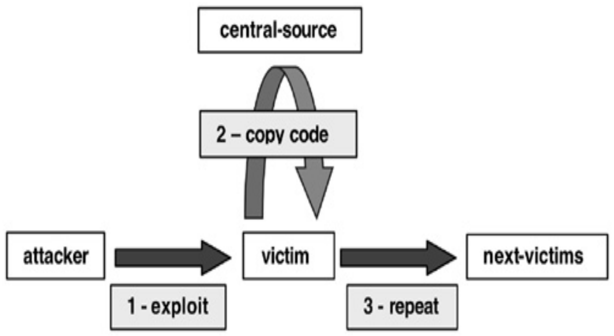


Fig2- 6 Propagation with central repository

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. Figure below illustrates propagation with back-chaining:

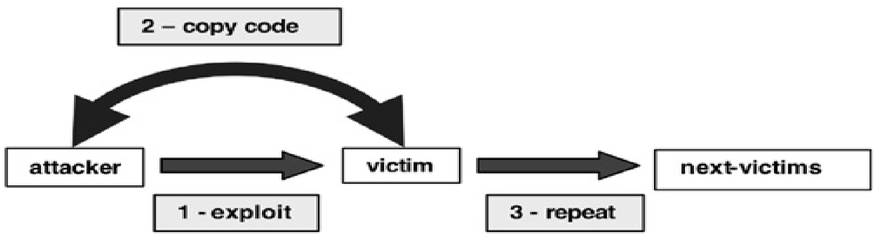


Fig2- 7 Propagation with back-chaining

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. Figure 2.8 illustrates autonomous propagation:

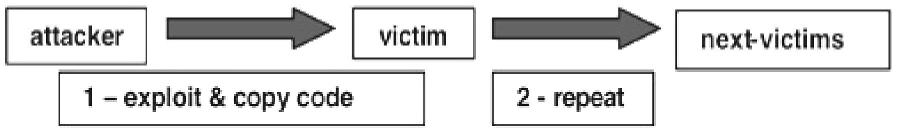


Fig2- 8 Autonomous Propagation

<u>2.2.4 Semantic Levels of DDoS Attacks</u>

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 2.9:

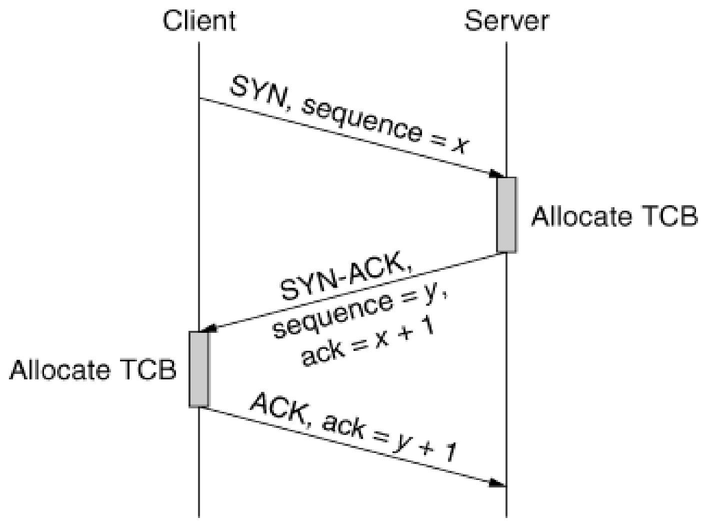


Fig2- 9 Three-way Handshake

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.

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

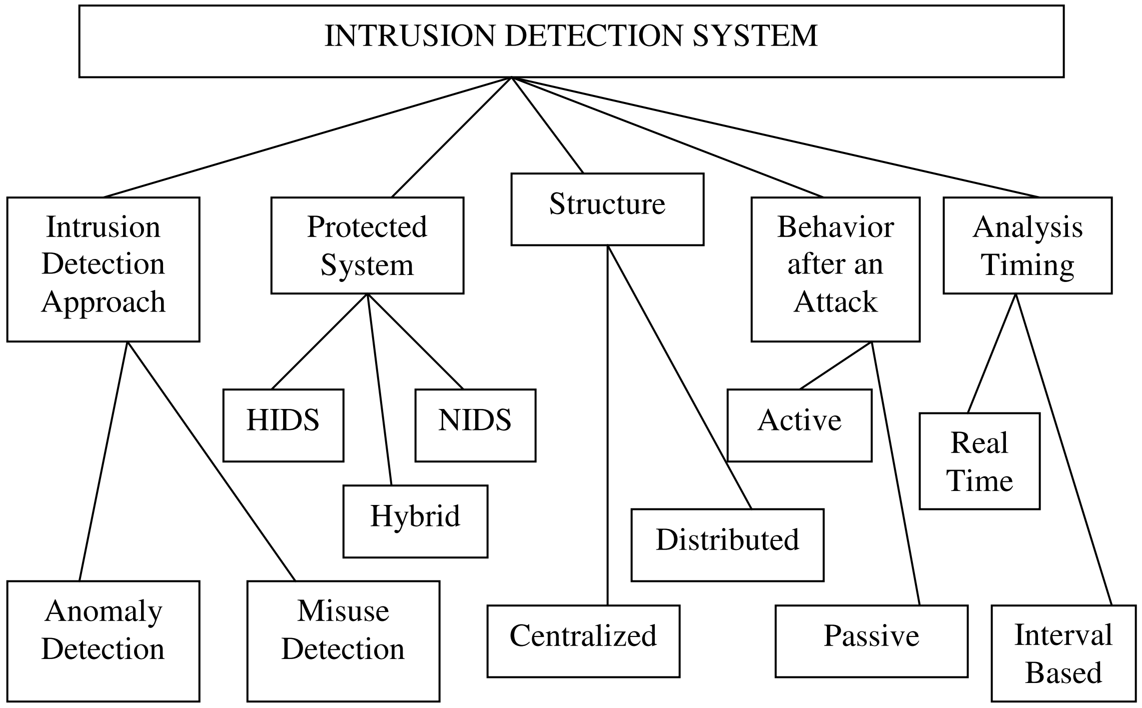


Fig2- 10 Classification of IDS

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.

2.4 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.

3.2 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.

2.6.1 Types of Firewalls

There are three common types of firewalls:

A.)Packet-Filtering Router

A packet-filtering router makes a permit/deny decision for each packet that it receives. The router examines each datagram to determine whether it matches one of its packet-filtering rules. The filtering rules are based on the packet header information that is made available to the IP forwarding process. This information consists of the IP source address, the IP destination address, the encapsulated protocol (TCP, UDP, ICMP, or IP Tunnel), the TCP/UDP source port, the TCP/UDP destination port, the ICMP message type, the incoming interface of the packet, and the outgoing interface of the packet. If a match is found and the rule permits the packet, the packet is forwarded according to the information in the routing table. If a match is found and the rule denies the packet, the packet is discarded. If there is no matching rule, a user-configurable default parameter determines whether the packet is forwarded or discarded

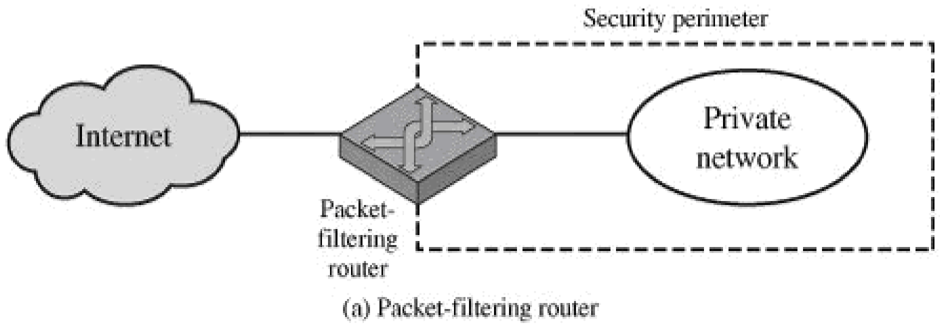


Fig2- 11 Packet-filtering Router

Some typical filtering rules include:

- Permit incoming Telnet sessions only to a specific list of internal hosts
- Permit incoming FTP sessions only to specific internal hosts
- Permit all outbound Telnet sessions
- Permit all outbound FTP sessions
- Deny all incoming traffic from specific external networks

B.) Application-level gateways

An application-level gateway, also called a proxy server, acts as a relay of application-level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall. Further, the gateway can be configured to support only specific features of an application that the network administrator considers acceptable while denying all other features.

Application-level gateways tend to be more secure than packet filters. Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the TCP and IP level, the application-level gateway need only scrutinize a few allowable applications. In addition, it is easy to log and audit all incoming traffic at the application level.

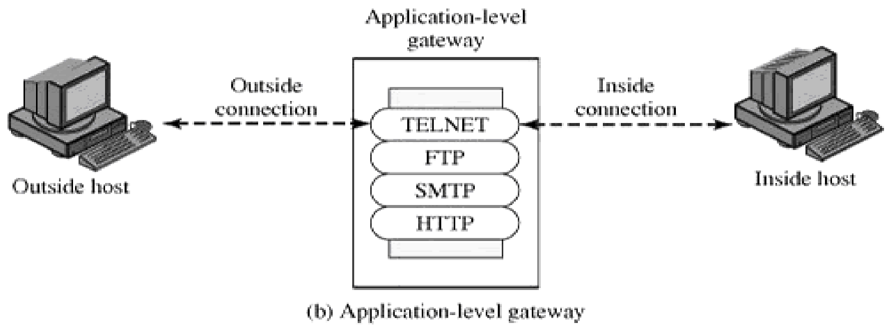


Fig2- 12 Application-level Gateway

A prime disadvantage of this type of gateway is the additional processing overhead on each connection. In effect, there are two spliced connections between the end users, with the gateway at the splice point, and the gateway must examine and forward all traffic in both directions.

C.) Circuit-level gateways

A third type of firewall is the circuit-level gateway .This can be a stand-alone system or it can be a specialized function performed by an application-level gateway for certain applications. A circuit-level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

A typical use of circuit-level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application-level or proxy service on inbound connections and circuit-level functions for outbound connections. In this configuration, the gateway can incur the processing overhead of examining incoming application data for forbidden functions but does not incur that overhead on outgoing data.

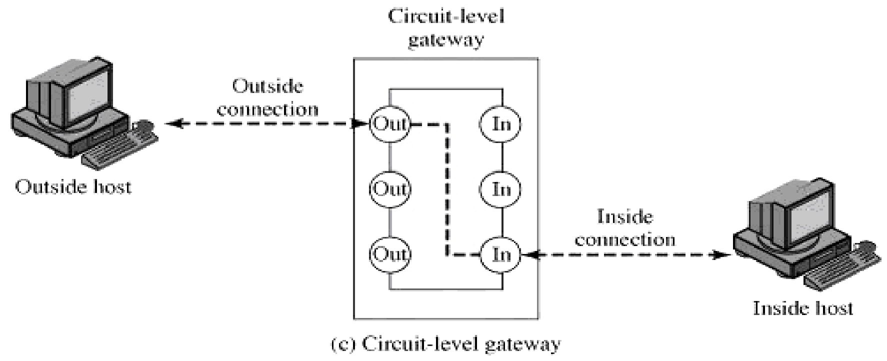


Fig2- 13 Circuit-level Gateway

<u>2.6.2 Applications of Firewall</u>

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

2.6.3 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.

2.6.4 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.

CHAPTER 3: HONEYPOT SYSTEM

3.1 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.

Honey pots 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.

(Note: Discussed in Detail in later Chapter 2.5)

3.2 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.

3.3 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.

3.4 Classification of Honeypots

Honeypots are broadly classified via three methods:

1.Based on implementation

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**.**

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.**

2. 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.

3. 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

Fig3- 1 Low interaction and High interaction Honeypot

3.5 Placement of Honeypots

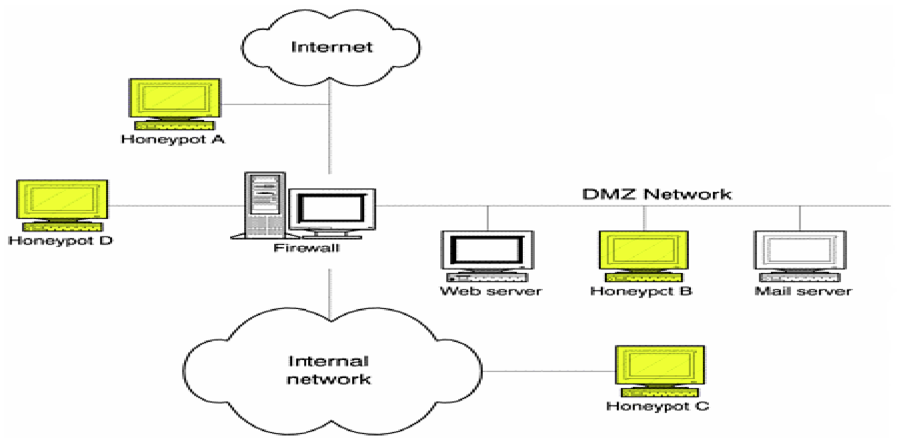


Fig3- 2 Placement of Honeypot

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.

3.6 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 4: DETECTION ALGORITHM

At the heart of the system lies our DDoS detection algorithm. Without a detection mechanism in place, we can only demonstrate a live DDoS and the affect that it has over a server. This has been explored before and so implementation of an algorithm was very crucial for our project. In the following section we give a brief overview of the various ways of detecting a DDoS and the ones that we have used to do so.

4.1 Detection Mechanisms

There are a number of mechanisms which have been developed to detect a Distributed Denial of Service Attack. Each of these is based on a unique technique such as monitoring of all packets which arrive at a particular node, analyzing the data contents and header values of a packet, traffic flow analysis, use of traffic fingerprints to detect an abnormal spike in packet volume, etc. We describe the various mechanisms which we analyzed briefly.

4.1.1 Packet Data Analysis

In this approach all the data packet which are destined for the server are scrutinized at the firewall before they are routed to the internal network. The data part of the packet is analyzed to check if its contents are those which match the services provided by the server. Thus a rule based packet filtering firewall is required to implement an algorithm based on this approach.

This detection algorithm works best against flooding attacks which send data packets with garbage values. However, it has a serious drawback. It fails miserably against attack which is carried out using legitimate traffic.

An attacker could very easily bring down a server using hundreds of zombies each of which will establish multiple connections with the server and pose as legitimate clients. Since the network traffic will consist of data packets whose contents are "legitimate" according to rules defined earlier, they will be allowed through. The server will be bogged down responding to zombies while not even realizing that a DDoS has been launched against it.

4.1.2 Use of Traffic Fingerprints

In this approach the traffic data is constantly monitored. A number of different parameters are measured such as the number of data packets which arrive at a port, the time in which they arrive, the protocol that they were based on, the order in which they arrive, etc. This technique makes use of complex mathematical functions to arrive at the fingerprint of the network traffic and then compares them with a large database of previously defined fingerprints. This database contains fingerprint of previously known attacks.

This approach has a number of drawbacks:

- It requires frequent update of the fingerprint database to correctly detect an attack.
- The network traffic needs to be monitored extensively and this adds consider overhead.
- A new attack is almost always missed by the detection mechanism as there doesn't exist a fingerprint in the database which is same as the fingerprint of the attack.
- For attacks which take place over long durations and are distributed over thousands of machines the detection mechanism fails miserably. This is because during the beginning stages the attack traffic is hidden under normal traffic and so detection fails.

4.1.3 Traffic Flow Analysis

In this mechanism, the traffic flow is analyzed and important parameters such as the originating IP Address of each packet, the traffic volume which

4.2 Our Approach

Of the numerous algorithms available we have used Source IP Address Monitoring algorithm developed by Tao Peng, Christopher Leckie and Kotagiri Ramamohanarao.

It is a traffic flow analysis algorithm and the parts of it which we have implemented are detailed below.

The proposes scheme is called Source IP address Monitoring(SIM) to detect the Distributed Denial of Service. It uses huge number IP addresses in the attack traffic to the victim.

SIM contains two parts: off-line training, and detection and learning. In off-line training a learning engine adds legitimate IP addresses (IAD) and keeps the IAD updated by adding new legitimate IP address and deleting expired IP addresses. A simple rule can be used to decide whether a new IP address is legitimate or not.

Second part is detection and learning. During this period , we collect several statistics of incoming traffic for the current time interval. In our detection engine, a hash table is used to record the IP addresses that appeared in the current time interval. Every hash table entry contains two fields, the number of IP packets and the time stamp of the most recent packet for that IP address, which is illustrated in the following diagram.

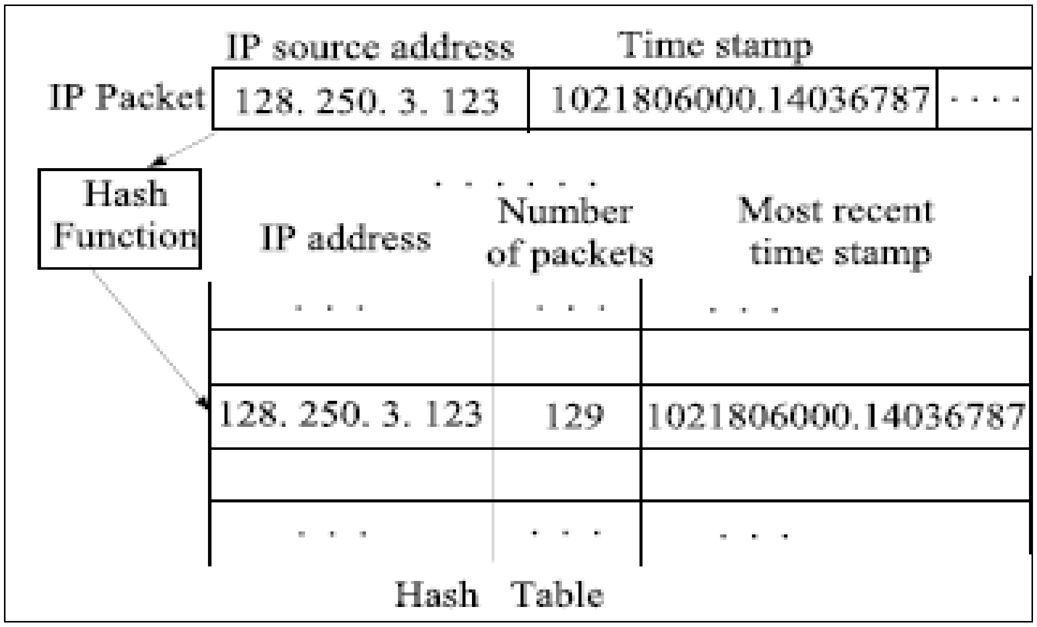


Fig4- 1 Hash Table for Detection Engines

Let A_{normal}, A_{flash}, A_{attack} represent the number of packets, and B_{normal}, B_{flash}, B_{attack} represent the number of new IP addresses for *normal traffic conditions*, *flash crowds* and *DDoS* attacks respectively in a certain time period Δn. Now we have made the following assumptions:

- $A_{normal} << A_{flash} \approx A_{attack}$
- $B_{normal} << B_{flash} \approx B_{attack}$

For a *typical DDoS* attack, since the attack traffic from each "zombie" is randomly spoofed, almost every packet will be new to the monitoring point. Hence B_{attack} will be large.

4.2.1 The detection feature

We collect the IP addresses during each time slot Δ_n (*n=1, 2, 3, ...*), which determines the detection resolution. We assume $\Delta_1 = \Delta_2 = ::: = \Delta_n$, which means the time slots are of equal length. The choice of Δ_n is a compromise between making Δ_n small so that the detection engine can quickly detect an attack, and making Δ_n large so that the detection engine has less computation load because it checks the traffic less often.

Let T_n represent the set of unique IP addresses and D_n represent the items of IP Address Database (IAD) at the end of the time interval Δ_n ($n = 1; 2; 3; :::$). As has been discussed before, $|T_n - T_n \cap D_n|$, which represents the number of new IP addresses in Δ_n, is used to detect the DDoS attack.

This is the proposed architecture of our algorithm.

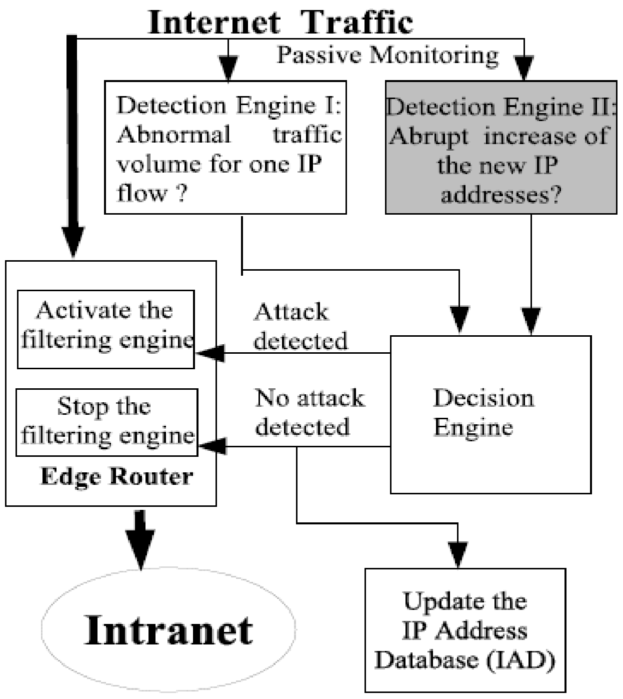

Internet Traffic
Passive Monitoring
Detection Engine I: Abnormal traffic volume for one IP flow ?
Detection Engine II: Abrupt increase of the new IP addresses?
Activate the filtering engine
Stop the filtering engine
Edge Router
Attack detected
No attack detected
Decision Engine
Intranet
Update the IP Address Database (IAD)

Fig4- 2 DDos attack detection and history-based IP source address filtering

CHAPTER 5: PROPOSED DESIGN AND IMPLEMENTATION

5.1 Proposed Design of Project

There are three main modules in this project. Each of this corresponds to a specific functionality which is required for the running of the system. We needed a robust firewall, a light weight server based application and a low interaction honeypot system.

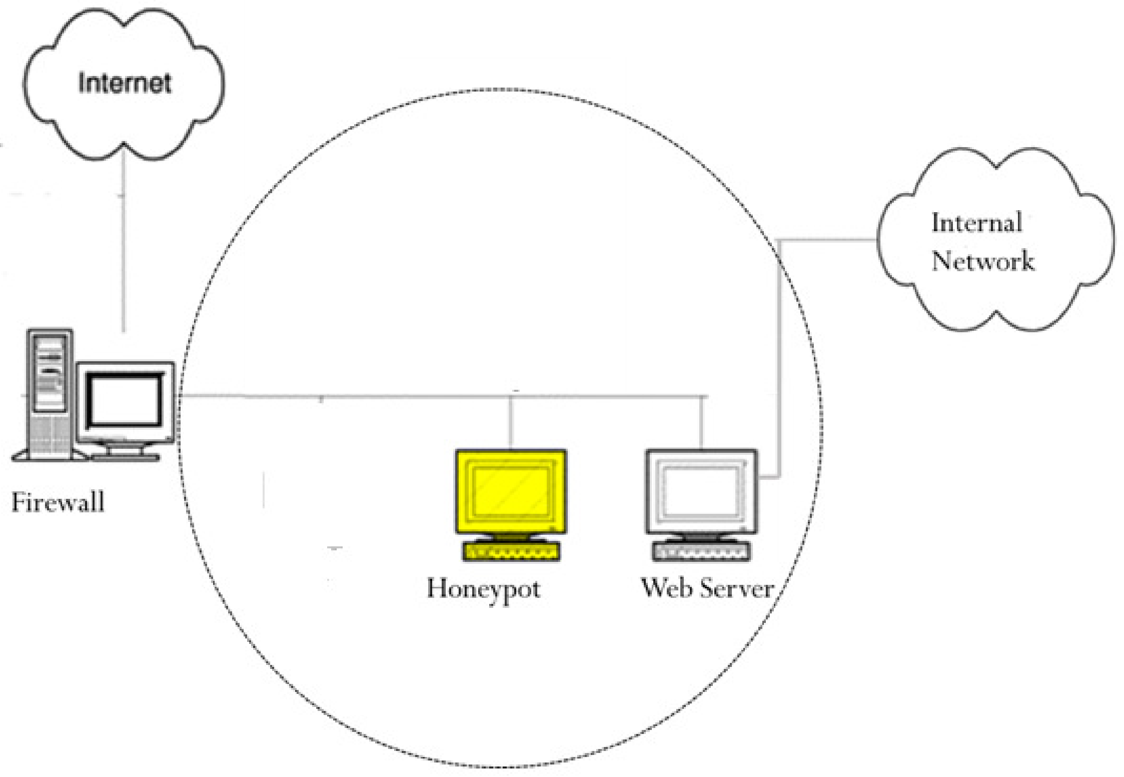


Fig5- 1 Proposed design of Web Server system for detection of DDoS attacks

<u>5.1.1 Firewall</u>

A firewall has been introduced to insure that all outgoing traffic was generated from the web server and is a response to a legit request made by the remote client. This ensures that just in case the system is compromised, it cannot be used to launch attacks against other computers.

A very unique feature of the firewall is that it allows all incoming traffic but restricts the outgoing traffic.

The firewall runs on a computer other than a honeypot system and is keeps a record of all incoming requests by storing the IP Address of each incoming packet in an array.

It then matches the IP Address of all outgoing packets to the list of stored Addresses and rejects all those which haven't been mapped before.

<u>5.1.2 Server</u>

We need a simple server based light application. The reason for choosing a simple light weight application is that we will have to launch a massive attack to disrupt the communication of our server application.

For this purpose, we have used as simple chat room application. In this system, multiple clients connect to the server which runs at a particular port and IP Address. It listens to the message sent by each of the application and the broadcasts the simple message to the remaining clients.

The program running on server end makes minimal use of the underlying processor power and network resources available to it and so is perfect for studying a how a Distributed Denial of Service Attack affects the functioning of a networked application.

<u>5.1.3 Honeypot</u>

The honeypot we intend to use is a low interaction port monitoring device. It emulates a telnet service and runs at port 23 (just like a real telnet service). It has been added to study what steps does an intrudertake to exploit vulnerability in a system. The honeypot records the key strokes which an attacker makes.

It allows the attacker to log in using the user-name, password pair guest, guest or user, user and then doesn't respond to any of the command issued to it. Any skilled attacker will very easily determine that the telnet service is fake and will log out but even in that short duration the honeypot would have traced the attacker's movements.

5.2 Modular Design

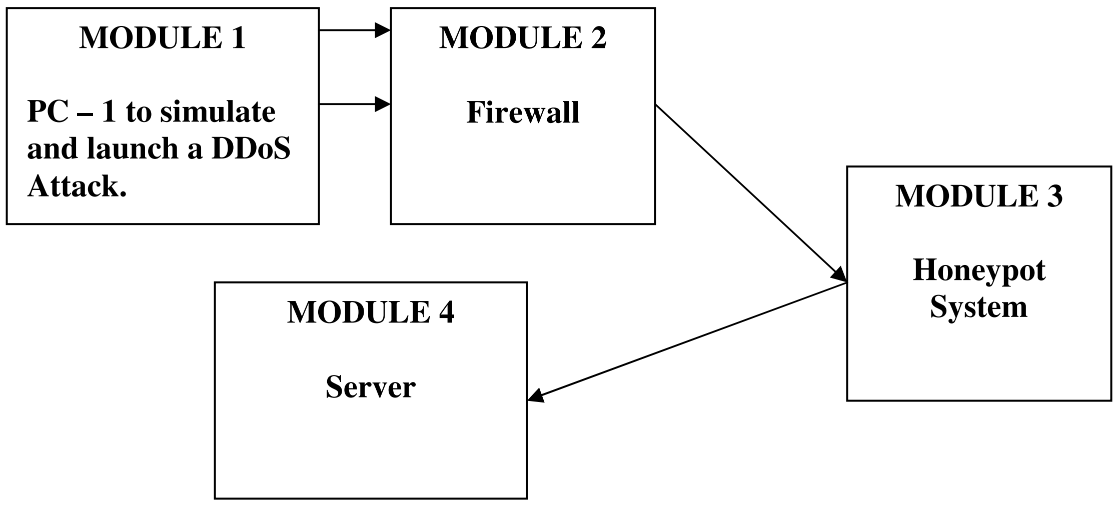


Fig5- 2 Design of the Project Part

5.3 Hardware and Software Requirements

Hardware Requirements

1. 5 PC's or workstations with Pentium IV processor or above
2. Each PC having at least 1 Gb of RAM.
3. One 8 to 1 fast Ethernet switch
4. 5 LAN wires.

Software Requirements

1. LOIC v1.07.0– to launch the DDoS attack.
2. OS - Windows XP SP2

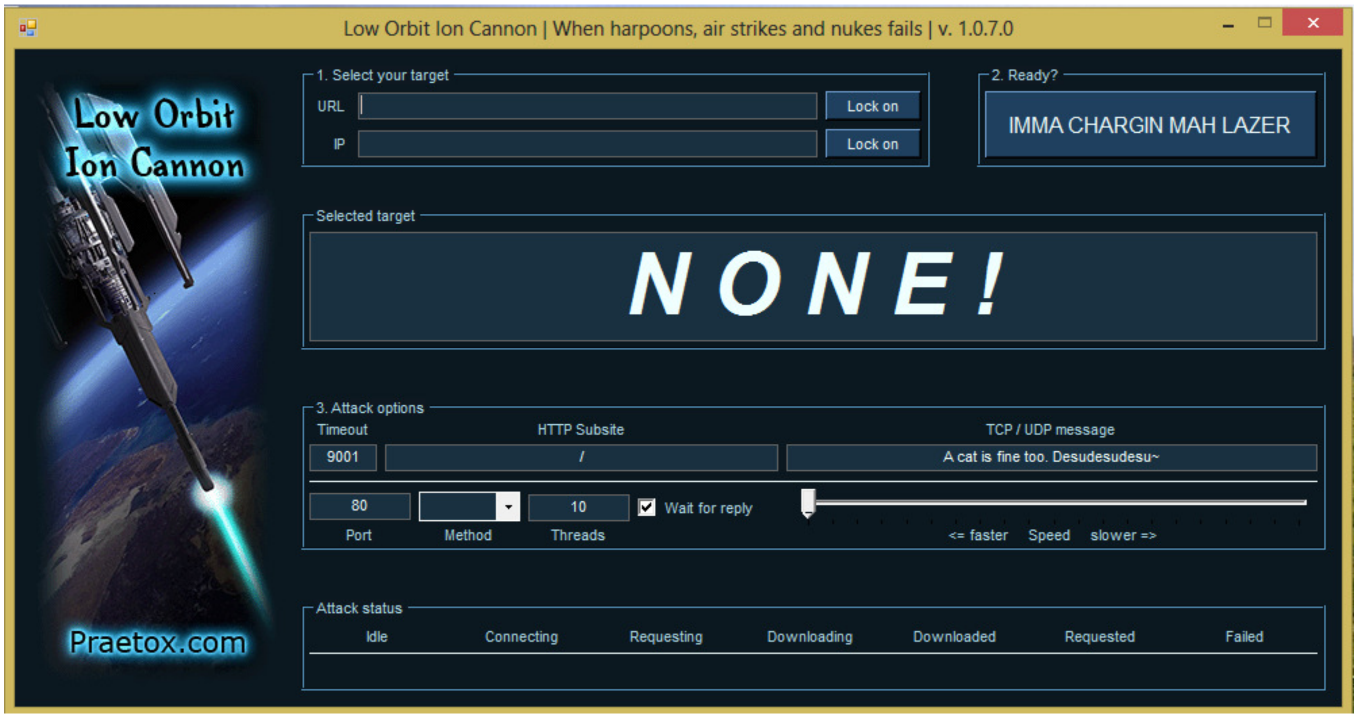
3. Net Beans IDE

4. 32 bit version of Java Development Kit v 1.6.0_01 and Java Runtime Environment v 1.6.0_01

5.4 Software Used for Launching Flooding Attack

The software used in the project for launching DDoS attack is LOIC-1.0.7.42 which was taken up from the internet, often used by script kiddies for launching attacks. It is used for flooding the destination machine with bogus traffic thus soaking up its network connections and disabling it from being able to respond in time to its legitimate clients.

It is a GUI based tool which allows a user to launch an attack against a machine.



URL:

IP:

Selected target: NONE!

Timeout: 9001

HTTP Subsite: /

TCP / UDP message: A cat is fine too. Desudesudesu~

Port: 80

Method:

Threads: 10

[✓] Wait for reply

Fig5- 3 GUI of LOIC-1.0.7.42

It has the following fields

1) **Select your target:** This allows the user to select the machine to be attacked . It has the following options

a) **URL:** Select the target by specifying the URL of the target machine.

b) **IP Address:** Select the target by specifying the IP Address of the target machine.

Each of these fields allow the option of lock on which lets the tool locate the device and tell the user if it can't be found.

2) **Attack options:** These contain a number of option which are provided by the tool. These are described below

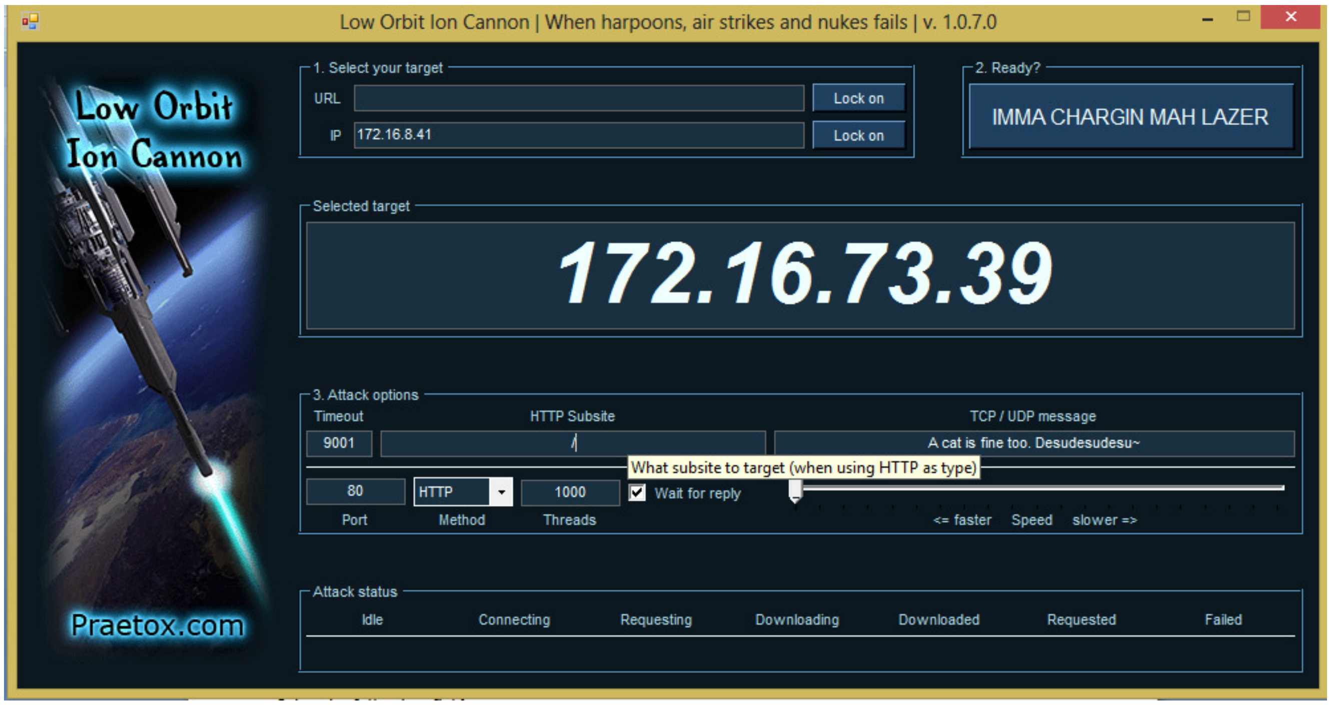
a) **Timeout:** Lets the user define the time, in seconds, after which he wants to terminate the attack.

b) **Threads:** Lets the user specify the number of threads with which he wants to flood the target machine. Each new thread creates a new connection with the target machine and sends in more packets to it.

c) **Type:** This allows the user to choose which protocol will be used to send the flooding packets. They can be either of TCP, UDP or HTTP.

d) **Port:** Specify the port number at which the packets are to be sent.

e) **HTTP Subtitle:** The domain name can be specified here and the tool sends flooding packets to here.


Low Orbit Ion Cannon | When harpoons, air strikes and nukes fails | v. 1.0.7.0

1. Select your target
URL
IP 172.16.8.41
Lock on
Lock on

2. Ready?
IMMA CHARGIN MAH LAZER

Selected target
172.16.73.39

3. Attack options
Timeout 9001
HTTP Subsite /
TCP / UDP message A cat is fine too. Desudesudesu~
What subsite to target (when using HTTP as type)
Port 80
Method HTTP
Threads 1000
Wait for reply ✓
<= faster Speed slower =>

Attack status
Idle Connecting Requesting Downloading Downloaded Requested Failed

Fig5- 4 An attack in progress using LOIC v1.07.0

<u>5.5 Working Mechanism</u>

Consider a scenario where a server of a network had been providing services through some port say 2222. We set up a honeypot at port 23 and wait for an intruder to try to gain access to. Next we launch an attack on port 2222 using the tools described above and see what steps the intruder will take to compromise the system.

1. When the attack takes place against our system it is detected by the DDoS detection algorithm and all the malicious traffic is stopped from being routed to the server.

2. We conduct the attack in absence of the DDoS detection mechanism to find how the server responds to this attack and how much the performance of the chat room application suffers.

3. We record the effectiveness of the DDoS detection mechanism in both detecting the attack and then in filtering the malicious traffic from the legitimate ones. This is essential for the smooth running of our server application.

4. Even during the attack the packets which are destined for the honeypot are allowed to pass through and routed to the firewall.

5.6 Software Development Life Cycle

The *incremental model* combines elements of the linear sequential model with the iterative philosophy of prototyping. The incremental model applies linear sequences in a staggered fashion as calendar time progresses.

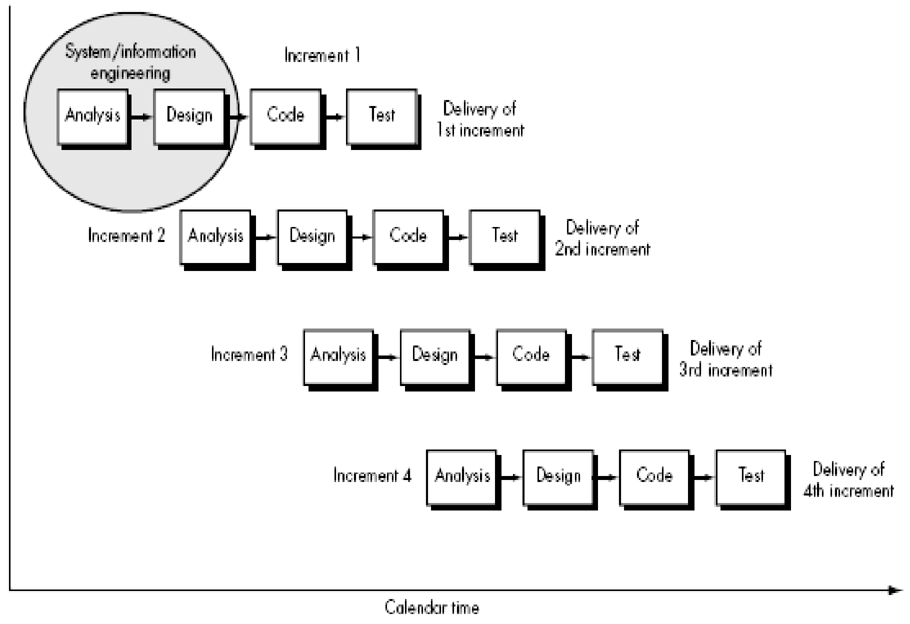


Fig5- 5 Incremental model

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, firewalls and Honeypot systems. The issues that were equally dealt with were virus, worms, spy ware and phishing.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.

5.7 Data Flow Diagrams

Level 0 DFD

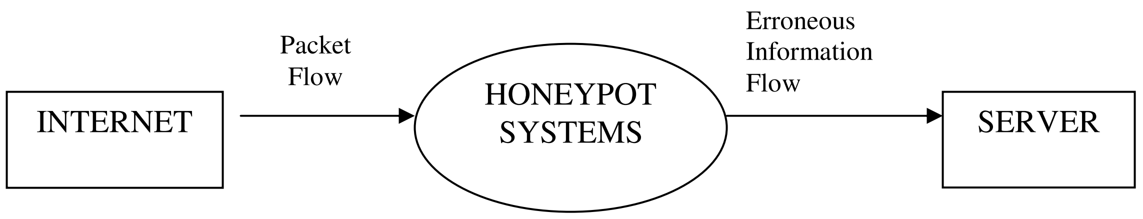


Fig5- 6 Level 0 DFD

Level 1 DFD:

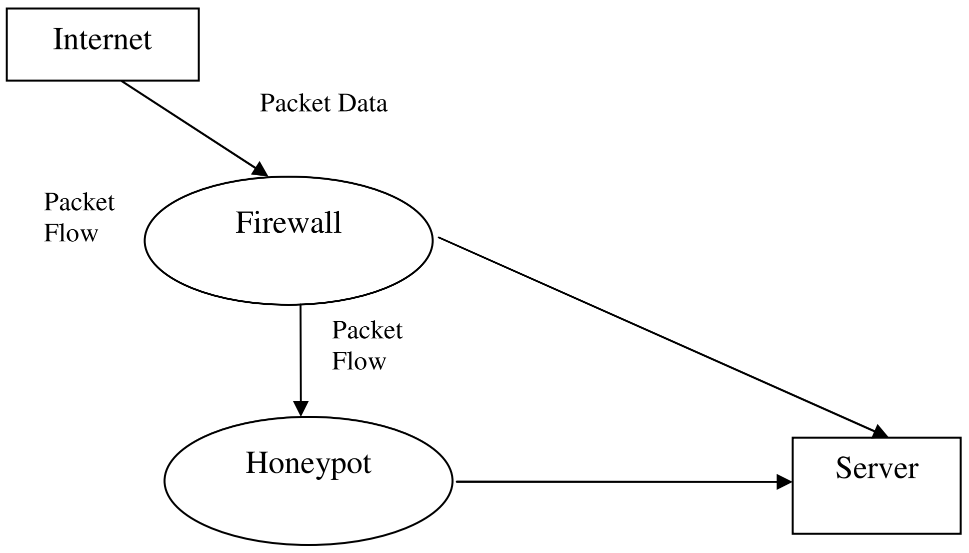


Fig5- 7 Level 1 DFD

CHAPTER 6: CODE SNIPPET

A brief overview of the all the classes used in different modules is given below. Apart from the native APIs provided in JDK two other libraries have been used. Their functionalities are also briefly explained.

6.1 jpcap Packet Capture Library

Java language doesn't have any native APIs which allow data to be captured in packet form enabling it to be read in the form of an IP packet. It was essential to read data in raw packet form for packet filtering firewall to function properly. jpcap implements bindings to the libpcap system library through JNI (the Java Native Interface). Jpcap is based on libpcap/winpcap, and is implemented in C and Java.

jpcap consists of a small shared-library which wraps libpcap plus a collection of Java classes.

The class jpcap.capture.PacketCapture allows packets to be captured either as a UDP, TCP, DataLink or as an IP packet format. This enables extracting of the IP Header and data segment part of the packet data by the firewall.

The jpcap distribution includes both

- A tool for real-time network traffic capture and analysis
- An API for developing packet capture applications in Java

6.1.1 NetworkInterfaceAddress

The NetworkInterfaceAddress class is used to store address that is retrieved from a captured packet. The fields address, subnet, broadcast and source are all of type InetAddress and store the ip address of the destination machine, subnet mask, broadcast value and the ip address of the source of the packet respectively

6.1.2 Class NetworkInterface

A NetworkInterface object contains some information about the corresponding network interface, such as its name, description, IP and MAC addresses, and datatlink name and description. To capture packets from a network, the first thing that needs to be done is to obtain the list of network interfaces on your machine. To do so, Jpcap provides JpcapCaptor.getDeviceList() method. It returns an array of NetworkInterface objects. The fields contained datalink, name and description are of type String while the field mac_address is of type byte. As their name suggests the mac_address field stores the mac_address of the source device while the name field contains the name of the device.

6.1.3 Class JpcapCaptor

It is used for capturing data packets. Once you obtain an instance of JpcapCaptor, you can capture packets from the interface. There are two major approaches to capture packets using a JpcapCaptor instance: using a callback method, and capturing packets one-by-one. In method callback approach, we implemented a callback method to process captured packets, and then pass the callback method to Jpcap so that Jpcap calls it back every time it captures a packet.

First, we implemented a callback method by defining a new class which implements the PacketReceiver interface. The PacketReceiver interface defines a receivePacket() method, so we needed to implement a receivePacket() method in our class as well.

6.1.4 Class PacketDispatcher

This class encapsulates a mechanism for dispatching network data and packets to a listener who has subscribed to such events.

Since this class doesn't contain an implementation for handling real network data, it is abstract.The following functions were utilised.

dispatchPacket()

Dispatches a packet to all registered listeners.

addPacketListener()

Registers a packet object listener with this capture system.

removePacketListener()

Deregisters a packet object listener from this capture system

<u>6.2 Package firewall</u>

The classes contained in this package are Firewall and DetectDDoS. This is the package which implements the code for the firewall module. It is here that each incoming packet is rerouted and each outgoing packet is scanned.

<u>6.2.1 Class Firewall</u>

The Firewall class implements a packet filtering proxy gateway. All incoming traffic packets are either forwarded to the Server or to the Honeypot. Since, the use of a honeypot exposes the organization to the risk of the Honeypot being compromised and being used for launching an attack against other computers, therefore each outgoing packet has to be scanned as well. This class has a number of function whose functionalities are explained below.

displayTrafficData()

This function calls a method of TrafficData to generate a GUI which displays the various packets which have been received by the firewall and how the firewall

connectToHoneypot()

This function creates a new connection to the honeypot for each malicious socket connection which is made to it by an attacker.

connectToServer()

This function is invoked only once for each new connection which a client establishes with the server. It creates a new connection for every inbound socket connection from the client.

forwardPacket()

This function forwards a packet to one of the previously established connections. It sends the packet either to the client or to the server. It makes use of the APIs present in jpcap.net to do so.

terminateConnection()

This function terminates a connection. Since the connection is terminated at the firewall, therefore it has to break two connections: one to the client and the other to the server or to the honeypot.

forwardToClient()

This function serves two purposes

- Checks all outbound packets to see if they are being sent as a response to some previously made request.
- Forwards the packets which arrive from the server or the honeypot to the client at the other end of the firewall.

The first functionality safeguards against the likelihood of the honeypot being compromised and being used to launch attacks against other devices.

genrateRecord()

This function creates a record of all the connections that were made and stores parameters such as the IP address from which they originated, the time at which a packet from a certain connection appeared at the firewall, the number of packets which originated from a certain connection, etc. These values are then used by DetectDDoS class.

6.2.1.1 Class AllConnections

This class is an inner class of Firewall. It implements the interface Runnable. It has functions which listen for new connections and respond to them accordingly. It allows for listening to new connections to take place in a new thread.It has two functions.

run()

This function is the implementation of the function run of interface Runnable required to run the code in a separate thread. It contains the code for reading from each of the new connection which is established from a device. It reads from both types of connections: incoming ones as well as outgoing ones.

listenForConnections()

Through this function the class listens to all inbound connections. It initializes some of the parameters and arrays and then calls upon the function run. It is called upon only once after which the function run() is invoked every time.

<u>6.2.1.2 Class TrafficData</u>

This is another inner class. It allows for creation of a GUI tool which lets the administrator look athe connections made to the proxy gateway graphically. It implements the Interface Runnable.

run()

This is the function which repaints the window which gives out the details of the traffic.

createGUI()

This function initializes data which is required to render the graphics. It creates the frame, adds buttons, initializes the default values and creates the window. It is called only once.

updateTrafficData()

This function is called to read from the records generated by generateRecord() function and the update the values in the window. Thus it periodically updates the number of connections made to the gateway, the number of incoming packets from each connection and the time at which a client connected to the gateway.

<u>6.2.2 Class DetectDDoS</u>

This class implements our version of the Source IP Address Monitoring algorithm. It contains several functions for record generation, creation of hash table and detecting the attack. It makes use of the records which were generated by class Firewall. The following are the functions contained in this class. That is why this class has been declared in the same package as the class firewall.

It also has two inner classes: DetectDDoS.ScanList , which generates relevant data, and DDoS.PacketPrinter, which is implemented for testing purposes.

run()

It is in this class that the SIM (Source IP Address Monitoring Scheme) is implemented.

It is implementation of run method of interface Runnable and describes the algorithm which needs to be run on a separate code stack. On discovering that the traffic from a particular IP Address is false, it designates it as hostile and invokes blockConnection().

blockConnection()

This function adds to the list of connections which must be blocked. It invokes Firewall.terminateConnection() and sends to it the network Address of the connection which has been recognized as hostile.

stopThread()

All that this function does is that it changes the state of boolean variable continue_running to false. Since it is a public function, it can be invoked from outside the class as well.

<u>6.2.2.1 Class ScanList</u>

Its an inner class of DetectDDoS. It is in this class that a DDoS is actually detected. The class extends the interface Runnable.

run()

This is the method which scans all the connection data that is collected by Firewall.generateRecord() and comes up with values for *Tn, Dn and* Δ_n. The detection algorithm works on the basis of the values generated here.

6.2.2.2 Class PacketPrinter

This class is implemented for testing purposes. It is used to track the problems in the code. The sole function in this class is receivePacket(). The reason that it is an inner class is that it also accesses data members of class DetectDDoS such as the packet data values, source IP Address of the packet and other relevant data.

receivePacket()

This function simply extracts all the information from the header and the data section of the packet which it receives and the prints those on the console output. It extracts values such as source network address, header length, data value, etc. and prints them to the screen.

6.3 Package faketelnetserver

This is the package where honeypot is implemented. As mentioned before the honeypot is a low interaction one and needs to be safeguarded against compromise by an intruder.

6.3.1 Class EngageClient

This is the class where the code for capturing an intruder's key strokes are present. It has three functions and implements the Ruunable interface. This is required to ensure that even if a honeypot crashes a file has been generated which can tell about the attacker's mechanisms.

stopClinet()

This function is invoked to terminate a connection. It does so by logging out client and sending the message, "Unexpected Error At Remote Server."

run()

This is the implementation of the Runnable.run() function. It contains the code which responds to each command of the intruder with the message, "Server Busy: Please Wait". It calls the function recordKeyStrokes().

recordKeyStrokes()

This function implements the code for reading the user's keystrokes and recording them. It writes each of the keystroke to a text file and updates it every time user hits enter key.

<u>6.3.2 Class FakeTelnetServer</u>

This is the class where the code for honeypot is implemented. The honeypot allows the user to login but then doesn't respond to any other command. The intruder tries to take advantage of some vulnerability which is recorded by the honeypot.

go()

This function implements the code for the honeypot. It creates instances of EngageClient and FetchPassword. It then calls the functions of these instances to simulate the login process of a real telnet server.

<u>6.3.3 Class FetchPassword</u>

This class makes the user feel that he is logging into a real telnet server. If this doesn't happen, then the intruder will know that the telnet server is a fake and will try other ways to gain entry into the system.

readPassword()

This function reads the password and calls upon MaskCharacter.run(). It does so by creating a new instance of MaskCharacters and invoking start().

<u>6.3.4 Class MaskCharacters</u>

This class is invoked to mask the characters which are input by the user while entering his or her password. It implements Runnable.

stopMasking()

It only changes the value of boolean variable continue_masking to false and stops the ruuning of the thread.

run()

Each character is read and then it is deleted. So it appears that the character can't be typed on the telnet screen. It is run on a separate stack as long as value of continue_masking is true.

<u>**6.4 Package chatserver**</u>

This package contains the code for server and a chat room application. The server itself is very simple and connects to all the clients.

<u>**6.4.1 Class ChatServer**</u>

This class contains the code for the chat room application. It receives the data sent to it by all the clients connected to it and regularly update their values. So all the clients can communicate to each other. It has an inner class which handles the connection with all the clients.

go()

It creates a single socket for listening to the requests coming from the various client. It next instantiates a new thread with an object of HandleClients and calls start() over it.

sendToAllClients()

It sends to all the clients connected to the message which was received from any one of them. Thus it carries out a multicast.

<u>**6.4.1.1 Class HandleClients**</u>

This class is an inner class of ChatServer. It extends Runnable interface. Its task is to read from the socket and forward the String to sendToAllClients().

run()

This is the function which reads from the socket to which all the clients write. It is required to keep the server multithreaded.

CHAPTER 7 : APPENDIX

7.1 List of Abbreviations

1. DoS: Denial of Service
2. DDoS: Distributed Denial of Service
3. IDS: Intrusion Detection System
4. RAT: Remote Administration Tool
5. DNS: Domain Name System
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. SMTP: Simple Mail Transfer Protocol
17. DFD: Data Flow Diagram

CHAPTER 8: BIBLOGRAPHY

1. Know Your Enemy http://project.honeynet.org

2. Honeypots: Tracking Hackers By Lance Spitzner

3. http://www.wikipedia.org/worms

4. http://www.wikipedia.org/virus

5. Lance Spitzner Definitions and Value of Honeypots.

6. http://www.trackinghackers.com/papers/honeypots.html

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

8. Tutorial on use of jpcap library http://www.eden.rutgers.edu/~muscarim/jpcap/tutorial/

9. Jpcap documentation Pages available at http://jpcap.sourceforge.net/javadoc/net/sourceforge/jpcap/capture/PacketCapture.html

10. Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring: Research Paper by Tao Peng Christopher Leckie Kotagiri Ramamohanarao

11. Honeypots: Concepts, Approaches, and Challenges by Iyatiti Mokube and Michele Adams