Design and Development of Anti-DoS/DDoS Attacks Framework Using IPtables

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

by

Misha Singhal

(Roll No. 800932013)

Under the supervision of:

Mrs. Shalini Batra

(Assistant Professor)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2011

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Design and Development of Anti-DoS/DDoS Attacks Framework using IPtables"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mrs. Shalini Batra*, and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Misha Singhal)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Mrs. Shalini Batra)
Assistant Professor
Computer Science and Engg. Deptt.
Thapar University
Patiala

Countersigned by

(Dr. Maninder Singh)
Associate Professor and Head
Computer Science and Engg. Deptt.
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Design and Development of Anti-DoS/DDoS Attacks Framework using IPtables"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mrs. Shalini Batra*, and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Misha Singhal)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(**Mrs. Shalini Batra**)
Assistant Professor
Computer Science and Engg. Deptt.
Thapar University
Patiala

Countersigned by

(Dr. Maninder Singh)
Associate Professor and Head
Computer Science and Engg. Deptt.
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

Many people have shared their time and expertise to help me accomplish my goal. First, I would like to sincerely thank my guide, Mrs. Shalini Batra, Assistant Professor, Thapar University , Patiala for her constant support and guidance. Her instant responses to my countless inquiries have been invaluable and motivational. It was a great opportunity to work under her supervision.

Many thanks to Dr. Maninder Singh, Associate Professor and Head, CSED, who first introduced me to the concept of DoS/DDoS attacks and whose lectures have inspired me to choose the current research focus. I would also like to thank him for his moral support and research environment he had facilitated for this work

I would also like to thank all my Teachers for their stimulating discussion and invaluable suggestions during the period of my work.

Finally I wish to thank my family and friends for their immense love and encouragement throughout my life without which it would not have been possible to complete this work.

Last but not the least I would like to thank God who have always been with me in my good and bad times.

Misha Singhal

(800932013)

Abstract

Denial-of-Service (DoS) is a network security problem that poses a serious challenge to trustworthiness of services deployed on the servers. The aim of DoS attacks is to make services unavailable to legitimate users by flooding the victim with legitimate-like requests and current network architectures allow easy-to-launch, hard-to-stop DoS attacks.

Threat of DoS attacks has become even more severe with DDoS (Distributed Denial-of-Service) attack .It is an attempt by malicious users to carry out DoS attack indirectly with the help of many compromised computers on the Internet. Attackers can compromise a huge number of computers by spreading a computer worm using vulnerabilities in popular operating systems. This exhausts the victim network of resources such as bandwidth, computing power, etc., the victim is unable to provide services to its legitimate clients and network performance is greatly deteriorated, moreover, with little or no advance warning, a DDoS attack can easily exhaust these resources within a short period of time.

Service providers are under mounting pressure to prevent, monitor and mitigate DoS/DDoS attacks directed toward their customers and their infrastructure. Defending against those types of attacks is not a trivial job, mainly due to the use of IP spoofing and the destination-based routing of the Internet, though there are many proposed methods which aim to alleviate the problem like Firewalls, Traffic Volume Normalization, Intrusion Detection Systems, Ingress filtering, IP Traceback , SYN Proxy etc.

This work discusses about the efficient packet filtering technique using firewall to defend against DoS/DDoS attacks. Firewall scripts are written using command-line tool IP Tables in Linux to deny the suspicious traffic. Packet sniffer tool is used to showcase the effectiveness of the scripts in mitigating the various kinds of DoS/DDoS attacks.

Table of Contents

Certificate…….……..…….. i

Acknowledgement…….……..... ii

Abstract…….……..…… iii

Table of Contents……..…….….. iv-vi

List of Figures……..…….…...……. vii-viii

List of Tables ……...…….……....ix

Chapter 1

Introduction……1

1.1 DoS Attacks……1

1.2 DDoS Attacks……..2

1.3 Types of DoS/DDoS Attacks……...3

1.3.1 UDP Flood Attack……3

1.3.2 ICMP Flood Attack……..3

1.3.3 SYN Flood Attack……4

1.3.4 Smurf Attack……5

1.3.5 Ping of Death Attack……6

1.3.6 Teardrop Attack……...6

1.3.7 Land Attack……..7

1.4 DDoS Attack Tools……..7

1.4.1 Trinoo……...7

1.4.2 Tribe Flood Network……9

1.4.3 TFN2K……9

1.4.4 Mstream……...9

1.4.5 Shaft……...…10

1.5 DoS/DDoS Attack Defense Challenges……....10

1.6 Thesis Outline……11

Chapter 2 Literature Review……...…….12

Chapter3 Problem Statement……..15

Chapter 4 DoS/DDoS Defense Mechanisms…….16

4.1 IP Traceback……..16

4.1.1 Link Testing……...…17

4.1.2 Logging……..18

4.1.3 ICMP based Traceback……..19

4.1.4 Packet Marking……..19

4.2 Network Ingress Filtering……..20

4.3 Intrusion Detection Systems……..22

4.3.1 Host based Intrusion Detection System…….22

4.3.2 Network based Intrusion Detection System……...23

4.4 Firewalls…….25

4.4.1 Stateless Packet Filtering……...28

4.4.2 Dynamic Filtering……..29

4.4.3 Application Proxies……30

4.4.4 Firewalls using Iptables in Linux……..….30

Chapter 5 Implementation Details and Experimental Results……...…35

5.1 Implementation Steps……35

5.2 Software setups and Testing……..36

5.3 Experiment and Results…….37

5.3.1 Module 1-SYN Flood Attack……37

5.3.2 Module 2-UDP Flood Attack……41

5.3.3 Module 3-ICMP Flood Attack……..45

5.3.4 Module 4-Trinoo Attack……49

5.3.5 Module 5-Mstream Attack…….50

5.3.6 Module 6-Shaft Attack……..51

Chapter 6 Conclusion and Future Scope…….52

6.1 Conclusion…….52

6.2 Future Scope……..53

References……54

List of Publications……..59

List of Figures

Figure 1.1 Distributed Denial of Service Attack…… 2

Figure 1.2 Client Server Communication…….……... 4

Figure 1.3 SYN Flood Attack……...…….. 5

Figure 1.4 Smurf Attack……...6

Figure 1.5 Teardrop Attack……..…….7

Figure 1.6 DDoS attack using Trinoo attack tool …….……8

Figure 1.7 DDoS attack using Mstream attack tool……..…9

Figure 4.1 Link Testing Approach…….……18

Figure 4.2 Logging Packets at Router…….……18

Figure 4.3 ICMP Traceback……..……..19

Figure 4.4 Packet Marking…….20

Figure 4.5 Ingress Filtering…….……21

Figure 4.6 Host based Intrusion Detection System…….23

Figure 4.7 Network based Intrusion Detection System…….……..24

Figure 4.8 Firewall System…….26

Figure 4.9 Packet Flow Diagram…….……33

Figure 5.1 Implementation Steps……35

Figure 5.2 Virtual Network Environment…….……..36

Figure 5.3 Wireshark Output showing SYN Flood Attack…….37

Figure 5.4 TCP Flow graph showing SYN Flood Attack……...38

Figure 5.5 I/O Graph showing SYN Flood Attack……38

Figure 5.6 List of Iptables Rules against SYN Flood Attack……39

Figure 5.7 Wireshark output after SYN Flood Attack protection……...39

Figure 5.8 TCP Flow Graph after SYN Flood Attack protection……...40

Figure 5.9 Wireshark Output showing UDP Flood Attack…….……41

Figure 5.10 Flow graph showing UDP Flood Attack……42

Figure 5.11 I/O Graph showing UDP Flood Attack…….42

Figure 5.12 List of IPtables Rules against UDP Flood Attack…….43

Figure 5.13 Wireshark output after UDP flood Attack protection……44

Figure 5.14 UDP Flow graph after UDP Flood Attack protection…..……44

Figure 5.15 Wireshark output showing ICMP flood attack……..……...45

Figure 5.16 Flow Graph showing ICMP Flood attack……...……...46

Figure 5.17 I/O Graph showing ICMP Flood Attack……46

Figure 5.18 List of Iptables rules against ICMP Flood attack……..47

Figure 5.19 Wireshark Output after ICMP Flood Attack protection……48

Figure 5.20 Flow graph after ICMP Flood Attack protection…….….48

List of Tables

Table 4.1 Firewalls Layers and Models…….27

Table 4.2 Firewall Rules……29

Table 4.3 Target and Jumps……...32

Chapter 1

Introduction

With the advancement of information and communication technologies, our societies are evolving into global information societies with a ubiquitous computing environment but unfortunately security systems and policies to govern this environment have not progressed rapidly as desired. Attackers keep no stone unturned in exploiting various vulnerabilities that exist in the applications running on the systems.

Among the various cyber attacks like SQL injection, Session hijacking, Cross-site scripting attack, Denial of Service (DoS) attack have emerged as the most threatening attack till date as very few techniques have been successful in alleviating this attack.

1.1 DoS Attacks

A Denial of Service (DoS) attack is a type of attack focused on disrupting availability of service. Such an attack can take many shapes, ranging from an attack on the physical IT environment, to the overloading of network connection capacity, or through exploiting application weaknesses.

Gligor et al. [1] defined DoS as: "a group of otherwise-authorized users of a specific service is said to deny service to another group of authorized users if the former group makes the specified service unavailable to the latter group for a period of time which exceeds the intended (and advertised) waiting time."

Internet-facing and other networked infrastructure components are at risk of DoS for two primary reasons:

1. Resources such as bandwidth, processing power, and storage capacities are not unlimited and so DoS attacks target these resources in order to disrupt systems and networks.
2. Internet security is highly interdependent and the weakest link in the chain may be controlled by someone else thus taking away the ability to be self reliant.

1.2 DDoS Attacks

In Distributed Denial of Service (DDoS) attacks, attackers do not use a single host for their attacks but a cluster of several dozens or even hundreds of computers to do a coordinated strike. From the beginning the evolution of solutions to resolve the occurrence of attacks promoted the evolution of the attacks itself, and nowadays DoS attacks have been superseded by DDoS attacks. The WWW Security FAQ [2] on Distributed Denial of Service says that:

"A Distributed Denial of Service attack uses many computers to launch a coordinated DoS attack against one or more targets. Using client/server technology, the perpetrator is able to multiply the effectiveness of the Denial of Service significantly by harnessing the resources of multiple unwitting accomplice computers which serve as attack platforms. Typically a DDoS master program is installed on one computer using a stolen account. The master program, at a designated time, then communicates to any number of "agent" programs, installed on computers anywhere on the Internet. The agents, when they receive the command, initiate the attack. Using client/server technology, the master program can initiate hundreds or even thousands of agent programs within seconds".

Figure 1.1 gives a clear idea about DDoS attack.

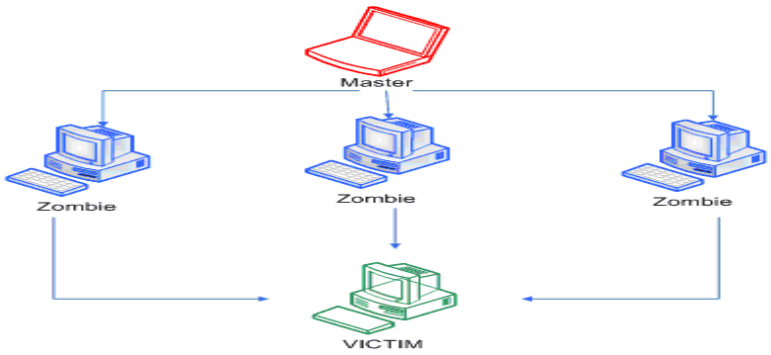

Figure 1.1: Distributed Denial of Service Attack [3]

There are two major reasons making DDoS attacks attractive for attackers. The first is that there are effective automatic tools available for attacking any victim [4], i.e., expertise is not necessarily required. The second is that it is usually impossible to locate an attacker without extensive human interaction or without new features in most routers of the Internet [5-7].

1.3 Types of DoS/DDoS Attacks

In DDOS attacks the attacker sends packets directly from his computer(s) to the victim's site but the source address of the packets may be forged. There are many tools available to allow this type of attack for a variety of protocols including ICMP, UDP and TCP. Some common tools include stream2, synhose, synk7, synsend, and hping2**.** Some of the common DDoS attacks are discussed below.

1.3.1 UDP Flood Attack

In UDP Flood attack attacker sends large number of UDP packets to a victim system, due to which there is saturation of the network and the depletion of available bandwidth for legitimate service requests to the victim system.

A UDP Flood attack is possible when an attacker sends a UDP packet to a random port on the victim system. When the victim system receives a UDP packet, it will determine what application is waiting on the destination port. When it realizes that there is no application that is waiting on the port, it will generate an ICMP packet of ''destination unreachable'' [8] to the forged source address. If enough UDP packets are delivered to ports of the victim, the system will go down. By the use of a DoS tool the source IP address of the attacking packets can be spoofed and this way the true identity of the secondary victims is prevented from exposure and the return packets from the victim system are not sent back to the zombies.

1.3.2 ICMP Flood Attack

ICMP Flood attacks exploit the Internet Control Message Protocol (ICMP), which enables users to send an echo packet to a remote host to check whether it's alive. More specifically during a DDoS ICMP flood attack the agents send large volumes of ICMP_ECHO_ REPLY packets (''ping'') to the victim.

These packets request reply from the victim and this results in saturation of the bandwidth of the victim's network connection [9]. During an ICMP flood attack the source IP address may be spoofed.

1.3.3 SYN Flood Attack

In a SYN Flood attack, the victim is flooded with Half open connections.

The client system begins by sending a SYN message to the server. The server then acknowledges the SYN message by sending SYN-ACK message to the client. The client then finishes establishing the connection by responding with an ACK message. The connection between the client and the server is then open, and the service-specific data can be exchanged between the client and the server. Figure 1.2 shows the view of this message flow:

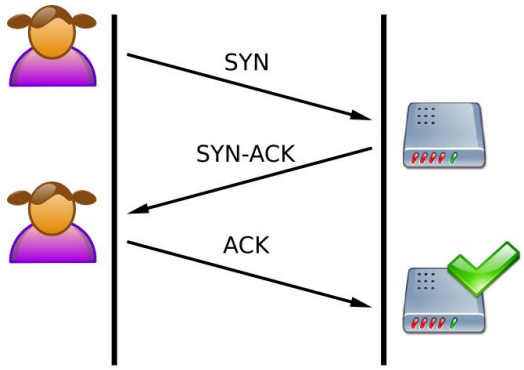

Figure 1.2: Client server communication [10]

The potential for abuse arises at the point where the server system has sent an acknowledgment (SYN-ACK) back to client but has not yet received the ACK message. This is known as half-open connection. The server has built in its system memory a data structure describing all pending connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially-open connections.

Creating half-open connections is easily accomplished with IP spoofing. The attacking system sends SYN messages to the victim server system; these appear to be legitimate but in fact reference a client system that is unable to respond to the SYN-ACK messages. This means that the final ACK message will never be sent to the victim server system.

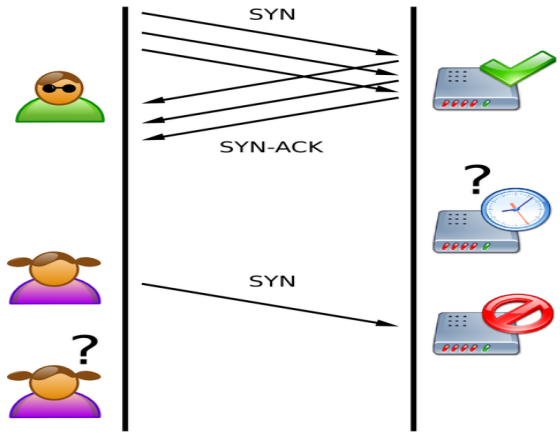

Figure 1.3: SYN Flood Attack [10]

Figure 1.3 shows the scenario of half open connection. The half-open connections data structure on the victim server system will eventually fill; then the system will be unable to accept any new incoming connections until the table is emptied out [11].

1.3.4 Smurf Attack

In a "smurf" attack, the victim is flooded with Internet Control Message Protocol (ICMP) "echo-reply" packets. On IP networks, a packet can be directed to an individual machine or broadcast to an entire network. When a packet is sent to an IP broadcast address from a machine on the local network, that packet is delivered to all machines on that network.

In the "smurf" attack [12], attackers are using ICMP echo request packets directed to IP broadcast addresses from remote locations to generate denial-of-service attacks. When the attackers create these packets, they do not use the IP address of their own machine as the source address. Instead, they create forged packets that contain the spoofed source address of the attacker's intended victim. The result is that when all the machines at the intermediary's site respond to the ICMP echo requests, they send replies to the victim's machine. The victim is subjected to network congestion that could potentially make the network unusable. Figure 1.4 depicts the stepwise process of the attack.

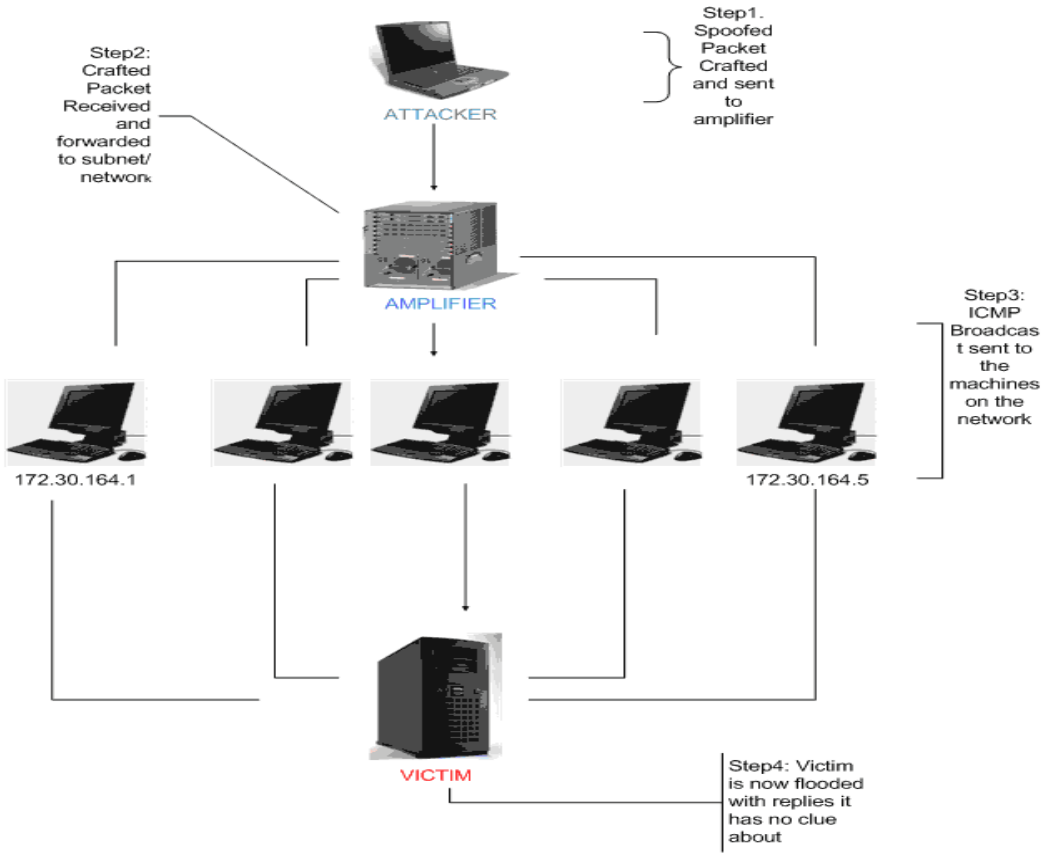

Figure 1.4: Smurf attack [14]

1.3.5 Ping of Death Attack

In Ping of Death attacks, the attacker creates a packet that contains more than 65,536 bytes, which is the limit that the IP protocol defines. This packet can cause different kinds of damage to the machine that receives it, such as crashing and rebooting.

```
ping -l 86600 victim.org
```

1.3.6 Teardrop Attack

This type of denial of service attack exploits the way that the Internet Protocol (IP) requires a packet that is too large for the next router to handle be divided into fragments. The fragment packet identifies an offset to the beginning of the first packet that enables the entire packet to be reassembled by the receiving system. In the teardrop attack, the

attacker's IP puts a confusing offset value in the second or later fragment. If the receiving operating system does not have a plan for this situation, it can cause the system to crash**.**

Figure 1.5 shows the packet prepared for teardrop attack.

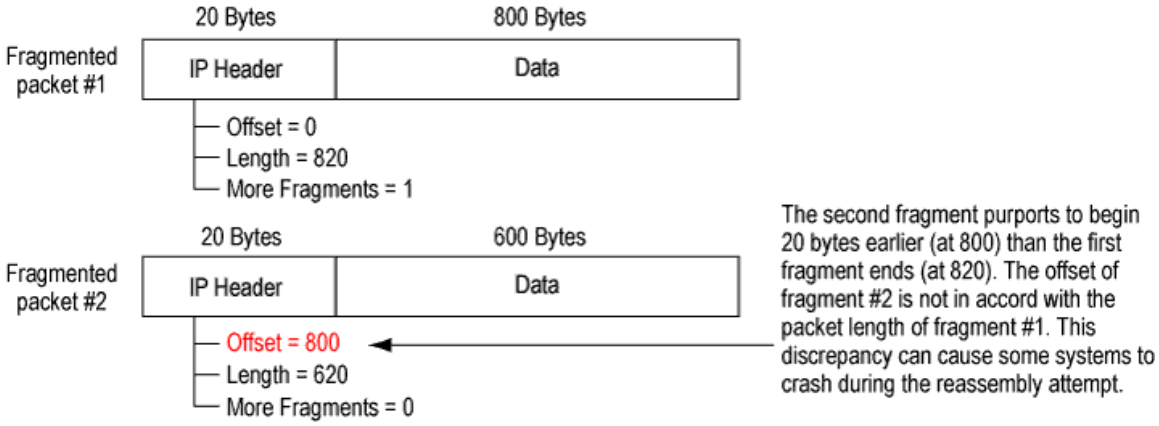

Figure 1.5: Teardrop Attack [14]

1.3.7 Land Attack

The attack involves sending a spoofed TCP SYN packet (connection initiation) with the target host's IP address to an open port as both source and destination. The reason a LAND attack works is because it causes the machine to reply to itself continuously.

Land attacks have been found in services like Simple Network Management Protocol (SNMP) and Windows 88/tcp (kerberos/global services) which were caused by design flaws where the devices accepted requests on the wire appearing to be from themselves and causing replies repeatedly [15].

1.4 DDoS Attack Tools

There are various attack tools used by the attackers to carry out distributed of denial service (DDoS) attacks. Some of thethem are discussed below.

1.4.1 Trinoo

Trinoo(Trin00) was the first DDOS tool to be discovered. It was found in the binary form on Solaris 2.x systems compromised by buffer overrun bug in RPC services: statd, cmsd,

ttdbserverd. Trinoo daemons were UDP based, password protected remote command shells running on compromised systems [9].

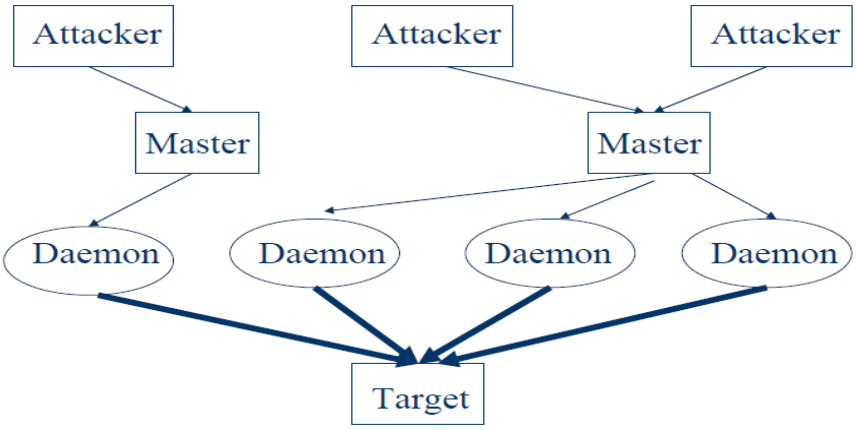

Figure 1.6: DDoS attack using Trinoo attack tool

As seen in the Figure 1.6 the attacker controls one or more master servers by password protected remote command shells and the master systems control multiple daemon systems. Trinoo calls the daemons "Bcast" hosts. Daemons fire packets at the target specified by the attacker.

Communication

- **Attacker to Master**: 27665/TCP. The attacker must supply the correct password (betaalmostdone). If someone else "logs in", a warning is flashed to the 1stuser.
- **Master to Daemons:** 27444/TCP. Command lines are of form: arg1 password arg2 and the default password for commands is 144asdl.Only Commands with "144" substrings are run.
- **Daemon to Master:** 31335/UDP. When daemon starts up, it sends a HELLO to the master. Master adds this daemon to its list.
- Master sends PNG to daemon on 27444/UDP, daemon replies PONG on 31335/UDP. This way, the master knows daemon is still alive.

1.4.2 Tribe Flood Network (TFN)

Communication between the master and the slaves is using ICMP echo reply packets. The attack could be of smurf, SYN flood, UDP flood and ICMP flood attacks.

1.4.3 TFN2K

This is the newer version of the TFN attack and it uses TCP, UDP, ICMP or all three to communicate between the master and the slaves and the communication is encrypted. The attacks implemented are the same as TFN.

1.4.4 Mstream

The "mstream" tool consists of a handler and an agent portion, much like Trinoo as seen in Figure 1.7

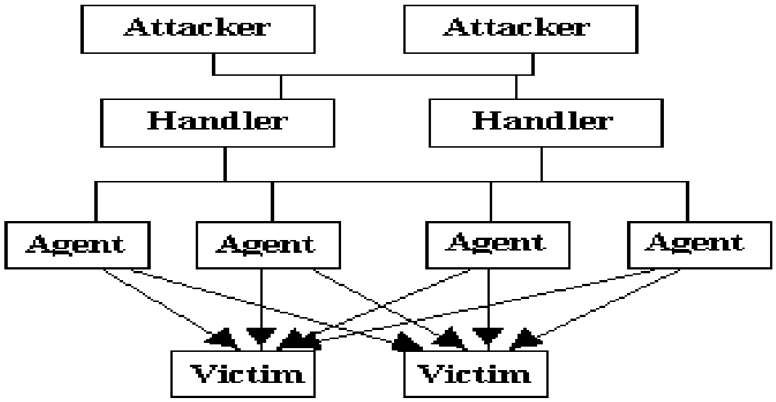

Figure 1.7: DDoS attack using Mstream attack tool [16]

Communication

- Attacker-handle- unencrypted TCP –6723/tcp, 12754/tcp, 15104/tcp
- Handler-agent - cleartextUDP –7983/udp, 6838/udp
- Agent to Handler(s) –9325/udp, 6838/udp

Handler expects commands to be contained entirely in the data field of a single TCP packet. This means telnet cannot be used to send commands. Handler/agent traffic is UDP based. Agent commands are / separated lists with some colon separated lists. Proper password (N7%diApf) must be given. All connected users are notified of the new connection success or failure [16].

1.4.5 Shaft

This attack is modeled after Trinoo. Communications between the master and the slave is achieved using the UDP packets and the attack implemented is the UDP flood attack. An important feature of this attack is its ability to switch control master servers and ports in real time, thereby making the detection by intrusion detection tools difficult.

In this attack tool Handler is called shaftmaster and Agents are called shaftnodes. Attacker uses a telnet program, "client" to talk to handlers [17].

Communication

- Client to handler: 20432/tcp
- Handler to agent: 18753/udp
- Agent to handler: 20433/udp

1.5 DoS/DDoS Attack Defense Challenges

In general, four fundamental challenges contribute to the difficulty of the DoS problem:

• **Hard prevention**. Software bugs, misconfiguration, and human's inherent vulnerability to social engineering [18] all enable a remote attacker to cause a resource to be unavailable to legitimate users. For instance, an attacker may exploit a bug in a packet queue implementation that enables a specially crafted packet to set tail = head, effectively rendering queue capacity to zero.

• **Weak deterrents**. It is easy for the real attack perpetrators to hide behind an army of unwitting zombies that launch the attack on their behalf.

• **Difficult resource control**. Effective resource management and protection ensures that entities (e.g., users and processes) be allocated their resource shares and no entity can gain more than its share. A challenge here is the definition of entities. Entities can be processes, sockets, IP addresses, user tokens, etc. The granularity of an entity determines how much control is available and how much overhead is incurred.

• **Difficult attacker identification**. It is hard to come up with a silver-bullet definition of malicious entities to block their access to attacked resources. Many attempts have been made to define characteristics of a malicious entity: sends one-way traffic [19], sends

more traffic than normal [20], sends more “heavy” requests than normal [21], appears on system logs only during periods of attack [22], and is controlled by *bots* (non-human agents controlling compromised machines) [23]. These attempts are either specific to particular services or can be bypassed by new attacks.

1.6 Thesis Outline

The thesis is divided into six chapters:

Chapter 1: The introduction to the work is provided.

Chapter 2: Literature review section gives a brief survey of research going in the area of mitigation of DoS/DDoS attacks and recent packet sniffing and generating tools are also discussed.

Chapter 3: After going through the literature review, the problem statement has been identified and defined in this chapter.

Chapter 4: The general concepts used in the thesis work are introduced. This chapter gives a description of the techniques used for mitigating DoS/DDoS attacks.

Chapter 5: In this chapter, experimental results are achieved by implementing techniques like Iptables scripts and packet capturing using Wireshark tool. The illustrations captured during the work done have also been discussed briefly.

Chapter 6: Focuses on the conclusion of the work presented in this thesis and ideas for further research have been proposed.

Chapter 2

Literature Review

DoS/DDoS attacks have become more and more frequent in the last years. Large scaled networks of infected PCs (bots or zombies) combine their bandwidth and computational power in order to overload a publicly available service and denial it for legal users.

The attacks against large e-commerce sites in February 2000 and the attacks against root DNS servers in 2003 and 2007 have drawn public attention to the problem of DDoS attacks [25,26]. Today, mainly mid-sized websites are attacked by criminals in order to extort protection money from their owners without attracting too much public attention. Besides that, also Internet Service Providers (ISP) have to deal with the problem that DDoS traffic is congesting their link bandwidths.

Early tools like TFN [9], Stacheldraht[25], Trinoo[9] or Mstream[16] used unencrypted and hierarchically organized communication structures. Most of these tools used TCP-SYN, UDP or ICMP floods with possibly identifiable parameters. Since some of these attacks have successfully been mitigated, a new generation of bots arose. SDBot[27], Agobot[28] or the very enhanced Phatbot[29] are known representatives which use IRC as a robust and secure communication [30]. These tools also contain methods for spreading themselves and have more sophisticated attack algorithms, which could be upgraded over the internet.

Mitigating DoS/ DDoS attacks at the origin or within the core of the internet seems to be an impossible task due to the distributed and authorization-free nature of the IP based network. Various approaches to find the source IP of attacker using filtering mechanisms have been proposed.

IETF documented Ingress filtering approach as described in RFC 2827 [31] helps mitigating DDoS attacks with forged source IP addresses (IP spoofing) thus indirectly combat various types of net abuse by making Internet traffic traceable to its source. Network ingress filtering is a “good neighbor” policy, which relies on mutual cooperation

between ISPs for their mutual benefit. To prevent IP address manipulation, Park et al. [32] proposed to install distributed packet filters on autonomous systems over the Internet to stop packets with spoofed IP addresses.

Savage et al. [33] suggested IP Traceback to find the source of spoofed IP addresses by probabilistically marking packets. Song et al. [34] propose an enhanced scheme of probabilistic packet marking to reduce the false positive rate for reconstructing the attack path. Another enhanced scheme of probabilistic packet marking has been proposed by Bellovin et al. [35] to reduce the computational overhead. This is ICMP traceback scheme, which is similar to the probabilistic packet marking scheme. In this scheme, routers generate ICMP packets to the destination with a low probability. For a significant traffic flow, the destination can gradually reconstruct the route that was taken by the packets in the flow.

Mahajan et al. [36] provide a scheme in which routers learn a congestion signature to tell good traffic from bad traffic. Siris et al. [37] present two anomaly detection algorithms for detecting TCP SYN attacks: an adaptive threshold algorithm and a particular application of the cumulative sum algorithm for change point detection. The adaptive threshold algorithm compares the number of SYN packets received over a given interval to the estimated number based on recent measurements. To raise an alarm, the threshold has to be exceeded consecutively. The CUSUM algorithm uses the difference between the number of SYN packets in a time interval and the estimated number for the same interval as a Gaussian random variable. A SYN flooding attack is then detected using the cumulative sum based on the likelihood of the momentary value being caused by a change in the mean traffic rate.

Various authors including Wang et al., [38] proposed technique to detect SYN Flood attack which was based on the protocol behavior of TCP SYN-FIN (RST) pairs at leaf routers that connect end hosts to the Internet. The difference between the number of observed SYN and FIN (RST) packets is observed and a CUSUM algorithm is applied for a change point detection. To reduce the impact of different access patterns of different sites, the difference between the number of SYNs and FINs (RSTs) is normalized by an estimated average number of FINs (RSTs).

Luo et al. [39] also suggested a scheme to detect Pulsing DoS (PDoS) attacks at the victim's network using two anomalies caused by PDoS attacks, namely the fluctuation of the incoming data traffic, and the decline of outgoing TCP ACK traffic.

Cabrera et al. [40] proposed system which detects DDoS using information from MIB (Management Information Base) traffic variables for attacker and target. Signatures to match for attack detection were determined from known attacks. On the attacker's side, a DDoS attack should be detected prior to its launch by identifying MIB-based precursors.

Apart from the above proposed works Intrusion detection Systems and Firewalls are among the best defensive systems which works by matching the packets against predefined rules and filtering them accordingly. Linux kernels had packet filtering since the 1.1 series. In late 1994, kernel hacker Alan Cox ported the firewalling functionality of ipfw from BSD into Linux. In mid-1998, Rusty Russell et al. [41] reworked much of the networking under the Linux kernel in the 2.1 development series and introduced the user space tool ipchains. The Linux kernel previous to this had some very serious shortcomings with respect to certain networking functionality. The older Linux firewalling code does not deal with fragments, has 32-bit counters (on Intel at least), does not allow specification of protocols other than TCP, UDP or ICMP, cannot make large changes atomically, cannot specify inverse rules, has some quirks, and can be tough to manage (making it prone to user error).

Russell's work radically redefined the Linux networking layer and allowed packet filtration to be moved from kernelspace into userspace (making it easier to use, setup, and keep secure).

Finally, the fourth-generation tool, `iptables', and another kernel rewrite occurred in mid-1999 for Linux 2.4. This continued the refining that began in 2.2, and augmented Linux's arsenal of networking tools. The kernel rewrite is known as Netfilter [41].

In this thesis we propose the packet filtering technique to defend against DoS/DDoS attack with the help of IPtables (Linux Firewall).

Chapter 3

Problem Statement

DoS/DDoS attacks are a big threat to the Internet. They decrease the service quality of Internet services. It matters because the Internet is now becoming a critical resource whose disruption has financial implications or even dire consequences on human safety. An increasing number of critical services are using the Internet for daily operation. A DoS attack may not just mean missing out on the latest sports scores or weather. It may mean losing a bid on an item you want to buy or losing your customers for a day or two while under attack , therefore it is important to have means to prevent or at mitigate them. There is a constant arms race between the attackers and the defenders. As soon as there are effective defenses against a certain type of attack, the attackers change tactics finding a way to bypass this defense. In addition, since absolute protection is not feasible, developing effective defense framework involves an often complicated set of trade-offs.

For this purpose, numerous solutions have been proposed as discussed in literature review and Firewall is one of them. By using a dedicated firewall we can block all the IP addresses that are flooding the server to consume its resources.

In this thesis following DoS Attacks have been mitigated using IPtables:

- SYN Flood Attack
- UDP Flood Attack
- ICMP Flood Attack

Also IPtables scripts against following DDoS Attack tools have been developed.

- Trinoo
- Mstream
- Shaft

Chapter 4

DoS/DDoS Defense Mechanisms

Protecting internet service applications from DoS/DDoS attacks is an open research question. Existing defense mechanisms try to mitigate these attacks by filtering the attack traffic to examine all the incoming network packets, and discard the suspected attack packets.

4.1 IP Traceback

IP traceback methods provide the victim's network administrators with the ability to identify the address of the true source of the packets causing DoS attack. IP traceback is vital for restoring normal network functionality as quickly as possible, preventing reoccurrences, and, ultimately, holding the attackers accountable.

Role of IP Address

Ideally, the network traffic used in an attack should include information identifying its source. The Internet protocol (IP) specifies a header field in all packets that contains the source IP address, which would seem to allow for identifying every packet's origin. However, the lack of security features in TCP/IP specifications facilitates IP spoofing, the manipulation and falsification of the source address in the header. The Internet's current routing infrastructure is stateless and largely based on destination addresses, but no entity is responsible for ensuring that source addresses are correct. Thus, an attacker could generate offending IP packets that appear to have originated from almost anywhere. The ability to reliably trace network attacks to their sources might provide some deterrence to risk-averse individuals. IP traceback methods are only the first step toward finding the true identity of the attacker.

IP Traceback approaches

Current IP traceback methods are either reactive or proactive [42]. Reactive measures initiate the traceback process in response to an attack. They must be completed while the attack is active; they're ineffective once it ceases. Most reactive measures require a large degree of ISP cooperation, which leads to extensive administrative burden and difficult legal, and policy issues, thus, effective IP traceback methods should require minimal or

no encroaching on ISP territory. Furthermore, reactive measures are not very effective against multipronged attacks and cannot be used for post-attack analysis. Essentially, reactive measures are more effective for controlled networks than for the Internet.

In contrast, proactive measures record tracing information as packets are routed through the network. The victim can use the resulting traceback data for attack path reconstruction and subsequent attacker identification.

The key requirements for IP traceback methods include:

• Compatibility with existing network protocols,

• Insignificant network traffic overhead,

• Support for incremental implementation,

• Compatibility with existing routers and network infrastructure,

• Effectiveness against DDoS attacks, and

• Minimal overhead in terms of time and resources.

ISP cooperation should not be required, and success should not depend on how long the attack lasts.

Current IP traceback methods fall into four major categories:

- link testing
- logging
- Internet control message protocol (ICMP)-based traceback
- packet marking

4.1.1Link testing

Link-testing methods [43] work by testing network links between routers to determine the origin of the attacker's traffic. Most techniques start from the router closest to the victim and interactively test its incoming (upstream) links to determine which one carries the attack traffic. This process repeats recursively on the upstream routers until reaching the traffic's source as seen in Figure 4.1. Link testing is a reactive method and requires the attack to remain active until the trace is completed.

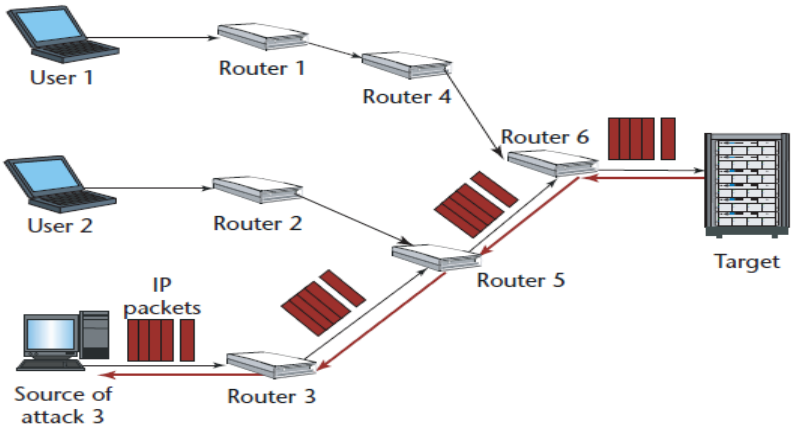

Figure 4.1 Link Testing Approach [44]

4.1.2Logging

Solution to establishing the true origin of offending Internet traffic is to log the packets at key routers throughout the Internet and then use data-mining techniques to extract information about the attack traffic's source as seen in Figure 4.2. Although this solution seems obvious and allows accurate analysis of attack traffic (even after the attack has stopped), its most significant drawbacks include the amount of processing and storage power needed to save the logs.

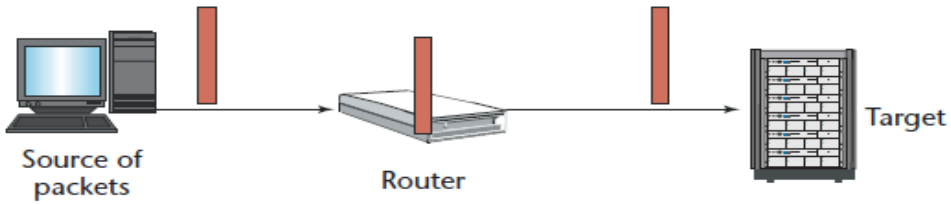

Figure 4.2: Logging packets at Router [44]

Baba et al. [42] proposed an alternate and innovative logging approach. It entailed an overlay network built of sensors that could detect potential attack traffic, tracing agents (tracers) that could log the attack packets on request, and managing agents that could coordinate the sensors and tracers and communicate with each other. This approach attempts to overcome traditional logging method's limitations and shortcomings by selectively logging traffic after an attack is recognized and logging only certain

characteristics, rather than entire packets. The approach increases the speed and requires less storage.

4.1.3 ICMP-based traceback

In July 2000, the Internet Engineering Task Force (IETF) formed a working group to develop ICMP traceback messages based on an approach called *iTrace* [45].This approach used ICMP traceback router-generated messages, which is received by victim in addition to information from regular network traffic. These messages contain partial path information including: information that indicates where the packet came from, when it was sent, and its authentication.

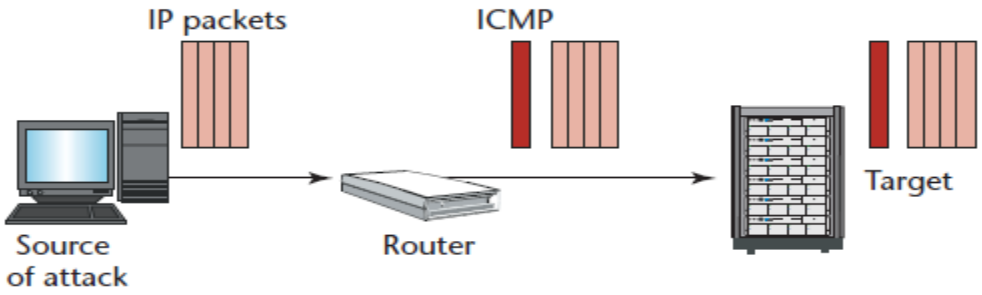

Figure 4.3: ICMP Traceback [44]

The router generates one ICMP packet for every 20,000 packets passing through it on the way to a specific target. The ICMP packet generated is then forwarded to the target
Network managers could piece together these messages to trace a packet's path back to its origin. This is illustrated in Figure 4.3.

4.1.4 Packet Marking

Packet marking methods[46] are characterized by inserting traceback data into the IP packet to be traced to mark the packet on its way through the various routers on the network to the destination host. This approach lets the host machine use markings in the individual packets to deduce the path the traffic has taken. The simplest implementation of packet marking is to use the record route option (as specified in RFC 791) to store router addresses in the IP header's option field [20]. However, this method increases the

packet's length at each router hop and can lead to additional fragmentation. Figure 4.4 shows the general concept of packet marking.

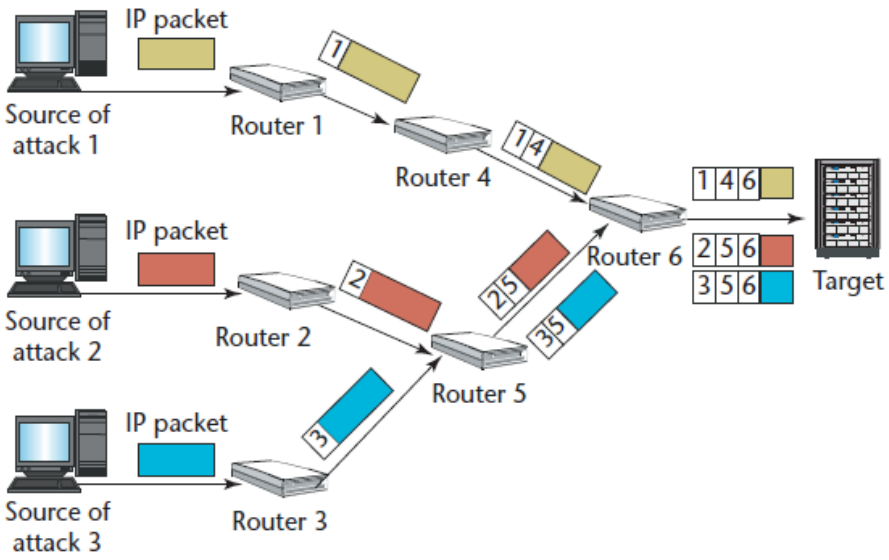

Figure 4.4 Packet Marking [44]

4.2 Network Ingress Filtering

Network ingress filtering is a packet filtering technique used by many Internet service providers to prevent the source address spoofing of Internet traffic, and thus indirectly combat various types of net abuse by making Internet traffic traceable to its source.

Network ingress filtering is a "good neighbor" policy which relies on mutual cooperation between ISPs for their mutual benefit.

Ingress Filtering configures edge routers to ensure that the packets routed with valid source addresses. The packets with spoofed source address that does not belong to this network address will be dropped. However, the effectiveness depends on widespread deployment at access ISPs.

Following figure explains the scenario.

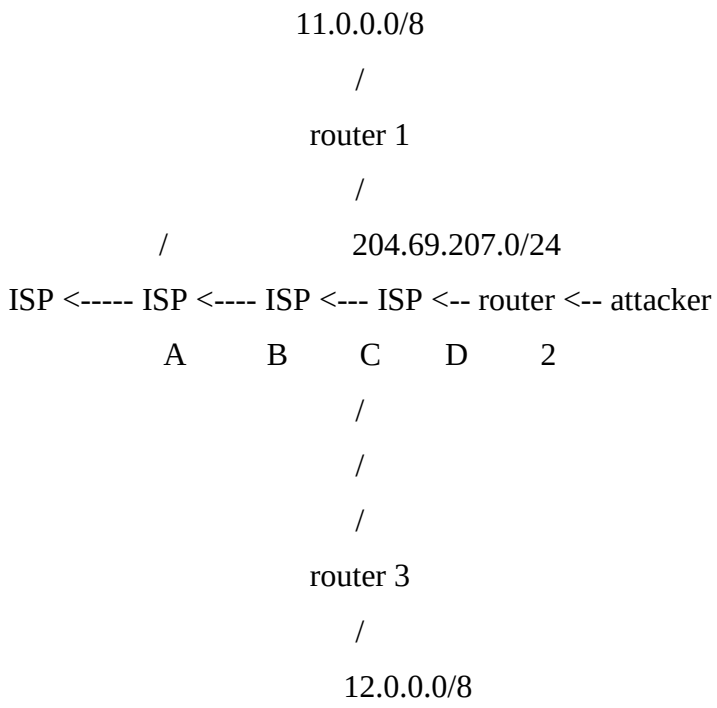

Figure 4.5: Ingress Filtering [31]

As seen in Figure 4.5, the attacker resides within 204.69.207.0/24, which is provided Internet connectivity by ISP D. An input traffic filter on the ingress (input) link of "router 2", which provides connectivity to the attacker's network, restricts traffic to allow only traffic originating from source addresses within the 204.69.207.0/24 prefix, and prohibits an attacker from using "invalid" source addresses which reside outside of this prefix range.

In other words, the ingress filter on "router 2" above would check:

IF packet's source address from within 204.69.207.0/24
THEN forward as appropriate

IF packet's source address is anything else
THEN deny packet

Network administrators should log information on packets which are dropped. This provides a basis for monitoring any suspicious activity [31].

4.3 Intrusion Detection Systems

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies. Intrusion prevention is the process of performing intrusion detection and attempting to stop detected possible incidents. Intrusion detection and prevention systems (IDPS) are primarily focused on identifying possible incidents, logging information about them, attempting to stop them, and reporting them to security administrators. In addition, organizations use IDPS for other purposes, such as identifying problems with security policies, documenting existing threats, and determining individuals from violating security policies. IDPS have become a necessary addition to the security infrastructure of nearly every organization. IDPS typically record information related to observed events, notify security administrators of important observed events, and produce reports. Many IDPS can also respond to a detected threat by attempting to prevent it from succeeding. They use several response techniques, which involve the IDPS stopping the attack itself, changing the security environment (e.g., reconfiguring a firewall), or changing the attack's content [47].

There are two main types of IDS's:

- Host-based Intrusion Detection System (HIDS)
- Network-based Intrusion Detection System (NIDS)

4.3.1 Host based Intrusion Detection System

The HIDS reside on a particular computer and provide protection for a specific computer system. Host intrusion detection systems are installed locally on host machines making it a very versatile system compared to NIDS. HIDS can be installed on many different types of machines namely servers, workstations and notebook computers..HIDS allows for remote monitoring, remote storage of events logs and ability to PUSH agents to new or existing hosts [48]. Figure 4.6 shows the detail model of HIDS.

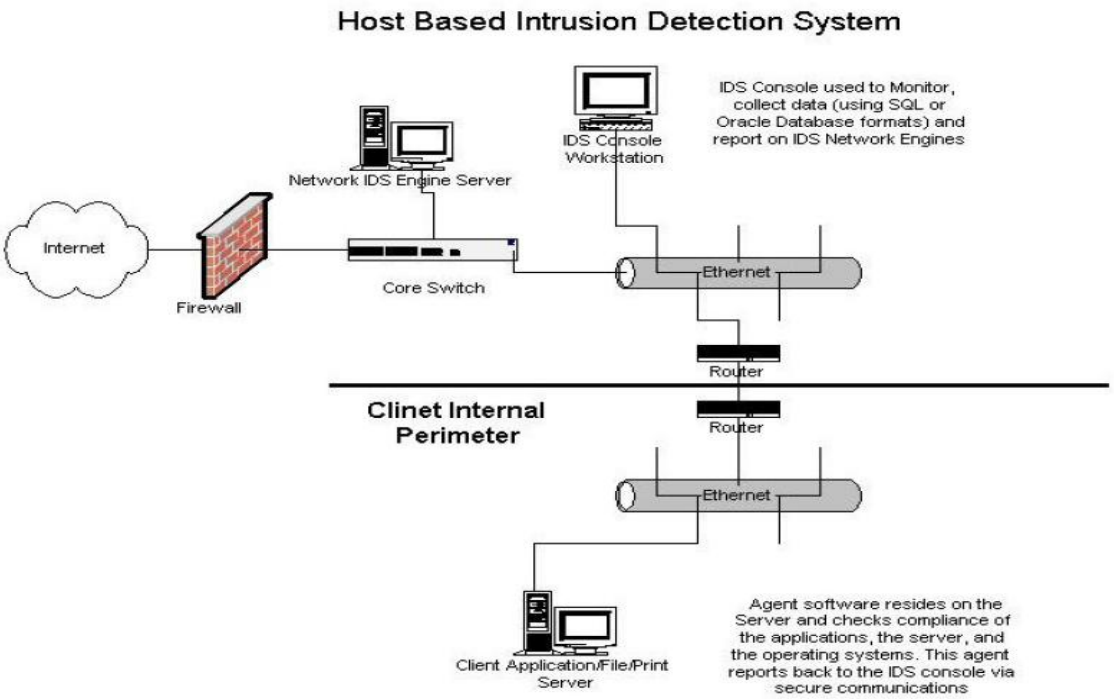

Figure 4.6: Host based Intrusion Detection System [49].

4.3.2 Network based Intrusion Detection System

Network based IDS captures network traffic packets (TCP, UDP) and analyzes the content against a set of rules or signatures to determine if a possible event took place. NIDS monitors packets on the network wire and attempts to discover if a hacker/cracker is attempting to break into a system (or cause a denial of service attack). A typical example is a system that watches for large number of TCP connection requests (SYN) to many different ports on a target machine, thus discovering if someone is attempting a TCP port scan. A NIDS may run either on the target machine who watches its own traffic or on an independent machine promiscuously watching all network traffic (hub, router).

NIDS is network based IDS that do not only deal with packets going to a specific host but all the machines in a network segment benefit from the protection of the NIDS. Figure 4.7 shows the detailed model of NIDS.

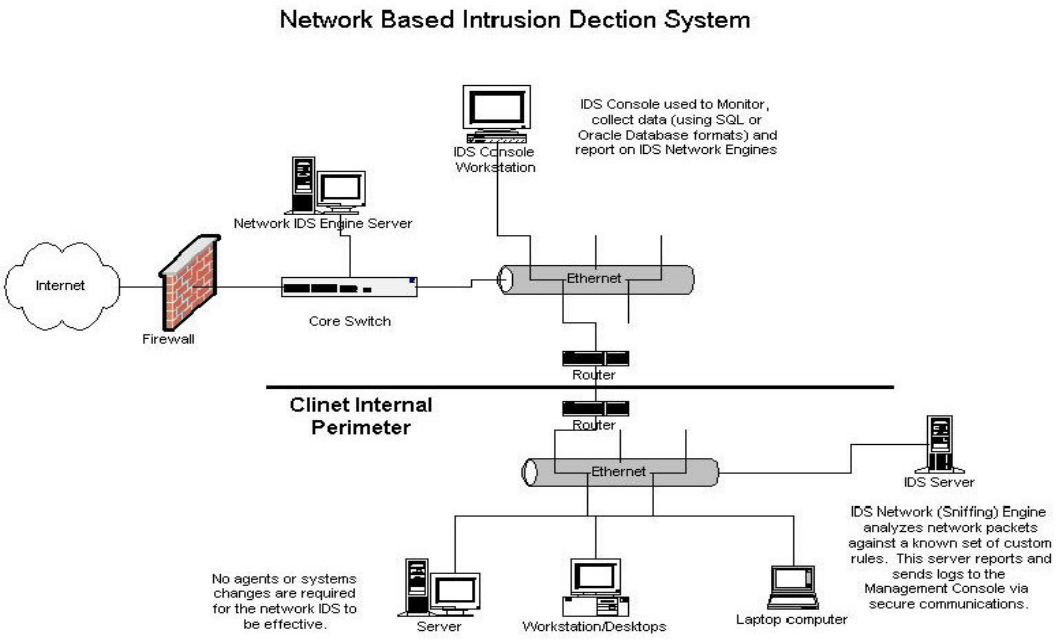

Figure 4.7: Network based Intrusion Detection System [49].

Network-based IDS can also be installed on active network elements, for example on routers. Typical Network Based IDS are Cisco Secure IDS, Hogwash, Dragon, and E-Trust IDS.

IDS products available include RealSecure by Internet Security Systems, IntrusionAlert by Unified Access Communications, SecureNet Pro by Intrusion.com, NetProwler by Axent, and freeware such as Snort, Bro etc. Current DoS/DDoS detection and defense approaches with the help of IDS can be classified into three categories: Proactive Mechanisms, Reactive Mechanisms and Post Attack Analysis.

Proactive defense mechanisms: The motivation for this mechanism is based on the observation that it is hard to detect DDoS attacks. So instead of detecting the attacks by using signatures (attack pattern) or anomaly behavior, this mechanism try to improve the reliability of the global Internet infrastructure by adding extra functionality to Internet components to prevent attacks and vulnerability exploitation. The primary goal is to make the infrastructure immune to the attacks and to continue to provide service to normal users under extreme conditions.

Reactive defense mechanisms using available IDS: This mechanism typically deploys third-party Intrusion Detection Systems (IDS) to obtain attack information and take action based on this information. Consequently their usefulness depends on the capability of the IDS systems. Different strategies are used based on the assumptions made by the IDS systems. If the IDS system can detect the DDoS attack packets accurately, filtering mechanism are used, which can filter out the attack stream completely, even at the source network. If the IDS cannot detect the attack stream accurately, rate limiting is used. This mechanism imposes a rate limit on the stream that is characterized as malicious by the IDS.

Post attack analysis: The purpose of post attack analysis is to either look for attack patterns that will be used by IDS or identify attackers using packet tracing. The goal of packet tracing is to trace Internet traffic back to the true source (not spoofed IP address). As attackers change their strategy frequently, analyzing huge amounts of traffic logs is time consuming and useless in detecting new attacks. Trace back mechanism can help to identify zombies, however, it is impractical to defend against DDoS attacks for the following reasons. First, during a DDoS attack, the attacker will control thousands of zombies (numbers will increase in the future) to launch an attack. As a result, identifying these zombies is expensive and infeasible. Second, since different network administrators control different sections of the global Internet, it would be difficult to determine who would be responsible for providing trace back information [50].

4.4 Firewalls

A firewall is a logical object (hardware and/or software) within a network infrastructure which prevents communications forbidden by the security policy of an organization from taking place, analogous to the function of firewalls in building construction. Often a firewall is also referred to as a packet filter. The basic task of a firewall is to control traffic between different zones of trust and/or administrative authorities. Typical zones of trust include the Internet (a zone with no trust) and an internal network (a zone with high trust). The ultimate goal is to provide controlled connectivity between zones of differing trust levels through the enforcement of a security policy and a connectivity model based on the least privilege principle. Proper configuration of firewalls demands skill from the

administrator. It requires considerable understanding of network protocols and of computer security. Small mistakes can lead to a firewall configuration worthless as a security tool and, in extreme situations, fake security where no security at all is left.

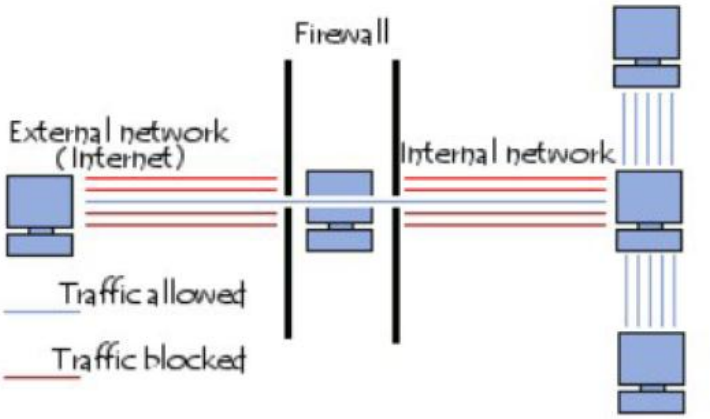

Figure 4.8: Firewall System [51].

As seen in Figure 4.8, firewall can protect internal network from the external network by accept/deny the traffic according to the rules specified in the list. A firewall is a system that protects a computer or a computer network against intrusions coming from a third-party network (generally the Internet). A firewall is a system that filters data packets that are exchanged over the network. Therefore, it is a filtering gateway that comprises at least the following network interfaces:

- An interface for the network being protected (internal network)
- An interface for the external network

Advantages of Firewalls

Some of the advantages of firewall are :

- They can stop incoming requests to inherently insecure services, e.g. you can disallow rlogin, or RPC services such as NFS.
- They can control access to other services e.g. bar callers from certain IP addresses, filter the service operations (both incoming and outgoing)

- They are more cost effective than securing each host on the corporate network since there are often only one or a few firewall systems to concentrate on.

Disadvantages of Firewalls

Firewalls are not the be all and end all of network security. They do have some disadvantages, such as:

- They are a central point for attack, and if an intruder breaks through the firewall they may have unlimited access to the corporate network.
- They may restrict legitimate users from accessing valuable services, for example, corporate users may not be let out onto the Web, or when working away from home a corporate user may not have full access to the organization's network.
- They can be a bottleneck to throughput, since all connections must go via the firewall system.
- The biggest disadvantage of a firewall is that it gives no protection against the inside attacker. Since most corporate computer crime is perpetrated by internal users, a firewall offers little protection against this threat.

Firewalls, Layers and Models

The working of firewall in different layers and models are seen as:

ISO 7 Layer Model	Internet 5 Layer Model	Firewalls
Application (7)	Application (5)	Proxy Service
Transport (4) Network (3)	TCP/UDP (4) IP/ICMP (3)	Packet Filtering Router/ Packet Screening Router Stateful Inspection

Table 4.1: Firewalls Layers and Models [51].

As seen in Table 4.1 ISO uses a 7 layer model for Open Systems Interconnection, whereas the Internet can be regarded as having a 5 layer model. Firewall systems are usually placed at layers 3, 4 and 5 of the Internet model, (3, 4 and 7 of the ISO model).

Their purpose is to control access to and from a protected network. Firewall can be placed between any two networks, for example between a corporate business network and its R&D network. In general, a firewall is placed between a high security domain and a lower security domain. A firewall system operating at layers 3 and 4 is sometimes called a packet filtering router or a screening router. Its purpose is to filter IP and ICMP packets and TCP/UDP ports. The router will have several ports and be able to route and filter the packets according to the filtering rules. Packet filters can also be built in software and run on dual homed PCs, but whilst these can filter packets they are not able to route them to different networks. A firewall at layer 5 Internet (7 ISO) is sometimes called a bastion host, application gateway, proxy server or guardian system. Its purpose is to filter the service provided by the application [51].

Types of Firewalls

Firewalls are classified into three basic types:

4.4.1 Stateless Packet Filtering

A firewall system operates on the principle of simple packet filtering, or stateless packet filtering. It analyses the header of each data packet (datagram) exchanged between an internal network computer and an external computer. Thus, the data packets exchanged between an external network computer and an internal network computer pass through the firewall and contain the following headers, which are systematically analyzed by the firewall:

- The IP address of the computer sending the packets
- The IP address of the computer receiving the packets
- The type of packet (TCP, UDP, etc.)
- The port number (port is a number associated with a service or a network application).

The IP addresses contained in the packets allow you to identify the computer that is sending the packets and the target computer, while the type of packet and the port number indicate the type of service being used. Table 4.2 below gives examples of firewall rules.

Rule	Action	Source IP	Target IP	Protocol	Source Port	Target Port
1	Accept	192.168.1.2	192.168.1.4	TCP	Any	25
2	Accept	Any	192.168.1.4	TCP	Any	80
3	Accept	192.168.1.2/24	Any	TCP	Any	80
4	Deny	Any	Any	Any	Any	Any

Table 4.2: Firewall rules [53]

Table 4.2 describes four firewall rules for allowing and denying the network packet based on Source IP, Target IP, Protocol, Source Port, and Target Port.

Recognized ports (whose numbers are between 0 and 1023) are associated with ordinary services (e.g. ports 25 and 110 are associated with e-mail and port 80 with the Web). Most firewall devices are at least configured to filter communications according to the port being used. It is generally recommended to block all ports that are not essential (depending on the security policy in place). For example, port 23 is often blocked by default by firewall devices because it corresponds to the TELNET protocol, which allows a person to emulate terminal access to a remote machine in order to remotely execute commands. The data exchanged over TELNET are not encrypted, which means that a hacker is likely to listen to the network and steal any unencrypted passwords. Administrators generally prefer the SSH protocol, which has a reputation for being safe and provides the same functionalities as TELNET.

4.4.2 Dynamic Filtering

Stateless Packet Filtering only attempts to examine the IP packets independently, which corresponds to level 3 of the OSI model. But most connections are supported by TCP protocol, which manages sessions, in order to ensure that exchanges take place smoothly. In addition, many services (e.g. FTP) initiate a connection on a static port but dynamically (i.e. randomly) open a port in order to establish a session between the machine acting as a server and the client machine. Thus, with stateless packet filtering it

is impossible to anticipate which ports should be authorized and which should be prohibited. To remedy this situation, the system of dynamic packet filtering is based on the inspection of layers 3 and 4 of the OSI model, allowing for full monitoring of the transactions between the client and the server. The term for this is "stateful inspection" or "stateful packet filtering". A "stateful inspection" firewall device is able to ensure the monitoring of exchanges, meaning that it takes into account the status of previous packets when defining filtering rules.

4.4.3 Application proxies

Application filtering allows filtering communications application by application.

It operates at level 7 (the application layer) of the OSI model, contrary to simple packet filtering (level 4). Application filtering implies knowledge of the applications on the network and notably of the way in which it structures the exchanged data (ports, etc.). A firewall performing application filtering is generally called an "application gateway" (or a "proxy") because it acts as a relay between two networks by intervening and performing a thorough evaluation of the content in the exchanged packets. Therefore, the proxy represents an intermediary between the internal network's computers and the external network, putting the attacks in their place. Moreover, application filtering allows for the headers that precede application messages to be destroyed, which provides an additional level of security. This type of firewall is highly effective and ensures good network protection if it is correctly run.

On the other hand, detailed analysis of application data requires a lot of computing power, which often means slowed communications because each packet must be thoroughly analyzed [52].

4.4.4 Firewall using IPtables

IPtables provides comparatively higher speed and reliability than the other firewall tools. Being a Linux product, its integration with the OS is also more robust and reliable. It keeps a stateful track of each connection passing through it and tries to anticipate future actions. Its capacity of filtering packets on the basis of MAC address makes it a formidable security system. It can filter out attacks using malformed packets and can

restrict access from locally attached servers to other networks in spite of their valid IP addresses. Network address translation (NAT) with masquerading capability of IPtables helps it to hide internal network IP sub networks behind one or a small poll of external IP addresses. NAT and masquerading enables the firewall system to open and close ports on the gateway. NAT and masquerading is very much helpful for network traffic analysis and devising relevant security measures. The rate limiting feature of IPtables can block attacks even from some types of DoS (denial of service) attacks [54].

IPtables is a current Linux Firewall mechanism and a successor of ipfilter and ipchains. The primary purpose is packet filtering based on header fields, e.g., IP addresses, TCP and UDP ports, and TCP flags. Originally, the most popular firewall/NAT package running on Linux was ipchains, but it had a number of shortcomings. To rectify this, the Netfilter organization decided to create a new product called IPtables, giving it such improvements as:

- Better integration with the Linux kernel with the capability of loading IPtables-specific kernel modules designed for improved speed and reliability.
- Stateful packet inspection. This means that the firewall keeps track of each connection passing through it and in certain cases will view the contents of data flows in an attempt to anticipate the next action of certain protocols. This is an important feature in the support of active FTP and DNS, as well as many other network services.
- Filtering packets based on a MAC address and the values of the flags in the TCP header. This is helpful in preventing attacks using malformed packets and in restricting access from locally attached servers to other networks in spite of their IP addresses.
- System logging that provides the option of adjusting the level of detail of the reporting.
- Better network address translation [54].

Targets and jumps in IPtables

Each firewall rule inspects each IP packet and then tries to identify it as the target of some sort of operation. Once a target is identified, the packet needs to jump over to it for further processing.

Descriptions of the most commonly used targets

Target	**Description**
ACCEPT	IPtables stops further processing. The packet is handed over to the end application or the operating system for processing.
DROP	IPtables stops further processing. The packet is blocked.
LOG	The packet information is sent to the syslog daemon for logging. IPtables continues processing with the next rule in the table. As you can not log and drop at the same time, it is common to have two similar rules in sequence. The first will log the packet, the second will drop it.
REJECT	Works like the DROP target, but will also return an error message to the host sending the packet that the packet was blocked.
DNAT	Used to do destination network address translation. .i.e. rewriting the destination IP address of the packet.
SNAT	Used to do source network address translation rewriting the source IP address of the packet. The source IP address is user defined.
MASQUERADE	Used to do Source Network Address Translation. By default the source IP address is the same as that used by the firewall's interface.

Table 4.3: Target and Jumps [54].

How does a Packet Flow?

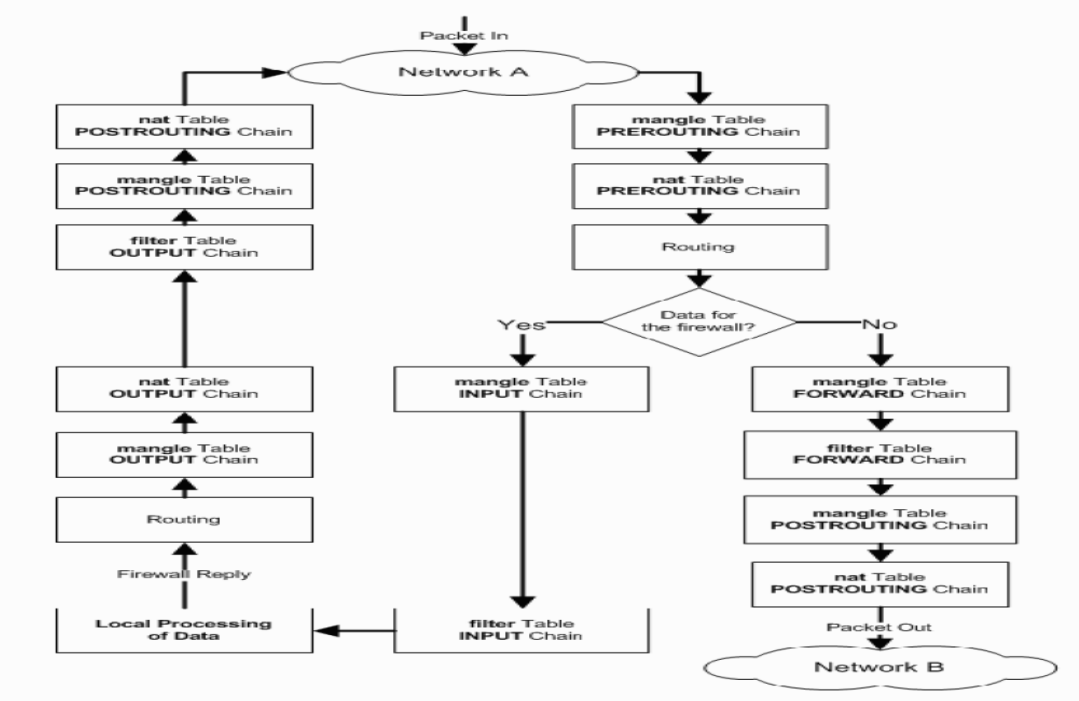

Figure 4.9: Packet Flow Diagram [54].

As seen in figure 4.9, a TCP packet from the Internet arrives at the firewall's interface on Network A to create a data connection. The packet is first examined by rules in the mangle table's PREROUTING chain, if any. It is then inspected by the rules in the NAT table's PREROUTING chain to see whether the packet requires DNAT. It is then routed.

If the packet is destined for a protected network, then it is filtered by the rules in the FORWARD chain of the filter table and, if necessary, the packet undergoes SNAT in the POSTROUTING chain before arriving at Network B. When the destination server decides to reply, the packet undergoes the same sequence of steps. Both the FORWARD and POSTROUTING chains may be configured to implement quality of service (QoS) features in their mangle tables, but this is not usually done in SOHO environments.

If the packet is destined for the firewall itself, then it passes through the mangle table of the INPUT chain, if configured, before being filtered by the rules in the INPUT chain of the filter table before. If it successfully passes these tests then it is processed by the intended application on the firewall.

At some point, the firewall needs to reply. This reply is routed and inspected by the rules in the OUTPUT chain of the mangle table, if any. Next, the rules in the OUTPUT chain of the NAT table determine whether DNAT is required and the rules in the OUTPUT chain of the filter table are then inspected to help restrict unauthorized packets. Finally, before the packet is sent back to the Internet, SNAT and QoS mangling is done by the POSTROUTING chain [54].

Packet sniffing tools have been available from the early days of networked computing environments. The tools are powerful software, which facilitate troubleshooting for network administrators. However, in the hands of a malicious third party, they are a devastating hacking tool, which can be used to glean passwords and other sensitive information from a LAN.

Chapter 5

Implementation Details and Experimental Results

5.1 Implementation steps

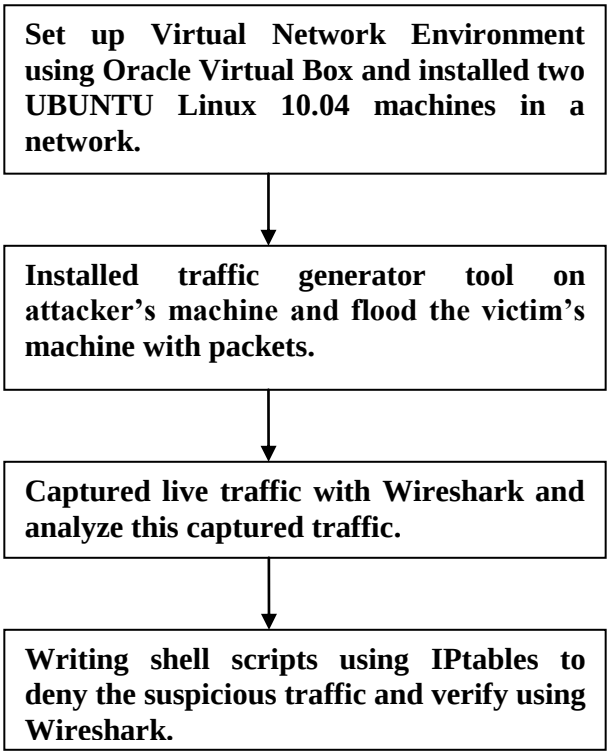

Figure 5.1: Implementation steps

Step1: To establish a segregate network using virtualization. Oracle Virtual Box is used to establish a segregate network and two UBUNTU 10.04 operating systems are installed on it.

Step2: Packet sniffer Wireshark is installed on 1st machine (Victim's machine) from command-line using following command

apt-get install wireshark

Step3: Packet generator Hping3 or Mausezahn is installed on 2nd machine (Attacker's machine) using following command

apt-get install Hping3

Step 4: Operating Hping3 to generate high volume traffic towards victim's machine so as to perform DoS attack on it.

Step 5: Operating Wireshark (as root) and capturing network packets on eth0 interface on victim's machine in order to detect DoS attack**.**

Step 6: Writing of Shell script using IPtables.

Step 7: After blocking traffic using IPtables again capturing and analyzing of Live Traffic using Wireshark.

5.2 Software Setup and Testing

All the results are checked on machine Intel® CeleronM® CPU 440 @ 1.86 GHz along with 1.5 GB RAM space. The Operating System used was Ubuntu-10.04 Linux operating system which was installed using virtualization software.

Oracle Virtual Box is used to establish a segregate network of two UBUNTU 10.04 operating systems . One of them is attacker's machine with IP address 192.168.1.3 and the other one is victim's machine with IP address 192.168.1.2 as seen in Figure 5.2.

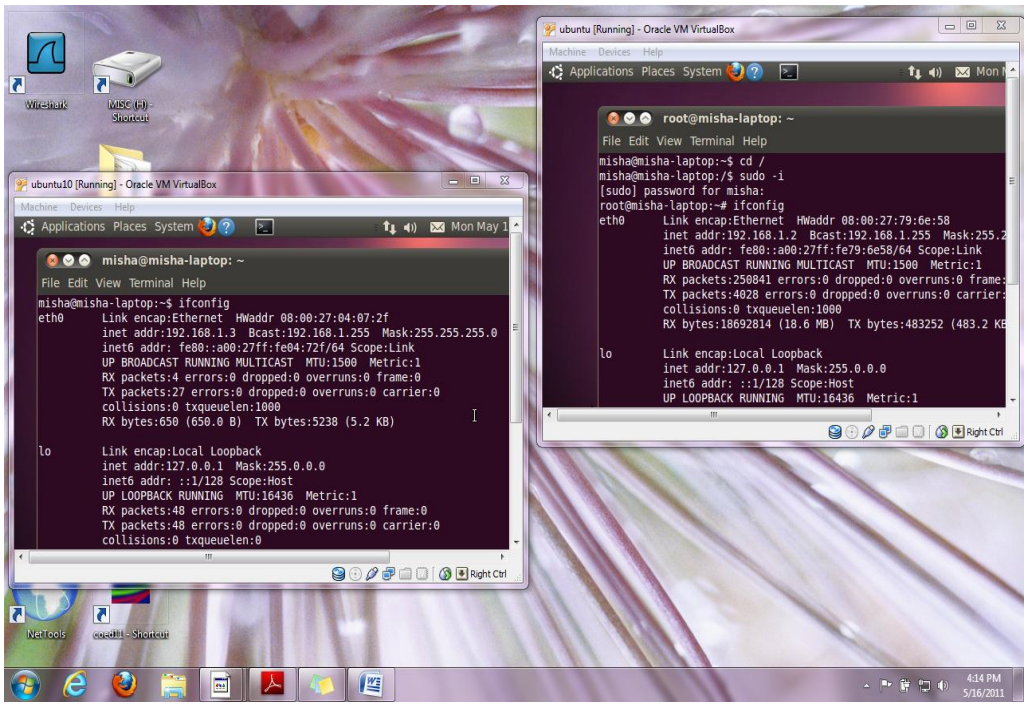

Figure 5.2: Virtual Network Environment

5.3 Experiment and Results

5.3.1 Module 1- SYN Flood Attack

Step 1: Flooding the victim's machine by running following Hping command from attacker's machine :

 # hping3 --flood –S –p 80 192.168.1.2

Description:

--flood flag sends the packet as fast as possible

-S flag sets the SYN flag on in TCP mode

-p 80 sends the packet to port 80 on victim's machine (192.168.1.2)

Step 2: Capturing and Analyzing Traffic on Victim's machine

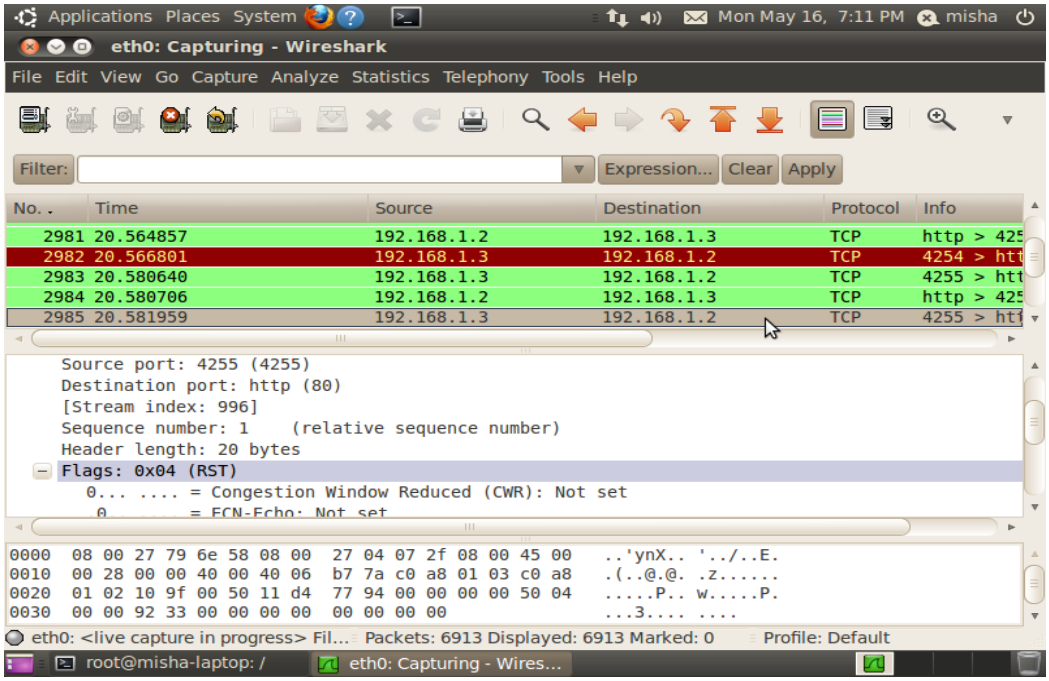

Figure 5.3: Wireshark Output showing SYN Flood Attack

As seen in Figure 5.3 victim's machine (192.168.1.2) is responding to SYN packet by sending back packets with SYN, ACK flags set but attacker's machine (192.168.1.3) is not participating three way handshake by sending back ACK ,instead it is sending RST flag set packet thereby resulting in half open connection .When thousands of such connections are made in few seconds victim's resources will get exhausted in no time.

This scenario is also depicted in the following flow graph (Figure 5.4) and I/O graph (Figure 5.5).

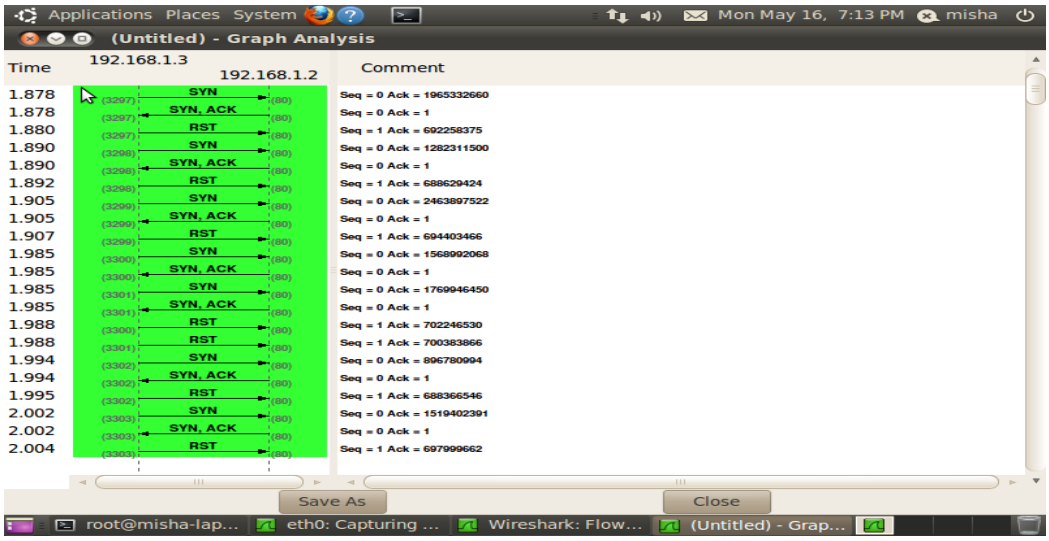

Figure 5.4: TCP Flow graph showing SYN Flood Attack

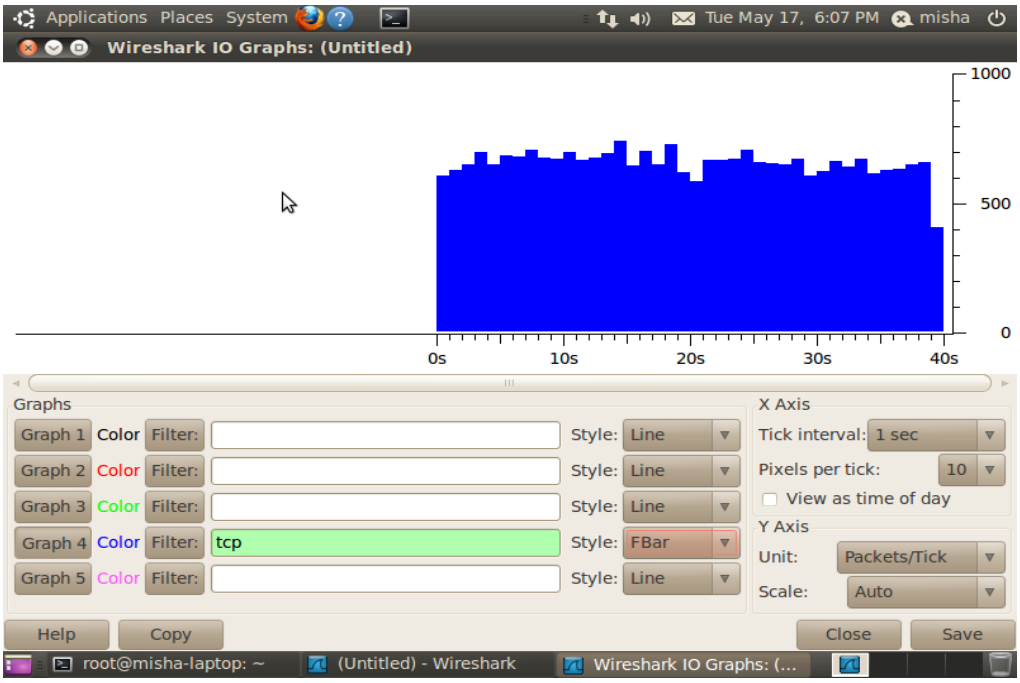

Figure 5.5: I/O Graph showing SYN Flood Attack

Step 3: Writing IPtables script to defend against SYN Flood Attack

Script: Syn_flood.sh

#!/bin/bash

IPTABLES=/sbin/iptables

 # start by flushing the rules

iptables –F

 # Stop SYN Flooding

iptables -N syn_flood

iptables -A INPUT -p tcp --syn -j syn_flood

iptables -A syn_flood -m limit --limit 1/s --limit-burst 3 -j RETURN

iptables -A syn_flood -j DROP

 # List Rules

iptables –L –v

Figure 5.6 shows the list of rules after running the script

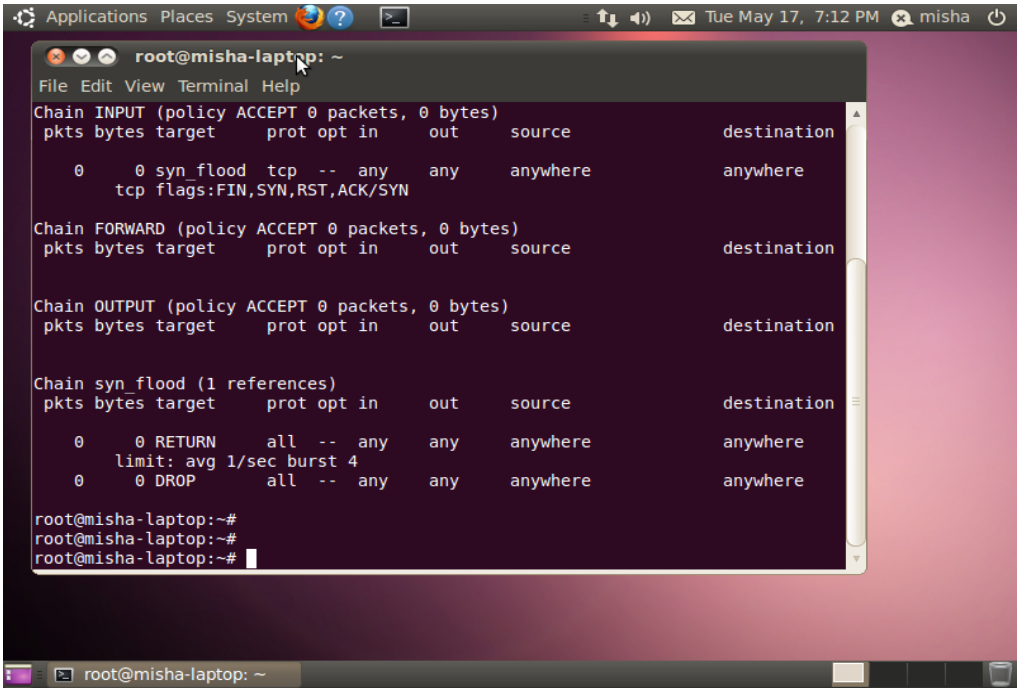

Figure 5.6: List of Iptables Rules against SYN Flood Attack

Step 4: Recapture packets using wireshark to test the working of syn_flood.sh script

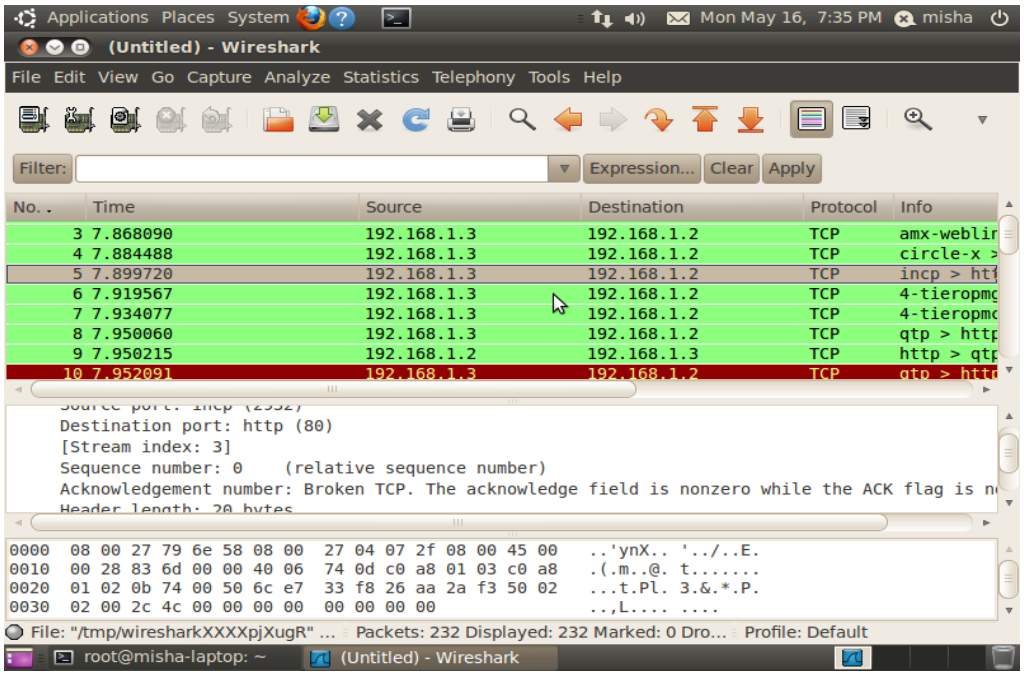

Figure 5.7: Wireshark output after SYN Flood Attack protection

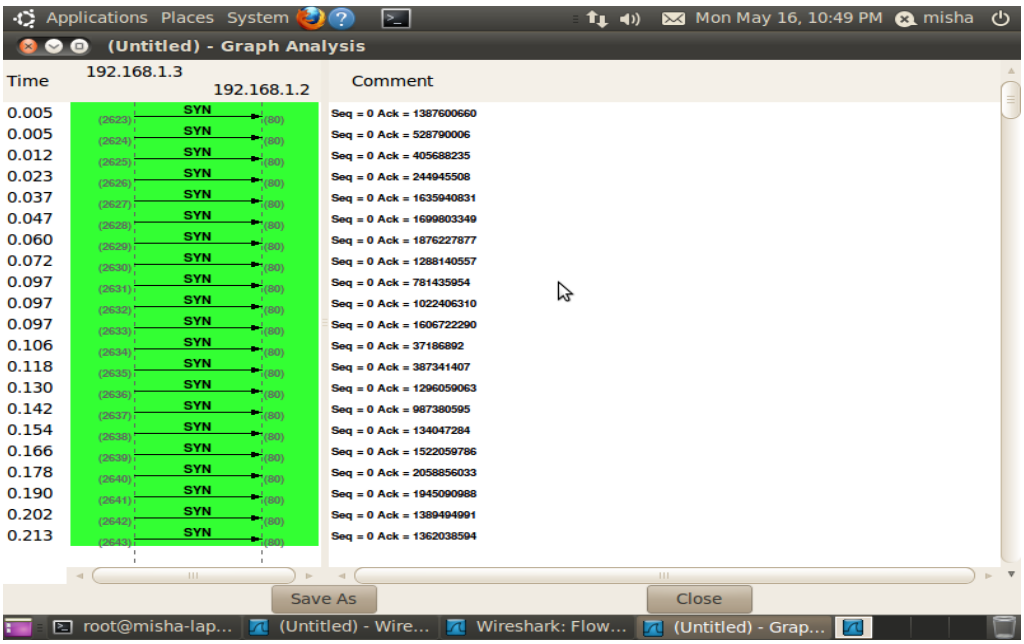

Figure 5.8: TCP Flow Graph after SYN Flood Attack protection

As seen in Figure 5.7 and Figure 5.8 , attacker is sending SYN packets continuously but victim's machine is not responding by sending SYN,ACK packets as all the packets are being dropped by the firewall according to the IPtables rules seen in Figure 5.6.

5.3.2 Module 2- UDP Flood Attack

Step 1: Flooding the victim's machine by running following Hping command from attacker's machine :

 # hping3 –p 80 –i u1000 --udp 192.168.1.2

Description:

-p 80 sends the packet to port 80 on victim's machine (192.168.1.2)

-i u1000 sets the interval between packets as 100 packets per second.

--udp flag sets the udp mode

Step 2: Capturing and Analyzing Traffic on Victim's machine

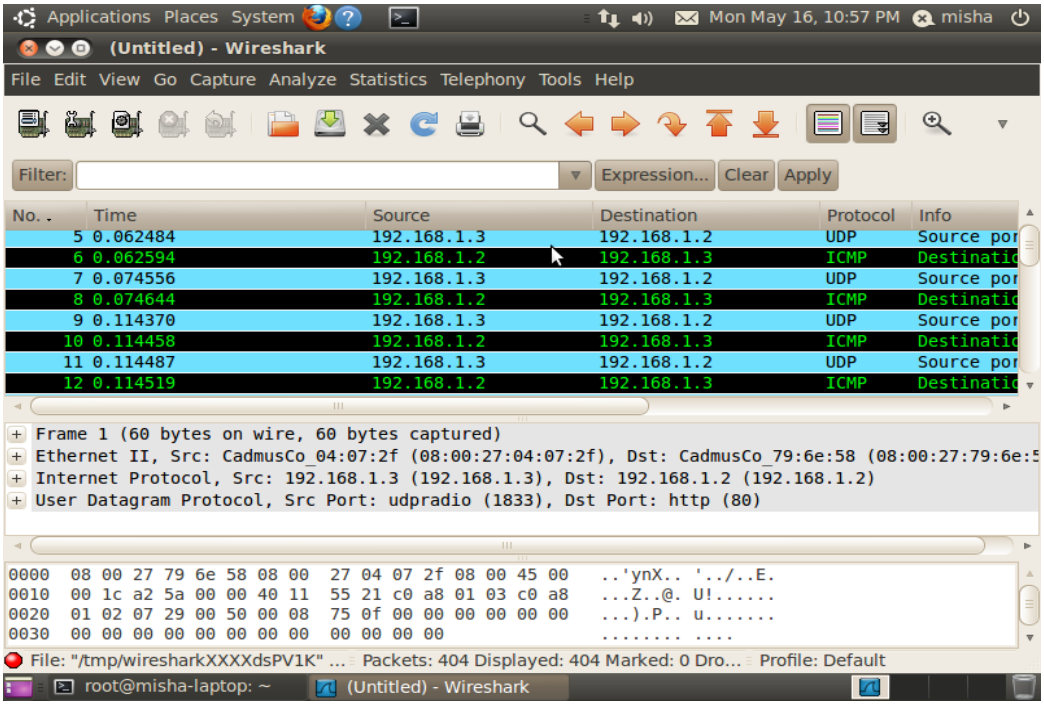

Figure 5.9: Wireshark Output showing UDP Flood Attack

As seen in Figure 5.9 victim's machine (192.168.1.2) is responding with ICMP port unreachable since there is no application running on attacker's machine which sent UDP packet. In this way all of the resources of victim's machine are consumed and legitimate

requests will not be served as victim will be busy in serving attacker's invalid requests. This scenario is also depicted in the following flow graph (Figure 5.10) and I/O Graph (Figure 5.11)

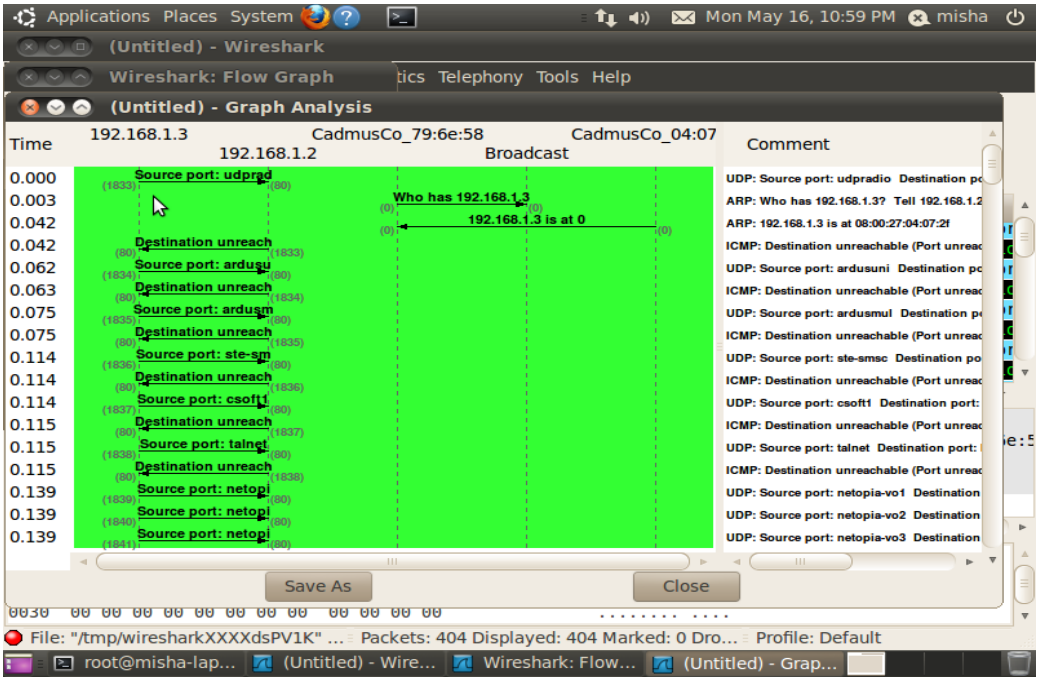

Figure 5.10: Flow graph showing UDP Flood Attack

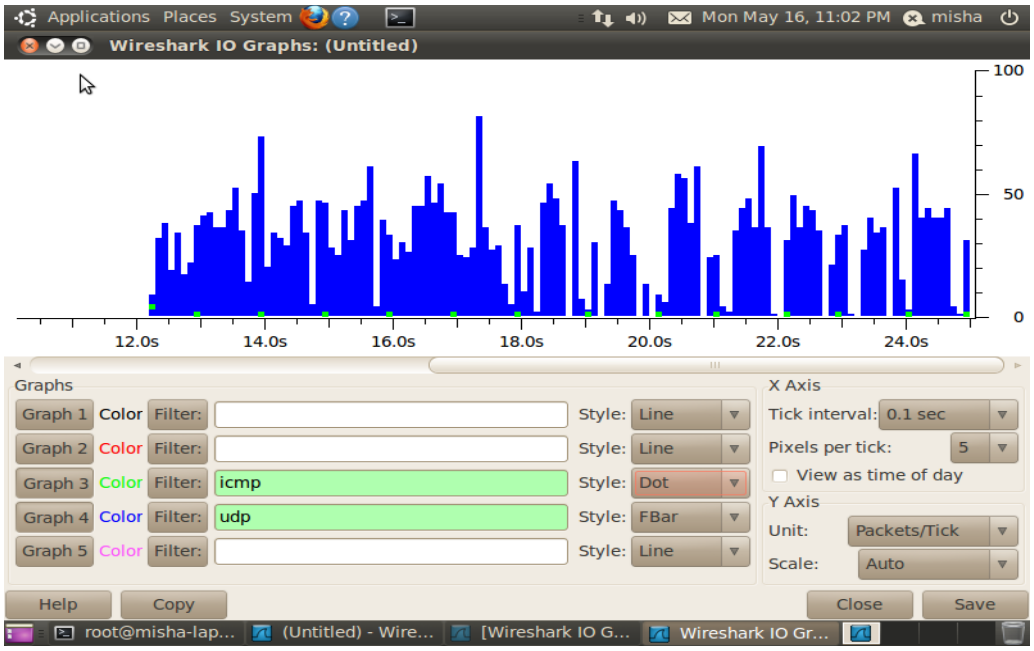

Figure 5.11 I/O Graph showing UDP Flood Attack

Step 3: Writing IPtables script to defend against UDP Flood Attack

Script: Udp_flood.sh

#!/bin/bash

IPTABLES=/sbin/iptables

 # start by flushing the rules

iptables –F

 # Stop UDP Flooding

iptables -N udp_flood

iptables -A INPUT -p udp -j udp_flood

iptables -A udp_flood -m state –state NEW –m recent –update –seconds 1 –hitcount 10 -j RETURN

iptables -A udp_flood -j DROP

 # List Rules

iptables –L –v

Figure 5.12 shows the list of rules after running the script

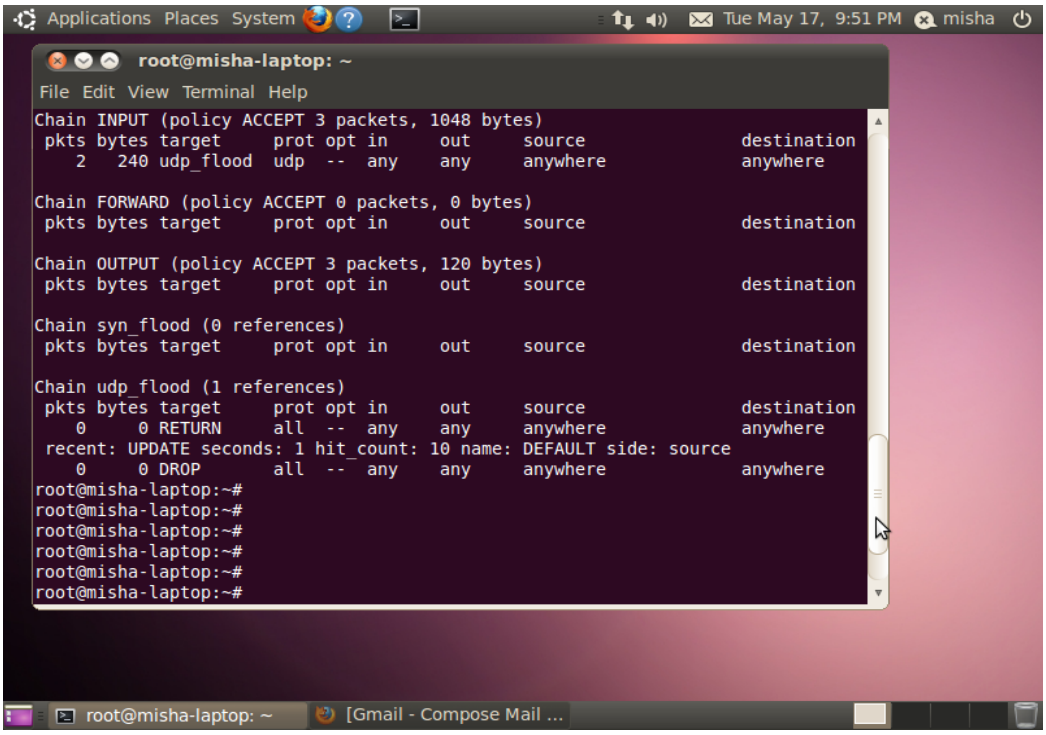

Figure 5.12 List of IPtables Rules against UDP Flood Attack

Step 4: Recapture packets using wireshark to test the working of udp_flood.sh script.

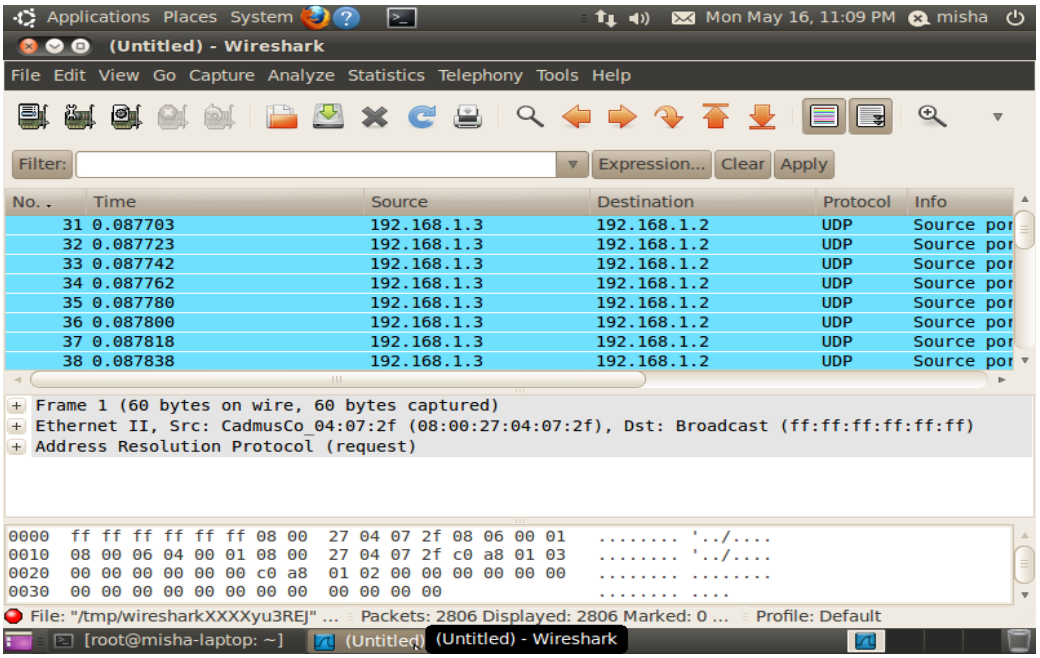

Figure 5.13 Wireshark output after UDP flood Attack protection

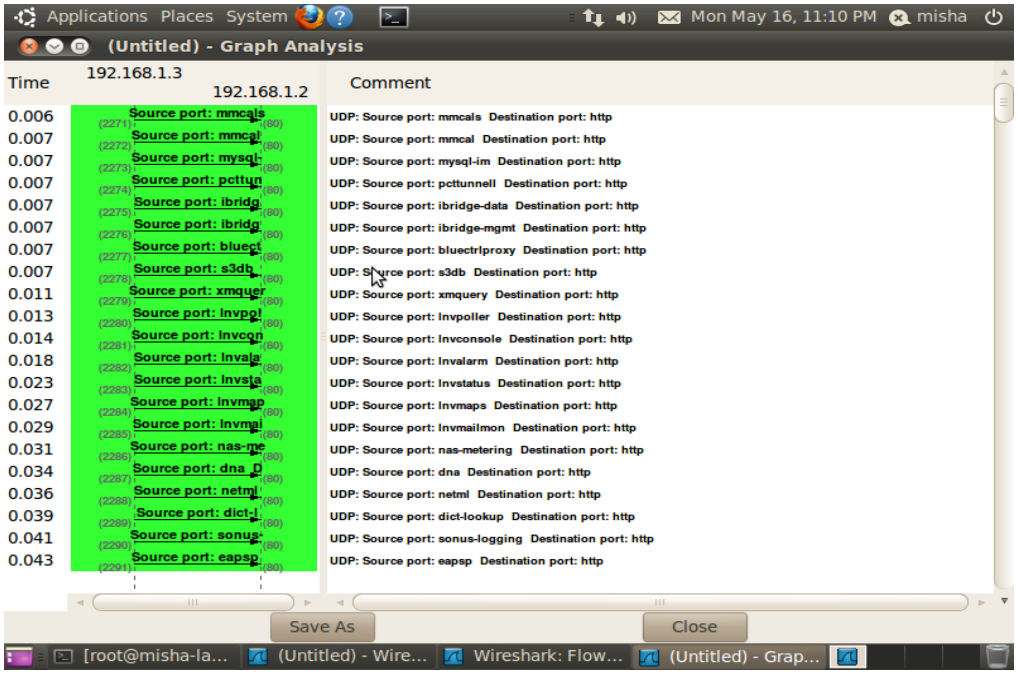

Figure 5.14 UDP Flow graph after UDP Flood Attack protection

As seen in Figure 5.13 and Figure 5.14, attacker is sending UDP packets continuously but victim's machine is not responding by sending ICMP port Unreachable packets as all the packets are being dropped by the firewall according to the IPtables rules seen in Figure 5.12.

5.3.3 Module 3- ICMP Flood Attack

Step 1: Flooding the victim's machine by running following Hping command from attacker's machine :

 # hping3 –p 80 --flood --icmp 192.168.1.2

Description:

-p 80 sends the packet to port 80 on victim's machine (192.168.1.2)

--flood flag sends the packet as fast as possible

--udp flag sets the icmp mode

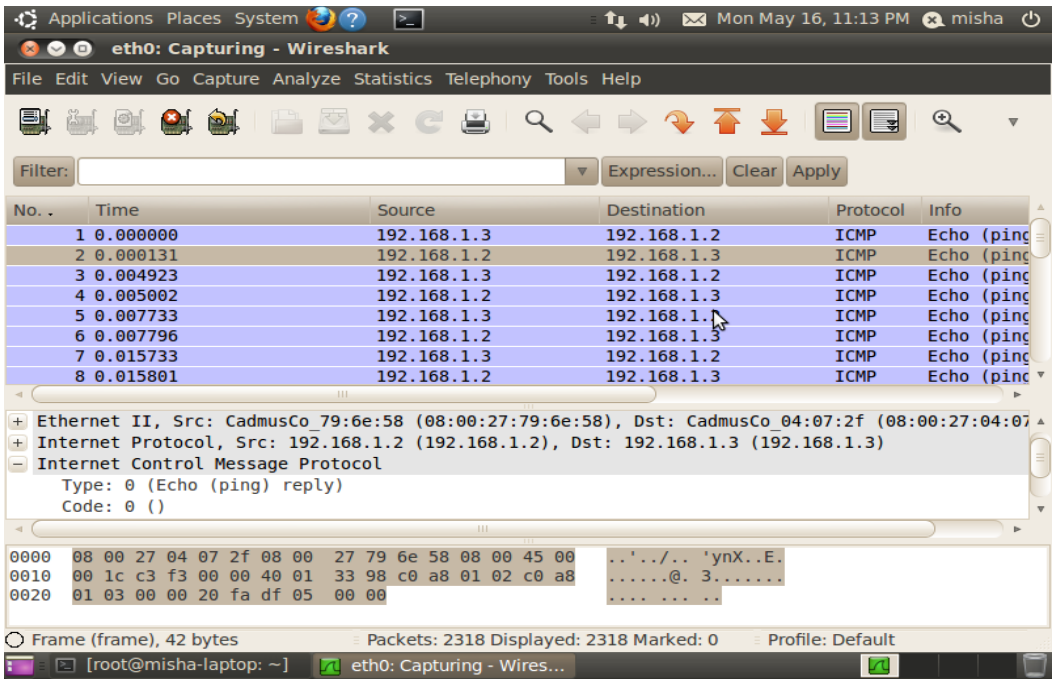

Figure 5.15: Wireshark output showing ICMP flood attack

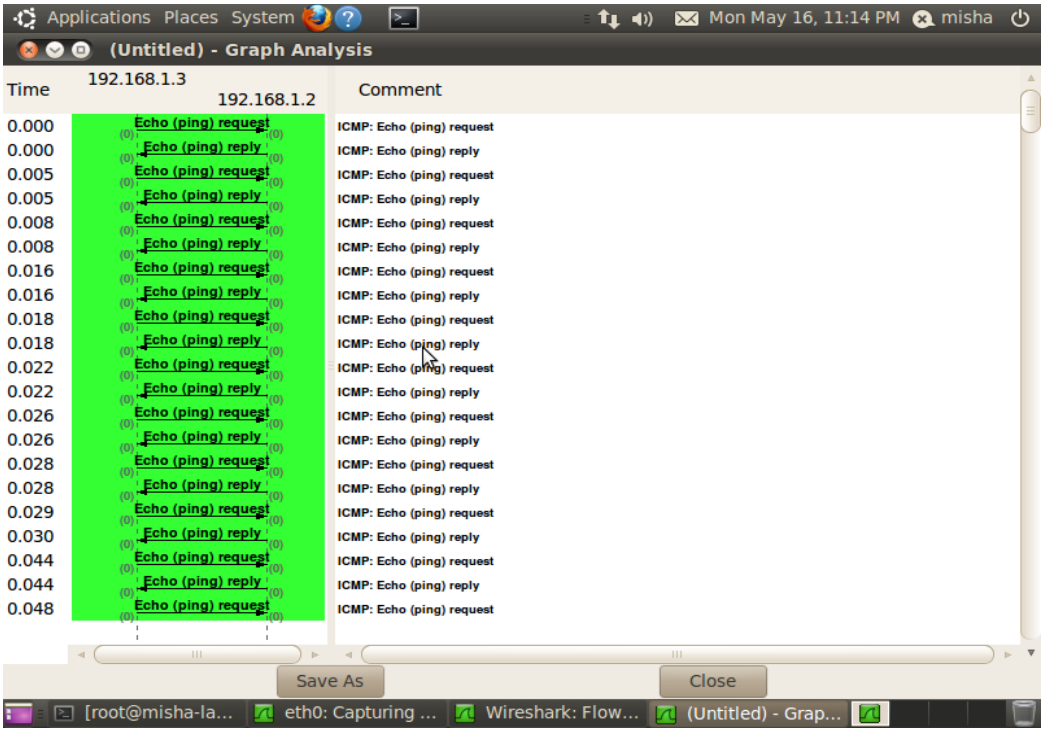

Figure 5.16: Flow Graph showing ICMP Flood attack

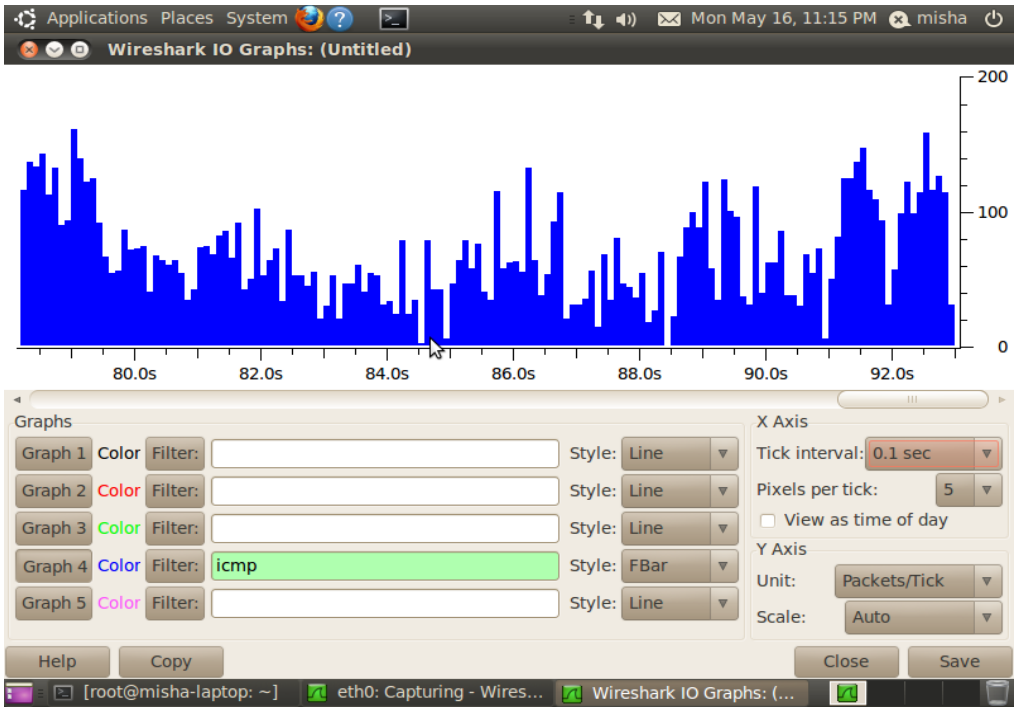

Figure 5.17: I/O Graph showing ICMP Flood Attack

Step 3: Writing IPtables script to defend against ICMP Flood Attack

Script: icmp_flood.sh

```
#!/bin/bash
IPTABLES=/sbin/iptables
# start by flushing the rules
iptables –F
# Stop ICMP Flooding
iptables -N icmp_flood
iptables -A INPUT -p icmp -j icmp_flood
iptables -A icmp_flood -m limit --limit 1/s --limit-burst 3 -j RETURN
iptables -A icmp_flood -j DROP
# List Rules
iptables –L –v
```

Figure 5.18 shows the list of rules after running the script

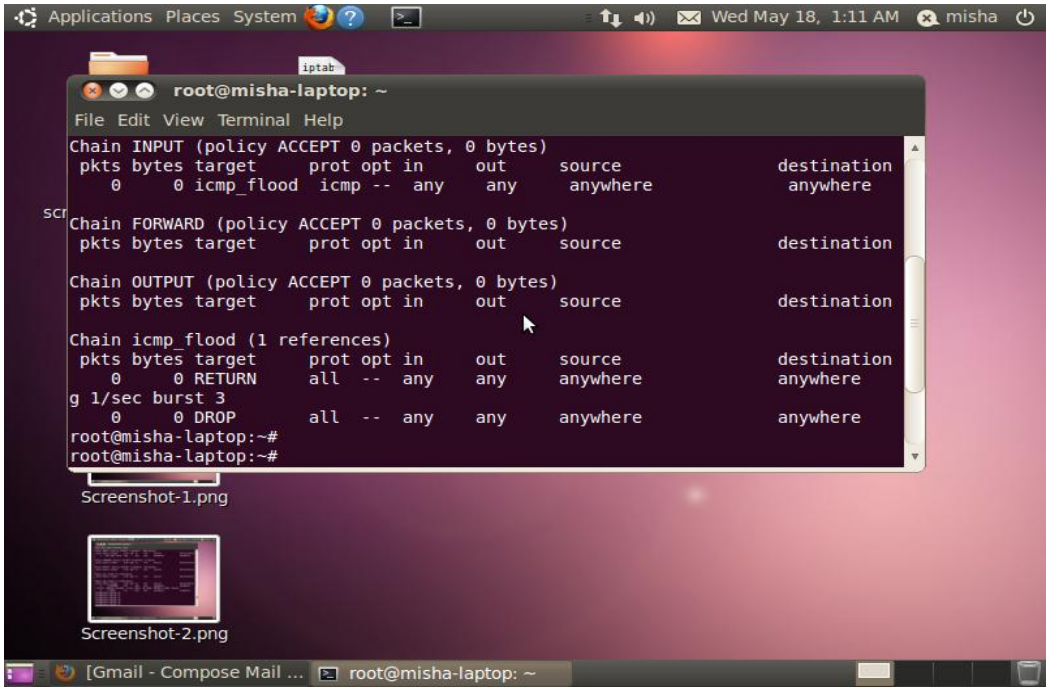

Figure 5.18 List of Iptables rules against ICMP Flood attack

Step 4: Recapture packets using wireshark to test the working of icmp_flood.sh script

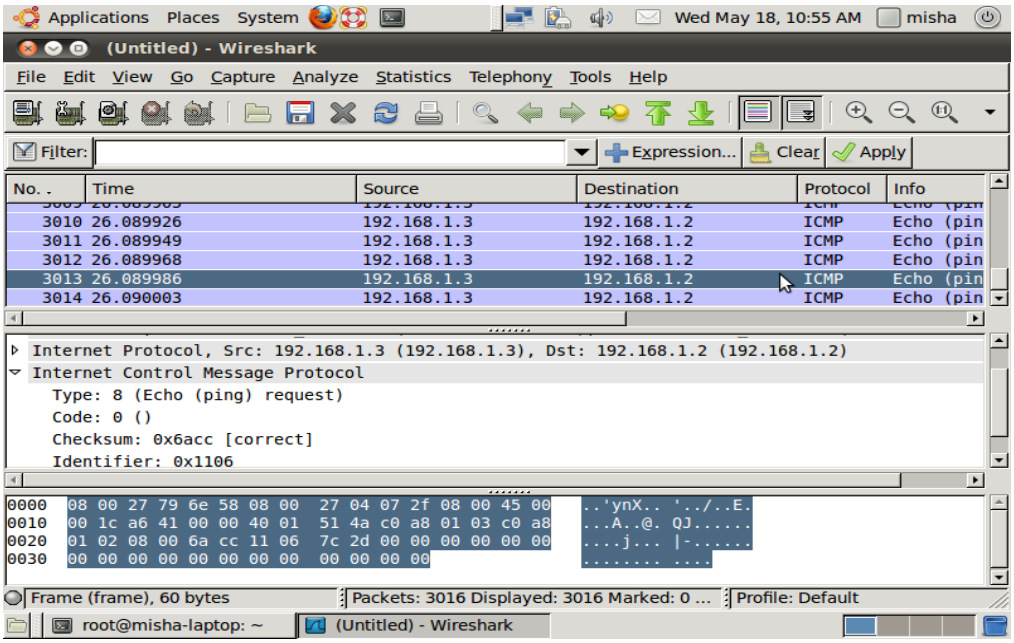

Figure 5.19: Wireshark Output after ICMP Flood Attack protection

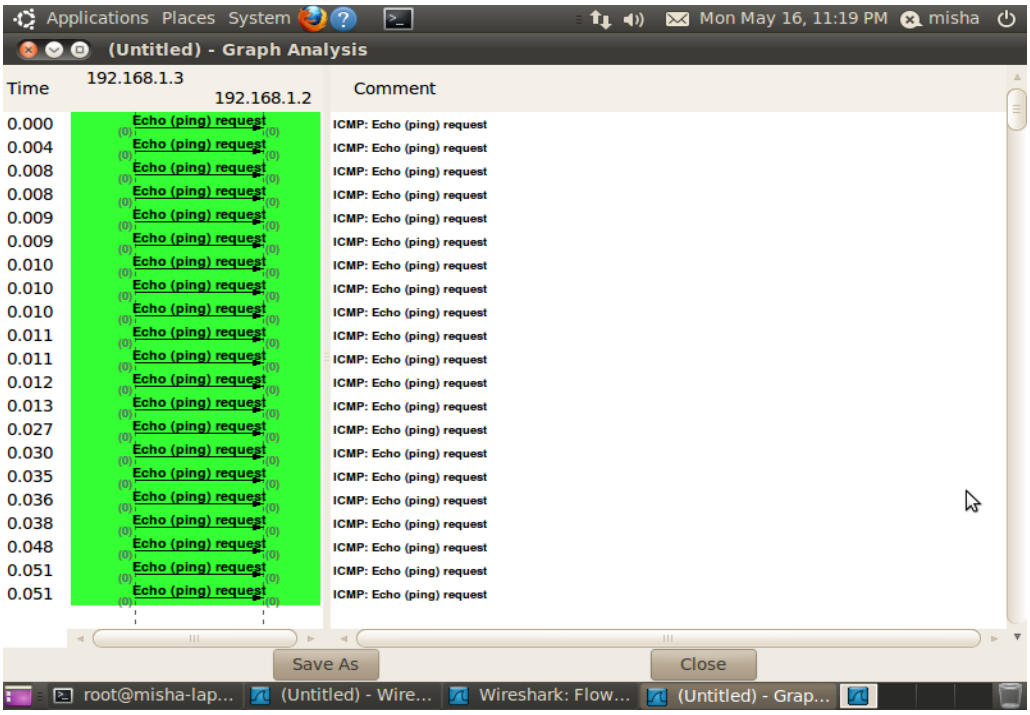

Figure 5.20: Flow graph after ICMP Flood Attack protection

As seen in Figure 5.19 and Figure 5.20 attacker is sending ICMP Echo Request packets continuously but victim's machine is not responding by sending ICMP Echo Reply packets as all the packets are being dropped by the firewall according to the IPtables rules seen in Figure 5.18.

5.3.4 Module 4-Trinoo Attack Protection

Proposed IPtables Scripts

IPTABLES –A INPUT -p udp --dport 31335 -m string --string "PONG" --algo bm -m comment --comment " msg:DDOS Trin00 Daemon to Master PONG message detected; classtype:attempted-recon;" –j DROP

IPTABLES –A INPUT -p udp --dport 31335 -m string --string "l44" --algo bm -m comment --comment " msg:DDOS Trin00 Daemon to Master message detected; classtype:attempted-dos; " –j DROP

IPTABLES –A INPUT -p tcp --dport 27665 -m string --string "gOrave" --algo bm -m comment --comment " msg:DDOS Trin00 Attacker to Master default password; classtype:attempted-dos; " –j DROP

IPTABLES –A INPUT -p tcp --dport 27665 -m string --string "killme" --algo bm -m comment --comment " msg:DDOS Trin00 Attacker to Master default mdie password; classtype:bad-unknown; " –j DROP

IPTABLES –A INPUT -p tcp --dport 27665 -m string --string "betaalmostdone" --algo bm -m comment --comment "sid:233; msg:DDOS Trin00 Attacker to Master default startup password; classtype:attempted-dos; " –j DROP

IPTABLES –A INPUT -p udp --dport 27444 -m string --string "l44adsl" --algo bm -m comment --comment "sid:237; msg:DDOS Trin00 Master to Daemon default password attempt; " –j DROP

5.3.5 Module 5-Mstream Attack Protection

Proposed IPtables Scripts

IPTABLES –A INPUT -p udp --dport 6838 -m string --string "newserver" --algo bm -m comment --comment " msg:DDOS mstream agent to handler; classtype:attempted-dos; " -j DROP

IPTABLES –A INPUT -p udp --dport 10498 -m string --string "stream/" --algo bm -m comment --comment " msg:DDOS mstream handler to agent; classtype:attempted-dos; " –j DROP

IPTABLES –A INPUT -p udp --dport 10498 -m string --string "ping" --algo bm -m comment --comment "msg:DDOS mstream handler ping to agent; classtype:attempted-dos;" –j DROP

IPTABLES -A INPUT -p udp --dport 10498 -m string --string "pong" --algo bm -m comment --comment "msg:DDOS mstream agent pong to handler; classtype:attempted-dos;" –j DROP

IPTABLES –A INPUT -p tcp --dport 12754 -m string --string ">" --algo bm -m comment --comment "msg:DDOS mstream client to handler; classtype:attempted-dos;" –j DROP

IPTABLES –A INPUT -p tcp --sport 12754 -m string --string ">" --algo bm -m comment --comment "msg:DDOS mstream handler to client; classtype:attempted-dos;" –j DROP

IPTABLES –A INPUT -p tcp --sport 15104 -m string --string ">" --algo bm -m comment --comment "msg:DDOS mstream handler to client; classtype:attempted-dos; " –j DROP

5.3.6 Module 6 Shaft Attack Protection

Proposed IPtables Scripts

IPTABLES –AINPUT -p tcp --sport 20432 -m string --hex-string "login|3A|" --algo bm -m comment --comment " msg:DDOS shaft client login to handler; classtype:attempted-dos; " –j DROP

IPTABLES -A INPUT -p udp --dport 18753 -m string --string "alive tijgu" --algo bm -m comment --comment " msg:DDOS shaft handler to agent; classtype:attempted-dos; "
–j DROP

IPTABLES –A INPUT -p udp --dport 20433 -m string --string "alive" --algo bm -m comment --comment " msg:DDOS shaft agent to handler; classtype:attempted-dos;
-j DROP

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

As seen, DoS/DDoS attacks are genuine threats that cause serious damage to many Internet users. The losses being suffered have escalated from being merely annoying to actually being debilitating and disastrous for some users. As long as DDoS attacks prove effective in achieving such aims, attackers are likely to continue using them. Until a reasonable defense against some kinds of DDoS attacks is found, their incidence, power, and seriousness increase because network bandwidth, processor speed, and number of available systems that can be attacked and compromised all continue to increase, as does the sophistication of attacker tools for compromising computers and using them to attack. Many different defense mechanisms are typically needed to mitigate DoS attacks, but it is not cost-effective to blindly choose a large set of defense mechanisms against DoS attacks.

In this work, capability of firewall is explored to defend against this attack. To determine whether the network traffic is legitimate or not, a firewall relies on a set of rules it contains that are predefined by a network or system administrator. These rules tell the firewall whether to consider as legitimate and what to do with the network traffic coming from a certain source, going to a certain destination, or having a certain protocol type.

Major concentration of the thesis has been on capturing the live traffic using the network protocol analyzer Wireshark and on the basis of analysis scripts using IPtables have been developed to allow/deny the network traffic depending upon the traffic rate of any IP address of the computer sending the packets and Iptables scripts have been developed against some DDoS attacks like Trinoo, Mstream etc.

6.2 Future Scope

The Netfilter/IPtables system is ideal for Linux system administrators, network administrators, and home users who want to configure firewalls according to their specific needs, save money on firewall solutions, and have total control over IP packet filtering hence even though this thesis provide ample defense against DoS attack, many areas of defense mechanism remain for further research.

The given work can be extended further by using advanced features of IPtables such as NAT, IP masquerading, packet redirect, IPtables has the ability to REDIRECT packets like IP Chains does, however it also has a generalized DNAT feature that allows arbitrary changing of the destination IP address and port number. Thus, it can actually disguise where packets of a given service go.

Another active area of research could be to develop policy scripts in Bro Language to detect DoS attack with the help of Bro IDS [55] or develop efficient Anti-DoS rules for Snort IDS.

References

[1] Che-Fn Yu and Virgil D. Gligor. A specification and verification method for preventing denial of service. IEEE Trans. Software Eng., 16(6):581–592, 1990**.**

[2] L.D. Stein, J.N. Stewart, The World Wide WebSecurity FAQ, version 3.1.2, February 4, 2002, Available: http://www.w3.org/Security/Faq.

[3] Aliens Attack Hogeschool Wittenborg Websites**,** Available: http://www.wittenborg-online.com/mod/forum/discuss.php?d=7167

[4] CERT Coordination Center, Overview of attack trends, Feb. 2002. [Online] Available: http://www.cert.org/archive/pdf/attack_trends.pdf.

[5] Cisco Systems, Inc., Characterizing and tracing packet floods using cisco routers, Feb.2003, Available: http://www.cisco.com/en/US/tech/tk59/technologies_tech_note09186a0080149ad6.shtml

[6] R. Stone, Centertrack: An IP overlay network for tracking DoS floods, in: Proceedings of the 9th USENIX Security Symposium, Denver, CO, 2000.

[7] R.K. Chang, Defending against flooding-based distributed denial-of-service attacks: A tutorial, IEEE Commun. Mag. 40(10) (2002), 42–51.

[8] K.J. Houle, G.M. Weaver, Trends in Denial of Service attack technology, CERT and CERT coordination center, Carnegie Mellon University, October 2001.

[9] P.J. Criscuolo, Distributed Denial of Service Trin00, Tribe Flood Network, Tribe Flood Network 2000, and Stacheldraht CIAC-2319, Department of Energy Computer Incident Advisory (CIAC), UCRL-ID-136939, Rev. 1, Lawrence Livermore National Laboratory, February 14, 2000.

[10]SYN Flood, Wikipedia Encyclopedia, Available: http://en.wikipedia.org/wiki/SYN_flood

[11] CERT® Advisory CA-1996-21, TCP SYN Flooding and IP Spoofing Attacks, September 19, 1996, Available: http://www.cert.org/advisories/CA-1996-21.html

[12] CERT® Advisory CA-1998-01, Smurf IP Denial-of-Service Attacks, January 5, 1998, Available: http://www.cert.org/advisories/CA-1998-01.html

[13] "Demystifying Denial-Of-Service attacks", December 2005, Available: http://aka-community.symantec.com/connect/es/articles/demystifying-denial-service-attacks-part-one.

[14] Juniper, Security Configuration Guide, Understanding Teardrop Attacks Available: http://www.juniper.net/techpubs/software/junos-es/junos-es93/junos-es-swconfig-security/understanding-teardrop-attacks.html#id-33015

[15] LAND , Wikipedia Encyclopedia Available: http://en.wikipedia.org/wiki/LAND

[16] D. Dittrich, G. Weaver, S. Dietrich, N. Long, The_mstream_ Distributed Denial of Service attack tool, May 2000, Available: http://staff.washington.edu/dittrich/misc.mstream.analysis.txt

[17] S. Dietrich, N. Long, D. Dittrich, Analyzing Distributed Denial of Service tools: the Shaft Case, in: Proceedings of the 14th Systems Administration Conference (LISA 2000), New Orleans, LA, USA, December 3–8, 2000, pp. 329–339.

[18] Kevin Mitnick and William Simon. The Art of Deception: Controlling the Human Element of Security. John Wiley and Sons, 2002.

[19] J. Mirkovic, G. Prier, and P. Reiher. Attacking DDoS at the Source. In Proceedings of ICNP 2002.

[20] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. 32(3):62–73, 2002.

[21] S. Ranjan, R. Swaminathan, M. Uysal, and E. Knightly. Ddos-resilient scheduling to counter application layer attacks under imperfect detection. In Proceedings of the IEEE Infocom, Barcelona, Spain, April 2006. IEEE.

[22] Tao Peng, C. Leckie, and K. Ramamohanarao. Protection from distributed denial of service attacks using history-based ip filtering. volume 1, pages 482–486 vol.1, 11-15 May 2003

[23] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur W. Berger. Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In 2nd Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA, May 2005

[24] D. Dittrich, The _Stacheldraht_ Distributed Denial of Service attack tool, University of Washington, December 1999,Available:staff.washington.edu/dittrich/misc/stacheldraht

[25] A. Harrison, "Cyberassaults hit Buy.com, eBay, CNN, and Amazon. com," February 2000, Available : http://www.computerworld.com/printthis/2000/0,4814,43010,00.html.

[26] P. Vixie (ISC), G. Sneeringer (UMD), and M. Schleifer (Cogent), November 2002, Available :http://www.isc.org/f-root-denial-of-service-21-oct-2002

[27] "Backdoor:W32/SdBot" , Available : http://www.f-secure.com/v-descs/sdbot.shtml

[28]"Backdoor:W32/Agobot", Available : http://www.f-secure.com/v-descs/agobot.shtml

[29]"Phatbot backdoor" , Available:
http://www.iss.net/security_center/reference/vuln/phatbot-backdoor.htm

[30]D. Dittrich, "Distributed Denial of Service (DDoS) Attacks/tools," Available: http://staff.washington.edu/dittrich/misc/ddos/.

[31] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", Available: http://www.ietf.org/rfc/rfc2827.txt

[32] W. Lee and K. Park, "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets," Proc. SIGCOMM, ACM Press, 2001, pp. 15–26.

[33]S. Savage et al., "Network Support for IP Traceback," IEEE/ACM Trans. Networking, vol. 9, no. 3, 2001, pp. 226–237

[34] D. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," Proc. IEEE INFOCOM, IEEE CS Press, 2001, pp. 878–886

[35] S. Bellovin, M. Leech, and T. Taylor, "ICMP Traceback Messages," Internet Draft, Internet Eng. Task Force, 2003.

[36] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, S. Shenker, Aggregate-based congestion control, ICSI Center for Internet Research (ICIR) AT&T Labs—Research

[37] V. A. Siris, F. Papagalou, Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks, In Proc. of IEEE Globecom 2004 (Security and Network Management Symposium), Dallas, USA, November 2004

[38] H. Wang, D. Zhang, K. G. Shin, Detecting SYN Flooding Attacks, In Proceedings of the IEEE Infocom, pages 000-001, New York, NY, June 2002, IEEE

[39] X. Luo, R. K. C. Chang, On a New Class of Pulsing Denial-of-Service Attacks and the Defense, In Internet Society (ISOC), NDSS05, 2005

[40] J. B. D. Cabrera, L. Lewis, X. Qin, W. Lee, R. K. Prasanth, B. Ravichandran, R. K. Mehra, Proactive Detection of Distributed Denial of Service Attacks using MIB Traffic Variables – A Feasibility Study, In Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management, Seattle, WA, May 2001.

[41] Netfilters, Available: http://www.netfilter.org/about.html

[42] T. Baba and S. Matsuda, "Tracing Network Attacks to Their Sources," IEEE Internet Computing, vol. 6, no. 3, 2002, pp. 20–26

[43] R. Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods," Proc. 9th Usenix Security Symp., Usenix Assoc., 2000, pp. 199–212.

[44] Aljafri , "IP Traceback: A new Denial-of-Service Detterant?" , IEEE Security and Privacy, vol. 1, no. 3 ,2003, pp. 24-31

[45] ICMP Traceback (itrace) , Available: www. ietf.org/html.charters/itrace-charter.html

[46] W. Lee and K. Park, "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack," *Proc. IEEE INFOCOM*, IEEE CS Press, 2001, pp. 338–347.

[47] Host Based IDS, Available: http://www.windowsecurity.com/articles/IDS-Part2-Classification-methodstechniques.html

[48] Wayne T Work "Intrusion Detection System What are They How do They Work"2003.

[49] Network- vs. Host-based Intrusion Detection "A Guide to Intrusion Detection Technology" 6600 Peachtree-Dunwoody Road 300 Embassy

[50] Zhang and Prashar, Cooperative Defence against DDoS Attacks, Journal of Research and Practice in Information Technology, Vol. 38, No. 1, February 2006

[51] David W Chadwick, "Network Firewall Technologies", IS Institute, University of Salford, Salford, M5 4WT, England, 2008.

[52] Internet Firewall Tutorial, A White Paper July 2002.

[53] Frederick M. Avolio and Marcus J. Ranum, "A Toolkit and Methods for Internet Firewalls", In Technical Summer Conference USENIX, Boston, Massachusetts, 2005.

[54]Ch14:Linux Firewall Using IPtables

Available:http://www.linuxhomenetworking.com/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables.

[55] Bro IDS, Available: http://www.bro-ids.org/

List of Paper Published/Communicated

1. Misha Singhal, Shalini Batra, "*Design and Development of Anti-DoS/DDoS Attacks Framework Using IPtables*", Journal of Global research in computer science (Communicated)