Fall 2004

An autonomous router-based solution to detect and defend low rate DDoS attacks

Karunakar Anantharam
New Jersey Institute of Technology

ABSTRACT

AN AUTONOMOUS ROUTER-BASED SOLUTION TO DETECT AND DEFEND LOW RATE DDoS ATTACKS

by
Karunakar Anantharam

Internet security was not a concern when the Internet was invented, but we cannot deny this fact anymore. Since all forms of businesses and communications are aligned to the Internet in one form or the other, the security of these assets (both infrastructure and content) is of prime importance. Some of the well known consequences of an attack include gaining access to a network, intellectual property thefts, and denial of service.

This thesis focuses on countering flood-type attacks that result in denial of service to end users. A new classification of this denial of service attacks, known as the low rate denial of service, will be the crux of our discussion. The average rate of this attack is so low that most routers or victims fail to detect the attack. Thus far, no solution can counter the low rate attacks without degrading the normal performance of the Transmission Control Protocol. This work proposes a router-based solution to detect and defend low as well as high rate distributed denial of service attacks (DDoS). A per flow approach coupled with the Deterministic Packet Marking scheme is used to detect and block attack flows autonomously. The solution provides a rapid detection and recovery procedure during an attack.

AN AUTONOMOUS ROUTER-BASED SOLUTION TO DETECT AND DEFEND LOW RATE DDoS ATTACKS

by
Karunakar Anantharam

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Department of Electrical and Computer Engineering

January 2005

APPROVAL PAGE

AN AUTONOMOUS ROUTER-BASED SOLUTION TO DETECT AND DEFEND LOW RATE DDoS ATTACKS

Karunakar Anantharam

Dr. Nirwan Ansari, Thesis Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Roberto-Rojas Cessa, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Swades De, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Karunakar Anantharam

Degree: Master of Science

Date: January 2005

Undergraduate and Graduate Education:

- Master of Science in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2005

- Bachelor of Science in Electrical and Electronics Engineering,
  University of Mysore, Mysore, India, 2001

Major: Computer Engineering

Presentations and Publications:

1. Z. Gao, N. Ansari, and K. Anantharam "A New Marking Scheme to Defend against DDoS Attacks," Proc. IEEE GLOBECOM'04, Dallas, TX, USA, Nov. 29 – Dec. 3, 2004.

2. A. B. Shevtekar, K. Anantharam and N.Ansari "Low Rate TCP Denial-of-Service Attack Detection at Edge Routers" accepted for publication in IEEE Communication Letters (2005).

To my family and friends

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Nirwan Ansari, who served as my research supervisor, providing valuable and countless resources, insight, and intuition. His constant support, encouragement, and reassurance gave me the motivation and the interest to focus on my research diligently. I would also like to thank Dr. Ansari for providing the financial support during this period. Special thanks are given to Dr. Roberto-Rojas Cessa and Dr. Swades De for actively participating in my committee.

I would like to thank Mr. Zhiqiang Gao and Mr. Amey B. Shevtekar for giving me an opportunity to work with them on their research projects and also actively involving me in their research discussions. These discussions helped me get a better insight of the field and also helped me explore new areas. I would also like to thank Mr. Pitipatana Sakirindr for helping me develop many diagrams used in this work. Many of my fellow graduate students in the Advanced Networking Laboratory also deserve recognition for their support.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF ACRONYMS

ARPA	Advanced Research Projects Agency
AIX	Ames Internet exchange point
DCL	DPM Control List
DPM	Deterministic Packet Marking
FMS	Flow Management Segment
IDS	Intrusion Detection System
IPS	Intrusion Protection System
NASA	National Aeronautics and Space Administration
NRN	NASA NREN connection at the AIX
NREN	NASA Research and Educational Network
RTO	Retransmission Timeout
RTT	Round Trip Time
SRTT	Smoothed Round Trip Time
TCP	Transmission Control Protocol
UCLA	University of California Los Angeles
UCSB	University of California Santa Barbara
UFMS	Un-established Flow Management Segment
UDP	User Datagram Protocol

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of this thesis is to present an autonomous solution to detect and defend low rate and high rate distributed denial of service attacks (DDoS) using a router based approach. This type of attack is very stealthy in nature: they can evade the users' detection systems easily while at the same time can deliver a lethal blow to the victim's infrastructure. It is also very important to note that since it has a very low rate, and the attack can be generated more easily as compared to the conventional DoS or DDoS attacking techniques. An effective analysis of the passing flows and monitoring their behavior at the edge routers is required to detect the malicious flows from the legitimate flows. Followed by detection, an effective filtering mechanism is used to defend against such attacks.

1.2 History of the Internet

The need for effective resource sharing and providing a remote access to military data and resources in times of crisis resulted in the birth of the Internet. It started as an ARPA project during the late 1950s and evolved rapidly to become one of the most viable means of communications today. The Internet originally known as the ARPANET was a collection of 4 nodes connecting the computers at universities in the south western United States of America (UCLA, Stanford Research Institute, UCSB, and the University of Utah). It went online for the first time in 1969 and was used mainly as a communications network in times of a contingency. Since the government initially funded the Internet, it

was originally limited to research, education and government uses only. Independent commercial networks began to emerge during the early part of the 1990s that made it possible to route traffic across the country from one commercial site to another without passing through the government funded Internet backbone like the NSFNet. Delphi was the first national commercial online service to offer Internet access to its subscribers.

The advent of graphical user interfaces and many more user friendly applications in the late 90s made the Internet a more viable form of communication. The ubiquitous nature of the Internet fueled the idea of eCommerce which caught the attention of many entrepreneurs. It was considered as a cost effective channel of advertising and marketing, and people saw many business opportunities with the growth of the Internet.

Evolution of the Internet began with the introduction of many protocols and applications in the late 1970s. With the advent of the TCP/IP network protocol suite which was much simplified, TCP/IP was widely accepted as the protocol standard for the Internet and many applications were developed based on this architecture. The following section provides a brief introduction to this Internet protocol architecture.

1.3 Protocol Architecture

Transmission Control Protocol/Internet Protocol is a suite of protocols that enable the networks/machines to be interconnected. TCP/IP forms the basic foundation for the Internet. If two nodes or systems have to communicate with each other, then the data or messages have to pass through several layers of the protocol suite. The ISO (International Standards Organization) proposed a highly modular, layered architecture for the

communication protocols over the Internet called the Open Systems Interconnection (OSI) model, published as OSI RM-ISO 7498. Figure 1.1 shows the OSI protocol architecture. Table 1.1 gives a brief description of the services provided by each layer.

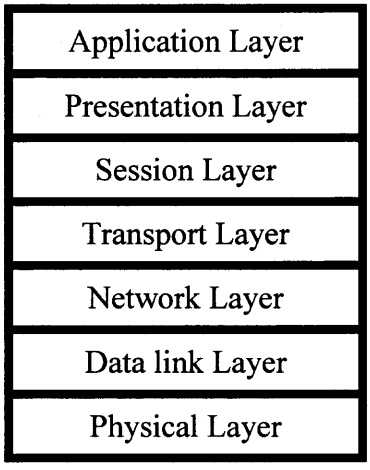


Figure 1.1 OSI Layer Architecture

The TCP/IP suite of protocols is a subset of the OSI model and contains 5 layers in its architecture. The services provided by the layers in the TCP/IP protocol suite are the same as the ones provided in the OSI model. A detailed description of the services provided by each layer of the protocol suite is described in Table 1.1. The layers in the TCP/IP suite are shown in Figure 1.2

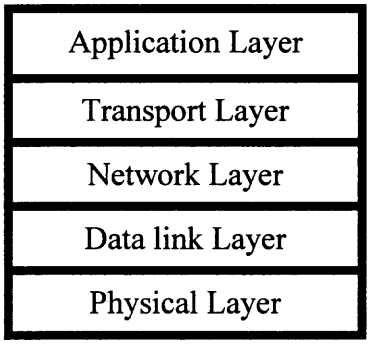


Figure 1.2 TCP/IP Layer Architecture

Table 1.1 OSI Layers and Services

Layer No.	Layer Name	Services provided by the layer
1	Application	• Provides application specific protocols for each application and each transport protocol system
2	Presentation	• Provides standard protocols for display purposes, data encryption and decryption
3	Session	• Establishes and clears applications, and thus helps minimizing the loss of data
4	Transport	• Multiplex and de-multiplex data from applications. • Makes and breaks connections for connection oriented communications • Controls data flow in both directions
5	Network	• Performs switching and routing operations on the network
6	Datalink	• LLC performs data formatting , error control and flow control operations • MAC layer is used for resolving data conflicts with other data on the LAN and also controls data transfer.
7	Physical	• Sends and receives data bits from the physical medium and handles the physical and electrical interfaces.

As the number of applications of the Internet increases, so does the number of users. A complex network which was once designed, maintained and used only by a few engineers and researchers is now used by a large community ranging from researchers, corporate users, to home users. As the dependence on this network increases, the security issues are becoming a critical concern. In the following chapters, we will discuss the protocol issues and the rising need for Internet security.

CHAPTER 2

TRANSPORT LAYER OVERVIEW

Chapter 1 presented a brief introduction and evolution of the Internet and its protocol standards. An overview of the TCP/IP protocol suite was also presented in the previous chapter. This chapter will discuss in more details on the protocols and the functions of the transport layer of the TCP/IP protocol suite which will be used in the following chapters.

2.1 Introduction

The transport layer of the TCP/IP protocol suite performs connection establishment and connection maintenance services. It is also used to control the flow of data in both directions between the source and the destination. This layer also multiplexes data from the application layer to lower layers and de-multiplexes the data from the network layer to the higher layers. The well known transport layer protocols are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP is a connection-oriented and reliable protocol while UDP is not connection oriented. Applications using TCP as a transport layer protocol, establish a formal connection before transferring data over the link. At the end of the data transfer, the application (client/server) performs a connection closure operation to indicate the end of data transfer. The TCP protocol is also responsible for verifying the correct delivery of data from the client to the server. Sometimes, data that are transmitted over the link are lost in the intermediate nodes. TCP adds support to detect errors or lost data, and to trigger retransmission until the data are correctly and completely received at the server. The protocol also implements many

algorithms to control data flow in both directions. On the contrary, there is no formal connection establishment and closure procedure while using UDP as the transport layer protocol. The protocol does not guarantee the delivery of packets and hence the reliability of this protocol is not guaranteed. However, one can observe that the processing overhead is significantly less using UDP than that using TCP. Hence, each protocol is adopted for applications based on their advantages and disadvantages. This chapter will provides the basic information required to understand the working of the TCP and UDP, and will also highlight how the schemes are being exploited by the attackers to create successful denial of service attacks.

2.2 Transmission Control Protocol

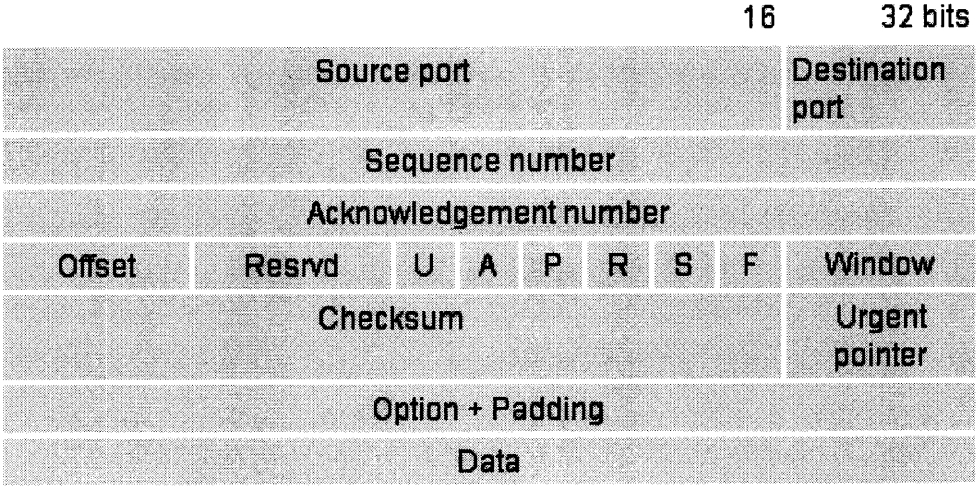


Figure 2.1 TCP header

Fig 2.1 shows the TCP header of a packet. The header contains the source and destination port number fields to identify the sending and receiving application. These two values uniquely identify each connection. Apart from this information, the header consists of the

sequence number and the acknowledgement number fields that maintain the sequence number of the current packet and the last acknowledgement number from the receiver. The TCP provisions reliability into the scheme by ensuring controlled flow of data between the source and destination. Whenever a TCP packet is sent out, a timer called the Retransmission Timeout (RTO) is maintained. The sender expects an acknowledgement for a packet sent out on the line before this timer expires. The receiver of the packet acknowledges the packet by sending an acknowledgement packet. The way in which a packet is acknowledged depends on the implementation. For example, the delayed acknowledgement scheme is one of them. The TCP header maintains a checksum field, which ensures the packet transmitted from the source reaches the destination without any errors. If the packet is corrupted, or if the checksums do not match, then the packet is dropped by the receiver. The receiver's acknowledgement packet indicates to the sender if the data was transmitted successfully. The sender retransmits the packet again on seeing a duplicate acknowledgement of the previous packet it sent. Finally, TCP maintains a buffer space at the source and destination end for each connection. The receiver "informs" the sender of its available buffer space to avoid buffer overflows. This will be discussed in greater details later in this Chapter 4.

One of the main advantages of using the TCP over the UDP as a transport layer protocol is its reliability. It means that the layer is responsible for ensuring the packet delivery to the destination. This is achieved by using various data flow control techniques at this (transport) layer. In this section we will be highlighting the congestion control mechanism provided by the TCP. The TCP congestion control mechanism uses various

algorithms to implement the data flow control. The slow start and congestion avoidance algorithms are used by a TCP sender to control the amount of data that can be transmitted to the receiver. The algorithms make use of two variables, congestion window size (cwnd) and the receiving window size (rwnd) to monitor the congestion states. The minimum of these two variables governs the rate of transmission. An *ssthresh* window is used to decide if the algorithm is in the slow start or in the congestion avoidance state. The initial value of the ssthresh is set according to the implementation and may vary in response to congestion. The slow start algorithm is used when the value of *cwnd* is less than the value of ssthresh, and congestion avoidance is used when the value of cwnd is greater than ssthresh. The TCP sender uses the fast retransmit algorithm to resend a packet once it detects 4 duplicate acknowledgement packets from the receiver.

The TCP's retransmission process uses a timeout mechanism which keeps track of the round trip time (RTT). An RTT is the time taken for a packet to reach the destination and a corresponding acknowledgement to return back to the source. If a packet is lost during transmission and if the number of duplicate acknowledgements received is less than 4, then the source waits for a time period equal to the retransmission timeout (RTO). Once the process times out, the congestion window is reduced, and the packet is retransmitted again. The congestion mechanism, RTT, and RTO will be explained in greater details in Chapter 4.

2.3 User Datagram Protocol

In the previous section, we have discussed the transmission control protocol as a reliable, stream-oriented protocol. In this section, we introduce another transport layer protocol called the User Datagram Protocol. As implied by the name, UDP is a datagram-oriented protocol: every output operation results in only one datagram. The UDP transmits the data as sent by the higher application layer. On the contrary, the TCP chops data sent by the application layer into the best size chunks before transferring them to the lower layers. The size of each packet depends on the path MTU. Figure 2.2 shows the UDP header.

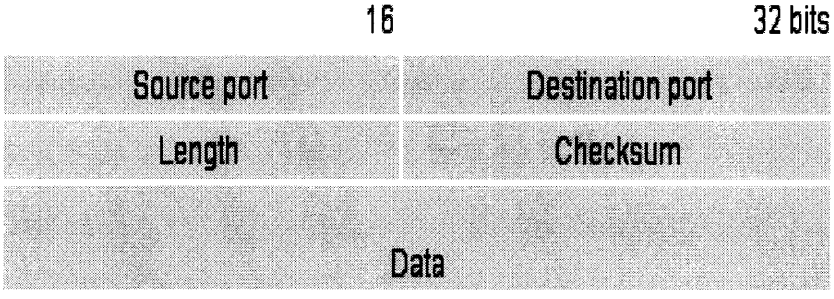


Figure 2.2 UDP header

The UDP header consists of a source port number and destination port number to identify the sending and receiving processes. The header also consists of a length field which indicates the length of the data and header. Finally, the header consists of a checksum field to check for the correctness of the received datagram. However, this field is optional and not used normally by UDP processes. The protocol is used normally for bulk data transfers and in applications where loss of packets is not of a big concern. The protocol is said to be very unreliable unlike TCP, and hence does not guarantee delivery, but only provides a best effort service. This protocol will not be dealt in details as it is not in the scope of this thesis.

2.4 Traffic Measurements

Internet traffic is increasing day by day and the diversity of the traffic increases with the increase in the variety of applications and the protocols designed and used for the applications. A major part of the Internet traffic is unicast traffic while a small part is multicast. Both forms of traffic can use TCP or UDP for the transport layer. Traffic measurements in [9] show the aggregated average composition of traffic by protocols at all sites except NRN as shown in Table 2.1. These fractions did not change appreciably during 10 months of monitoring. The NASA NREN connection at AIX is the only link heavily dominated by the UDP traffic during all six months of monitoring.

Metric	*TCP%*	*UDP%*	*Other%*
Bytes	(83±11)	(16±11)	(1±1)
Packets	(75±12)	(22±11)	(3±2)
Flows	(56±15)	(33±10)	(11±7)

Table 2.1 Composition of traffic by protocol for different metrics

Another trace-route experiment conducted by [15] to test the change in packet routes revealed that packets of a flow maintained a route consistently for 99% of the times. Only 1% of the time, a packet route change was observed. The results are deduced from 898 trace-route samples recovered from the tests. These tests were conducted at different time intervals and at different network load conditions. These test results form the basic assumptions for this work. In the subsequent chapters, the security aspect of the transport layer protocol and its exploits are studied in greater details.

CHAPTER 3

INTRODUCTION TO NETWORK SECURITY

Chapter 2 presented a brief overview of the transport layer protocols used today on the Internet. This chapter will shed light on the growing concern of network security. A detailed classification of the network attacks is documented to give readers a better insight into the field of network security. Finally, flood-based attacks are introduced with the emphasis on denial of service attacks and its variants, which will be the crux of our discussion in the following chapters. This chapter is also used as a quick reference to the attack scenarios referred in the subsequent chapters.

3.1 Background Information

In the early years of the Internet, a great deal of importance was given to the development of the Internet infrastructure, technology and the applications to support its growth. Today, along with the technological development to support the Internet Infrastructure, there is a new focus on the security of the Internet. This is largely attributed to the fact that there is a growing dependence on this Internet technology and the convergence of all technologies to use the Internet. A single point failure on this system can cause considerable damages to the technologies and the businesses depending on it. This has been experienced in many occasions previously in the form of network attacks. These attacks are created or executed by an attacker or a group of attackers working in tandem but sometimes located at different geographical locations. The motivation for an attack may be different that helps in classifying the attacks. However, most attacks cause some

form of damage or disruption of service to the end user operations. In the following sections, we discuss in details the classification of these attacks and their effects.

3.2 Classification of Network Attacks

There are several ways to classify the network attacks. Normally, these attacks are classified based on the way in which the system is attacked or on the goal of an attack. The attacks can be classified as local attacks or remote attacks. Local attacks normally aim at stopping services, by killing or crashing running processes or by reconfiguring the system. These attacks sometimes aim at exhausting resources by forking processes or by filling up the memory allotted for the file system. These attacks are executed by first breaking into a large network and by installing programs, or by modifying kernel root kits. Whenever the kernel executes a process and if the attacker modifies this process, then an attack is executed locally. Remote attacks also aim at stopping services or exhausting resources. In case of remote attacks, an attacker pumps in malformed packets or floods the victim network with spoofed packets, that sometimes causes stoppage in service. Sometimes they may exhaust the resources in the system, thereby leads to a denial of service. Fig 3.summarizes the classification of network attacks. The following section describes some of the networks in details to provide readers a better understanding of these attacks.

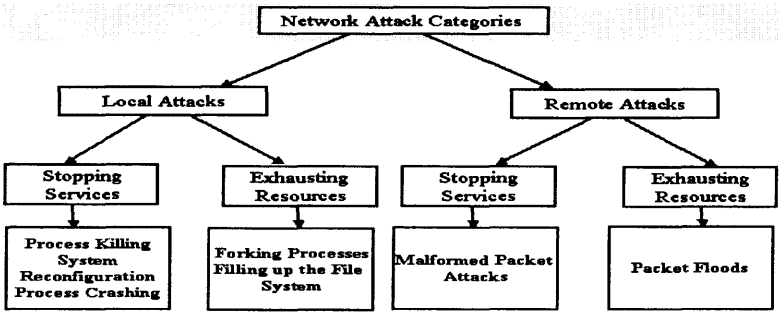


Fig 3.1 Classification of network attacks

3.3 Network Attacks Revisited

Some of the well known attacks are SYN flood attack, Smurf attack, land attack, man in the middle attack, DNS spoofing attack, ARP poisoning attack, session hijacking attack, buffer overflow attack, denial of service attack, and distributed denial of service attack. Each one of the above mentioned attacks can be categorized according to Figure 3.1. This section illustrates the above-mentioned attacks and their method of operation.

3.3.1 SYN Flood Attack

A small buffer space is assigned to store information about the session request whenever a new session is initiated between the TCP client and server in a network. The session-establishing packets include a one-bit SYN flag that identifies the request to establish the session. An attacker can send a number of connection requests very rapidly and then fails to respond to the reply sent from the server. Fig 3.2 illustrates a SYN Flood attack.

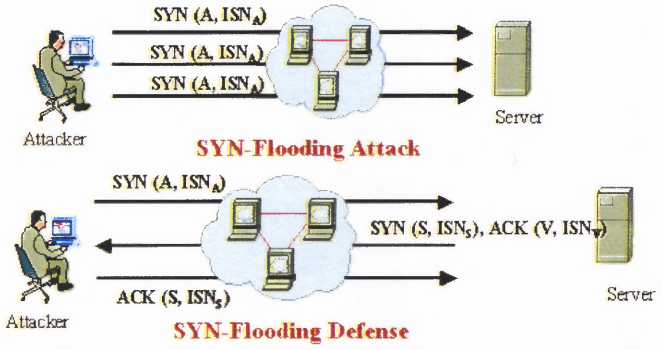


Figure 3.2 SYN flood attack

Although the packets in the buffer are dropped after a certain period of time without a reply, the consequence that there are too many of these SYN-set connection requests in the buffer at a time denies legitimate connection requests from getting established. The

damage resulted from this attack depends on the operating system so that correct settings or allowing the network administrator to adjust the size of the buffer and the timeout period might reduce the damage. However, the bandwidth used in flooding the SYN messages to the server is wasted, and might consume available bandwidth from legitimate users.

3.3.2 Smurf (Directed Broadcast) Attack

The attacker sends a ping echo request packet with a spoofed source address of the victim to the broadcast address of the victim's local network and/or of some network on the Internet, resulting in a lot of ping replies flooding back to the innocent, spoofed host. If the volume of the flooding packets is very high, then the spoofed host will no longer be able to send or receive packets as all bandwidth is consumed. The spoofed host will be blocked from connecting to the Internet. As the number of machines on the network that are allowed to directly broadcast increases, the number of ping response packets flooding the victim increases. These broadcasting machines are known as the Smurf amplifier. The Smurf attack aims to consume all of the victim's available bandwidth. If the attacker uses UDP packets instead of ICMP packets, it is known as the Fraggle attack. The attacker sends out an IP broadcast packet with the destination UDP port, such as the UDP echo service (port# 7). Upon receiving this packet, all machines on the network will echo with the UDP packets to the victim.

3.3.3 Session Hijacking Attack

In a session hijacking attack, the attacker monitors the communications between the victim and the server, and then tries to guess the sequence number (SN) of the next message. Fig 3.3 illustrates the session hijacking attack.

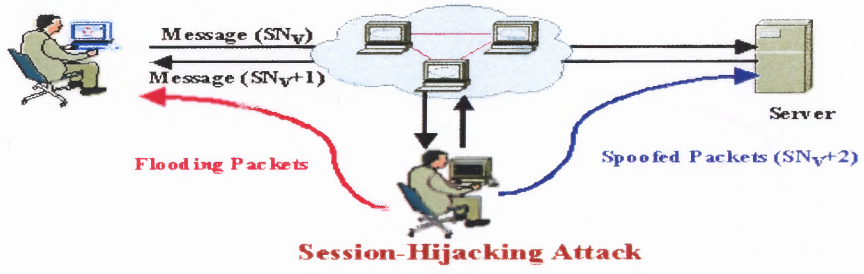


Figure 3.3 Session hijacking attack

The attacker then floods the victim with a high volume of useless packets so that the victim cannot send out or receive any packets from the server during the attacking period. Simultaneously, the attacker generates the spoofed packets to the server with guessed sequence number and spoofed source address of the victim's address. If the guessed sequence number is valid, the attacker can then communicates with the server. Since the server thinks the other end is a legitimate victim, the attacker can now harm the server or control the server as a "Zombie" in case of a DDoS attack.

3.3.4 Man-in-the-Middle Attack

In this type of attack, the attacker monitors the communications between the victim and the server by using several sniffing tools without modifying the packets. The attacker may try to modify the message without triggering the two ends of the message exchange. Fig 3.4 illustrates a man in the middle attack.

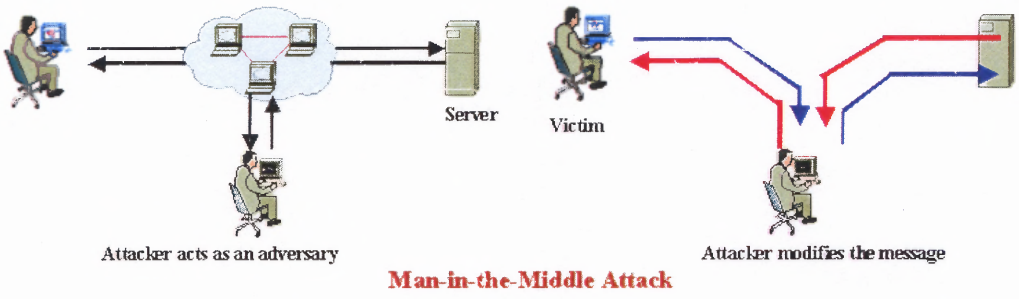


Figure 3.4 Man-in-the middle attack

The attacker is assumed to have some sort of encryption breaking mechanism applied to the communications. Unlike the session-hijacking attack, the victim still generates the message. In contrast, this attack seems similar to the session-hijacking attack in the sense that the attacker has full control over the messages because he can add or delete some portion or a whole of any packet.

3.3.5 DNS Spoofing (Poisoning) Attack

In case of a DNS spoofing attack, the attacker exploits the way the translation between the domain name and the corresponding IP address by replying the DNS request from the victim with a false IP address. The attacker may flood the DNS server in the local network in an attempt to block the DNS reply with the real IP address. The victim then generates the connection request with the false server, which is usually an attacker's server. Fig 3.5 illustrates a DNS spoofing attack.

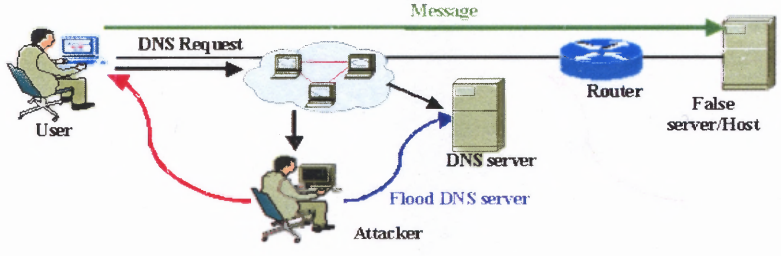


Figure 3.5 DNS spoofing attack

The attacker can learn the username, password, and other important information from the victim as the victim is deceived that he is accessing the true server. The victim may divulge to the attacker other authentication and security information to an attacker, that can be used to gain access to his/her account.

3.3.6 ARP Poisoning Attack

The ARP poisoning attack is similar to the man in the middle attack but the victim in this case is not overwhelmed by packet floods. Instead, the victim is deceived of the server's original location. When the user broadcasts an ARP request corresponding to the server/host IP address, an attacker tries to deceive the user by replying to the false physical address, which is usually the attacker's physical address. Later, if the user communicates with the server/host, it will use the false physical address. As a result, the packets are sent to the attacker, that may be forwarded to the server/host with the true physical address. The attacker is now able to successfully sniff the packets between the user and that server/host. Figure 3.6 illustrates this ARP poisoning attack.

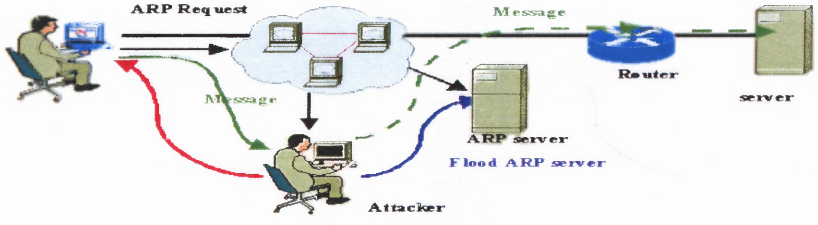


Figure 3.6 ARP poisoning attack

3.3.7 Buffer Overflow Attack

Stack-based buffers are assigned space to keep track of data received from a user. If an attacker sends extra data (larger than data buffers allowed by the programmer/administrator) along with the required information and if the programmer

forgets to write the code to check the size of data kept in the buffer, then the system is vulnerable to a buffer overflow attack. A savvy attacker who is aware of the target system weakness can exploit the system. The extra portion of the data, which is normally a virus or a malicious program, is stored in the buffer and overwrites the instruction pointer. This pointer points to the position that the next instruction that will be loaded and executed by CPU. As a result, the malicious program is executed to cause security flaws or create a backdoor for any future attack. As shown in Figure 3.7, when the function call is called, the return pointer and the frame pointer are saved along with the malicious code entered by the attacker. The size of code entered by the attacker is larger than what the validation program expects. The return pointer is overwritten and is loaded into an instruction pointer. The malicious code is then loaded by CPU corresponding to the new the value of the return pointer

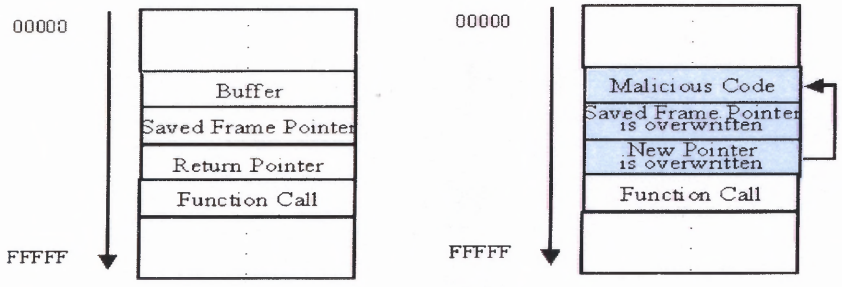


Figure 3.7 Buffer overflow attack

The malicious code can be a zombie in case of a denial of service attack that helps initiating distributed denial of service attacks on the command of a master machine located at a remote location.

3.3.9 Denial of Service (DoS) attacks

Denial of service attacks are a new class of attacks that are initiated by an attacker or a group of attackers to deny legitimate access to the end-users of a system. This class of

attacks exploits the weakness of the Internet Protocol to achieve its objectives. Some of the main objectives of this type of attacks are service disruption, resource consumption and bandwidth consumption. TCP targeted DoS attacks can be classified into two main categories: The connection oriented attacks and connectionless TCP attacks. Connection oriented attacks complete a three way hand shake in which it establishes a connection with the victim. By spawning multiple connections to the same host, the CPU utilization is increased to the point such that new requests for services from the end users are denied. In this case, the attacker IP address is mostly legitimate. In case of a connectionless TCP attack, the attacker does not complete the handshake initiated by the originator. In this case, the memory is utilized as the server/host has to wait till the request times out, thus causing a denial of service to the users.

3.3.8.1 Distributed Denial-of-Service (DDoS) Attack. DDoS is a DoS attack targeting a single or multiple server/hosts from various hosts. A DDoS attack is more lethal than a DoS attack, and often requires a combined effort to conduct a successful attack. Figure 3.8 illustrates a DDoS attack. Some of the well known DDoS attacks are Mstream, Trinoo, TN2k, Stacheldraht and Shaft. DDoS attack is a perfect example of the bandwidth attack.

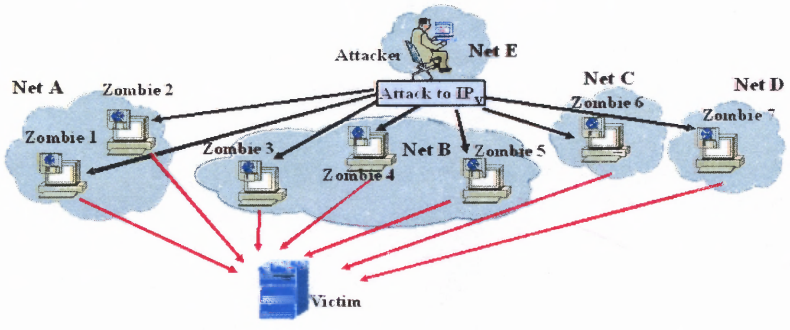


Figure 3.8 DDoS attack

3.3.8.2 Detection of Dos/DDoS Attacks. Degradation in the performance of a network or a device or an unusual consumption of bandwidth which results in a very high CPU utilization can be taken as the symptoms or indicators of a DoS/DDoS attack on a network. A premature conclusion can prove expensive as this effect is sometimes due to a network device mis-configuration. Unlike other attacks, the attacking host will be affected as much as the victim host. Hence, the attacker has to shield against this by an appropriate modification to the protocol stack on the attacking host(s). In the following chapter, we will discuss recent works of DoS attacks and their defense strategies. The remainder of the chapter will discuss the low rate denial of service attacks which will be the crux of this thesis.

3.3.8.3 Low Rate DoS/DDoS Attacks. In the previous sections, we talked about the DDoS attack classifications and their detection strategies. The detection of DDoS attacks is prompted by a major congestion or CPU utilization on the network because of the nature of its attack. This attack is mainly targeted at TCP services and hence the attack is termed as "TCP targeted low rate denial of service". This class of DoS attacks is unique in nature. The very name implies that the attack rate is very low, i.e., too low to signal a confirmed congestion or attack. It easily evades the detection systems designed to detect the DoS or DDoS attacks. Although the attack infrastructure and the attack initiation procedures remain the same as that of a DDoS attack, it differs in the attack duration or the resulting attack pattern which reduces the over all attack rate to a very low value, thus

easily evading the intrusion detection system. Figure 3.9 shows the attack patterns of a DoS/DDoS attack and low rate DoS attack patterns.

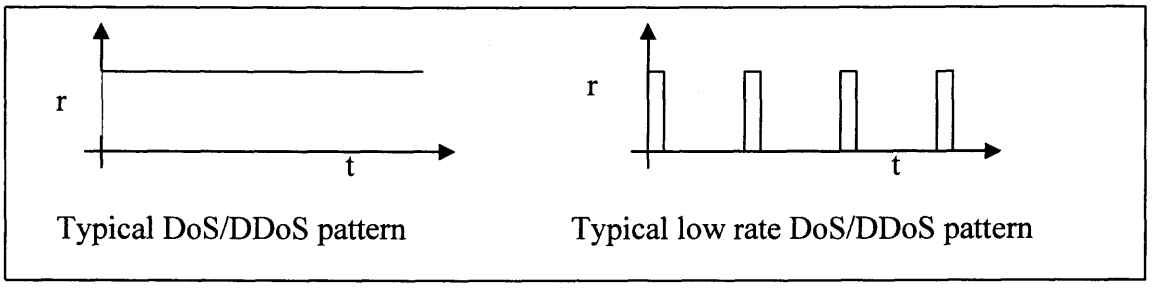


Figure 3.9 Typical DDoS attack and low rate DDoS attack patterns

To create a successful low rate attack, the attacker has to carefully monitor the existing flows and meticulously design the attack pattern with an attack duration just capable to timeout all the existing TCP flows on the path. More details on the design of this attack will be provided in Chapter 5.

Note that this kind of attack is attractive to attackers who can easily evade detection while carrying out a ghastly attack on the network resources. Since the attack initiation does not require huge infrastructure, it is easier to initiate and coordinate such attacks. As mentioned earlier, the detection of this kind of attack holds the key, and no remedy is present at this point of time to tackle this type of attack. With the advances of the personal computing technology, this attack is seen an imminent threat to the security of the Internet. In the following chapters, recent works on DDoS attacks and modeling of these attacks are discussed, and finally a router based solution to detect and defend against these attacks is proposed.

CHAPTER 4

LOW RATE DDoS ATTACKS

Chapter 3 presented a brief introduction to the low rate DDoS attacks. This chapter discusses in details the nature of the attack, its origin and modeling issues, and finally the existing solutions used to mitigate the attack.

4.1 Introduction

Chapter 3 described a variety of flooding attacks and all these attacks can be used to execute a successful DoS/DDoS attack. The concept of a flooding attack is to consume the network resources and bandwidth, thus disrupting the targeted services. On examining the attack traces, the attacker's identity is revealed in most cases. The limitation of these attacks is that in the attack process they raise alarms, and hence the IDS and the IPS are able to detect and defend the attacks using online or offline techniques. Although spoofing can mask the attacker's identity to a certain extent, an efficient traceback schemes like the ones mentioned in [3], [4] and [5] can detect the source of the attack, and the attack path can be identified easily. Low rate attacks differ from all these attacks, and hence all the detection systems built to detect and defend against high rate DDoS attacks do not apply. This category of attacks targets the TCP services on the Internet, which is a major chunk of the Internet traffic. Unlike the high rate attacks, the low rate counterpart does not use flooding as the means to attack the victims. Instead, these attacks exploit the inherent TCP fixed minimum retransmission timeout (minRTO) mechanism to attack the TCP flows. The attacker provokes a TCP flow to repeatedly enter the timeout state by sending a high rate, short periodic bursts.

The burst lengths are meticulously calculated by the attacker so as to timeout the targeted set of legitimate TCP flows. This periodic burst throttles the TCP throughput to a near zero level or causes substantial degradation of the cross flow throughput. The attack period is designed such that the average rate over the entire period of the pulse is very low, thus eluding the detection from most IDS and traffic monitoring systems. The following sections detail the attack model, queue analysis, and the aggregation issues.

4.2 RTT and RTO

The TCP congestion control mechanism effectively uses the two time scales; round trip time (RTT) and the retransmission timeout (RTO) to maintain a fair rate of transmission for each TCP flow. RTT is a smaller time scale variable operating in 10-100s of milli-seconds while the RTO is a slightly longer time scale variable. The time taken between transmission and the timeout is called the RTO.

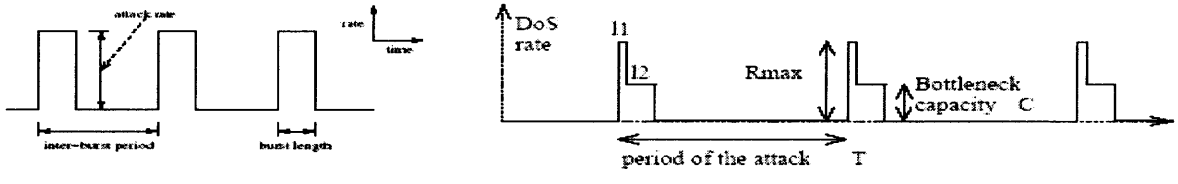
$$RTO = \max \{SRTT + 4^* RTTVAR, minRTO\} \quad (1)$$

Where SRTT is the smoother round trip time, RTTVAR is the round trip time variation defined in [1], and minRTO is the minimum RTO value that RTO can assume. The value of [SRTT + 4* RTTVAR] is normally less than one, and hence from Equation 1 the value of RTO is set to 1. Another reason for choosing RTO very carefully is to avoid spurious retransmissions and very high packet loss recovery times. A value of RTO lesser than 1 leads to spurious retransmissions while a value of RTO greater than 1 leads to a very high recovery time. The latter effect is attributed to the packet loss in the intermediate network. Allman and Paxson [10] showed that TCP achieves a near maximum throughput if the minRTO is chosen as 1 second. If a flow timeout occurs, then the RTO value is

doubled, and the congestion window for the flow enters the slow start phase as described in Chapter 2. In the TCP RENO scheme, a timeout is detected by no acknowledgement or by 3 duplicate acknowledgements. If the packet loss is sporadic, the DUPACKS may reach the sender and then the fast retransmit algorithm is applied. If there are not enough DUPACKS reaching the sender, then the connection has to timeout. Hence, these two variables are used to monitor the state of the flow, control the congestion window, and avoid a congestion collapse for the flow.

4.3 Attack Modeling

A low rate attack can be illustrated as a square wave that signifies a small attack burst and a long period of silence called the inter-burst period. The burst lengths can also be a summation of smaller bursts from different hosts located at different geographical locations. In [1], the authors also mentioned a possibility of a double rate or a multi-rate attack that can deliver the same effect. A bandwidth limitation encountered by the attackers may result in a multi-rate attack. For simplicity of modeling, the former case is considered. Simulation results show that detection of these attack packets produces a similar pattern as that of the rectangular pulse attack. Ideally, the attack pattern would be as shown in Figure 4.1a



(A) **(B)**

Figure 4.1 Typical Low rate DDoS attack pattern

Consider a low rate attack pattern with burst length *l* and period *T* with an attack rate of *R*. The attack rate is normally equal to or greater than the channel capacity. The period T [1] is defined as minRTO equals to 1 second. However, in reality, the attacker can vary the period of the attack. If minRTO of 1 second is maintained, the targeted flows encounter a periodic burst at the end of their timeout, and this will lead to successive timeouts and the RTO value will double every period. Eventually, there is a sufficient degradation in the throughput, and hence the flow decays to a near zero value over a short period of time. However, the period of the attack pulse can be varied such that the targeted flows encounter periodic outages. In this case, the RTO value hovers around a range of values and does not increment. The final effect on the targeted flows is either throughput stagnation or even degradation over a period of time. With the implementation of the Karn's back off algorithm, a repeated timeout puts the TCP connection in the back off phase where it can stay until the RTO count reaches 64. This will lead to a near zero service state to the flows. In [2], the authors discuss the possible scenarios of a TCP flow under a low rate attack. Consider an attack as shown in Figure 4.2. After the first attack pulse passes, all or most flows enter the timeout phase with each timeout based on their previous value of RTO. When the TCP flow exits from the timeout phase, the flow enters one of the three cases mentioned below. The left end of the circular arc shows the time when a flow is attacked while the right end of the arc shows the end of the timeout period.

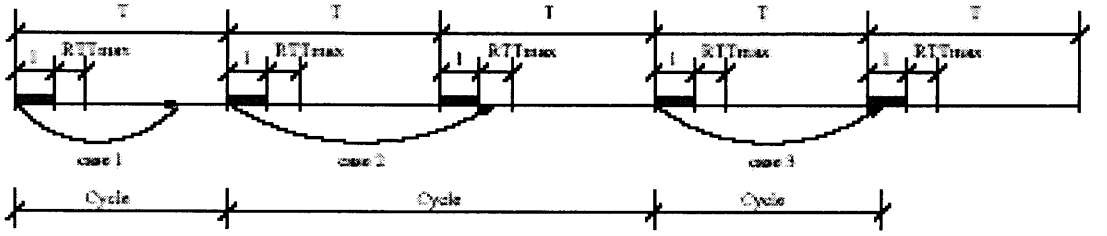


Figure 4.2 Three cases of the attack without a back-off scheme implementation

Case 1: In this case, the retransmission packets find the bottleneck free from attack packets and the queue is free. It is assumed here that under the no attack condition the queues are not congested or congestion is minimal. The flow uses the complete available bandwidth and quickly moves from the slow start to the congestion avoidance phase before the start of the next attack burst.

Case 2: In this case, when the TCP flow starts retransmitting packets, it encounters some residual attack packets in the bottleneck queues along the attack path. The retransmitted TCP traffic has to wait till the attack packets are cleared before reaching their destinations. The waiting time is estimated [2] as the RTT_{max} that is the maximum RTT of the flows at the bottleneck.

Case 3: This is the worst case where the TCP flow packets encounter the attack packets at the bottleneck queues, and hence are forced into another timeout. The future cycles of the flow face the same effect, thus leading to a complete denial of service over a short period of time.

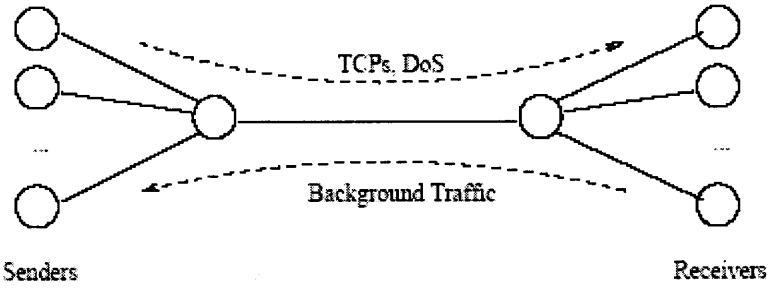


Figure 4.3 Schematic of a network bottleneck with TCP and DoS Flows

From the bottleneck shown in Figure 4.3 where the attack flows cross the TCP aggregate, if there are n flows and RTT_i is the RTT of the i^{th} flow and if the burst length of the attack is chosen such that l is greater than the maximum RTT of all the flow aggregates, then an attack burst can result in timing out all the TCP flows. Following this timeout, all the TCP flows are synchronized. Although the RTTs of the flows are different, the RTOs are synchronized to integer multiples of minRTO. A longer burst keeps the queues full for a longer time, thus increasing the effect of the attack. This method also increases the detection probability as the average rate increases. If during a low rate attack, the queues at the bottleneck burst at a constant rate R_d for a duration l. The time taken to fill the queues as described in [1] is equal to:

$$l_1 = \left(\frac{B - B_0}{R_d + R_{TCPi} - C} \right),$$

where: B = Size of the Queue
B_0 = Size of the queue on the onset of an attack
R_d = Attack Rate
C = Channel Capacity
R_{TCP} = Rate of the TCP Connection

4.4 Recent Works

The attack detection monitors in place today are all designed to detect high rate attacks, and hence the thresholds to confirm an attack are very high. As mentioned earlier, the solutions already in place to detect DoS/DDoS attacks cannot detect this low rate attack. RTO Randomization and router-based solution using queuing schemes are the only two solutions suggested so far in [1] and [2] to tackle the low rate attacks.

4.4.1 RTO Randomization: The solution proposed in [1] pointed out that randomizing the minRTO smoothens the TCP null frequencies. It is known that the TCP does not perform a doubling of the RTO value after n=6, implying that once the RTO value touches 64, the RTO value is no longer updated as 2^n, where n is the n^{th} consecutive timeout period. A random value between 2^n and 2^{n+1} is chosen for this RTO. It is said that the attacker no longer can synchronize the attack. Since this scheme does not involve any attack detection activity, the RTO randomization has to be performed even when there is no attack. This is a fundamental drawback, which affects the TCP performance, while it is still vulnerable to a different rate attack.

4.4.2 Router based Solution: The second approach mentioned in [1] and [2] is the router-based scheme which uses adaptive queue management schemes to tackle the attack. This approach is based on the preferential dropping of DoS flow packets such as Flow Random Early Detection (FRED), RED with preferential dropping (RED-PD), and Stabilized RED (SRED). In this approach, the attack packets are monitored and dropped with a probability, which is dependent on the sending rate. This approach fails when the

TCP traffic rate to the queues reduces to very low values. According to [1], the existing AQM schemes such as RED and RED-PD cannot detect burst lengths shorter than 300ms. If the attack is coordinated from different locations, then the burst length at a particular bottleneck goes completely undetected, thus leading to black-outs of TCP connections in that region or network. The solutions suggested could only mitigate an attack but cannot eliminate them, and also their limitations outweigh their advantages and ease of implementation.

Apart from these solutions, there is no solution, which can defend low rate attacks effectively. In the following chapter, a novel router–assisted approach to detect and defend the low rate attacks is proposed. The novelty lies in the autonomous approach used for flow classification and the object based approach. The solution performs attack detection effectively and proposes the use of a Traceback scheme mentioned in [3] to use to defend against the attack. The advantage of this solution is its scalability. Simulation results show that this approach can also detect low profile attacks and multi-rate attacks effectively. Details of the approach and the architecture design are elaborated in the next chapter.

CHAPTER 5

PROPOSED SOLUTION ARCHITECTURE

Chapter 4 analyzes the low rate attack in details and also discusses the current solutions available to counter or mitigate the attack. This chapter proposes a router-based solution architecture to detect and defend against the low rate attacks. The system architecture and the proposed lightweight data structure design form the crux of the discussion in this chapter.

5.1 Introduction

The schemes in [1] does not mention clearly about the detection mechanism for such an attack. In a high rate attack, the detection of the attack is obvious because of the nature in which the attack is executed. In the case of a low rate attack, detection of the attack holds the key to its defense. Simulation results show that network congestions and the gridlocks are an easy way to trigger a detection scheme in the case of high rate attacks. In the case of low rate attacks, the rate is very low and it can be easily mistaken for a bulk data transfer. Also, the solutions presented in the previous chapter can only mitigate the attack but cannot defend the attack completely. Hence, the limitations of the scheme in [1] and [2] far outweigh their advantages. Also, the solutions suggested earlier cause a throughput degradation in the absence of an attack. A critique on [1] suggested that a flow-based approach would be an appropriate solution to counter the attack effectively. In the following sections, a new autonomous router-based approach, which performs flow analysis, is suggested to detect the attack. The architecture also encompasses a packet-

marking scheme mentioned in [3] which, when used in conjunction with our detection scheme, can work effectively to provide a good defense against the attack.

5.2 New Layer Structure

The network stack structure through which a packet traverses varies along the network path as it encounters different network devices. The end hosts normally have a complete implementation of the stack as described in Chapter 1. The stack structure is based on the function of every device, and thus a switch which performs datalink layer operations has only the first 2 layers in its stack, and routers which perform network layer operations have the first 3 layers of the TCP/IP model in the stack structure. As a packet traverses from the source towards its destination, it can take any available route depending upon parameters like network congestion and route failures. Routers look at the header information to forward packets onto the next hop.

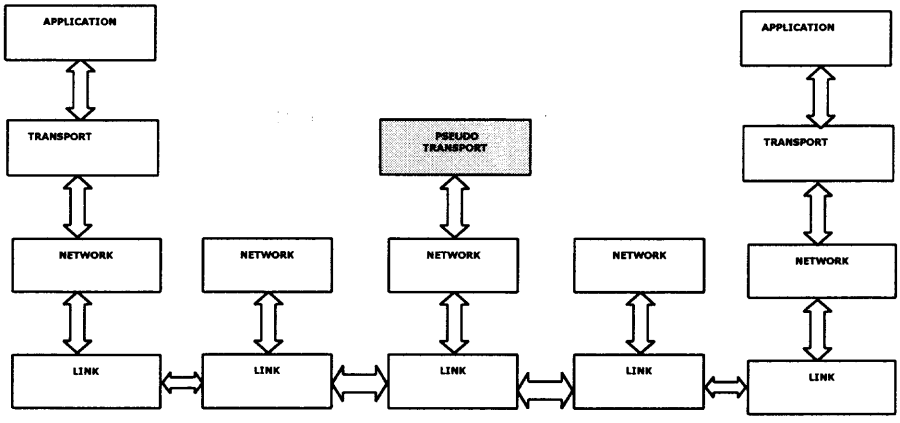


Figure 5.1 A new layer structure

In most cases, a system's vulnerability is exploited to launch an attack, and it is possible to identify such attacks at edge routers if we analyze each flow by monitoring parameters exploited in the attack. Routers are playing a significant role in provisioning QoS and

security; they will be required to perform higher layer functions such as application based traffic classification, and maintaining per flow information. In general, the proposed detection system will modify the Internet layer structure as shown in Figure 5.1, where edge routers will perform the transport layer function of maintaining some connection parameters.

5.3 Proposed Detection Architecture

Consider a scenario as shown in Figure 5.2 to explain the deployment of the proposed detection scheme, in which the edge routers connect a local area network to the Internet with typical client server connections. Each edge router acts as an entry and exit point for traffic originating from that local area network; essentially all incoming and outgoing traffic will pass through this point. The proposed detection system can be deployed at the edge routers of a local area network in which the server is present. For illustrative purposes, we assume that all clients are outside the local area network in which the server is present, so that the detection system can monitor all flows connecting to the server.

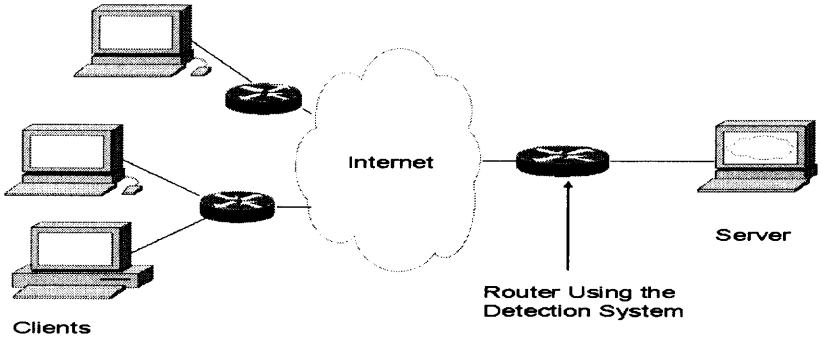


Figure 5.2 Proposed detection system deployed at an edge router.

It is assumed that most of the packets of a particular TCP connection flow through a particular path from the start of the connection till the end, and the deviation in its path is

very minimal. This assumption is based on the study conducted by [11] which showed that 99% of the packets for a TCP connection maintained the same path under several trials spread over several months and under different network conditions. The edge router of the network is chosen to implement our solution as all the input traffic to the network aggregates at this point. Figure 5.3 shows the basic architectural layout of the detection system. The system encompasses a flow classifier, an object module, and finally a filter to block the suspected attack flows. The following sections describe the design and operations of each subsystem in greater detail.

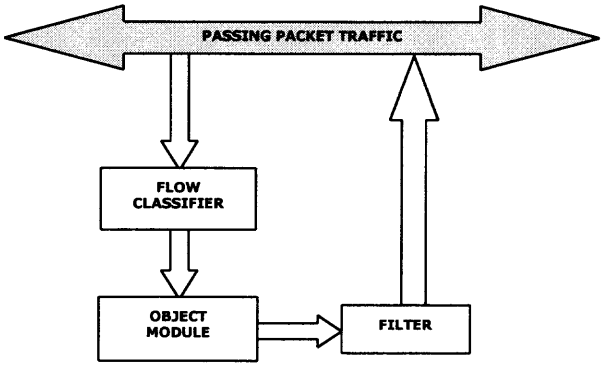


Figure 5.3 Architecture of the detection system

5.3.1 Flow Classifier

A flow is defined as an end-to-end connection between a client and a server. Although flows can be classified as TCP and UDP flows, we consider the TCP flows here for the sake of discussion. Each flow is identified uniquely by a flow id. Some of the parameters required to generate a flow id are the source and destination IP addresses, and source and destination port numbers. In order to tackle the problem of spoofing, we require some more parameters to properly assign a unique flow id for every source. Details of generation of a flow id are discussed later in this chapter. The flow classifier module is

the first block in the detection system to receive the passing packet traffic. The flow classifier, which encompasses some of the functionalities of a transport layer, acts as a pseudo transport layer and captures the flow information from the packet and forwards it back to the link. There is no additional delay apart from the packet lookup delay. The flow information thus gathered is used to identify the flow id for the packet, and this information is the passed to the object module. The flow classifier also passes some additional data to the object module.

5.3.2 Object Module

The object module takes the input from the flow classifier to further classify the flows. An object is created for every flow id and the transactions of the flow are stored as attributes of the object or the history of the flow. The packet manager, with the available flow information, decides the segment of the memory to be accessed to update the data structure. The flow objects are stored in the proposed lightweight data structure, which resides in the router main memory. This area of the memory is called the flow management segment (FMS). This space is further subdivided into the established and the un-established flow management space. The semantics of the flow information is used to decide the location of the memory to be accessed. Hence, all the established flows structure is located in one area while the un-established and the attack flow data reside in a different area. Grading of flows helps in releasing the area of memory reserved by the established flows. A flow's trustworthiness is determined by observing the TCP semantics stored in these objects. If a flow is found trust worthy, then there is no need to

monitor it any longer and the memory allocated to the flow is released with minimal monitoring.

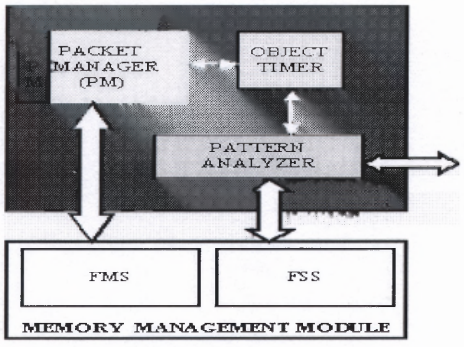


Figure 5.4 Proposed Object Module

The object module shown in Figure 5.4 also houses the pattern analyzer. The pattern analyzer runs on its own without interfering with data stored in the flow scan segment (FSS). This segment is a volatile space as it gets periodic dump of the flow objects to be analyzed. The analyzer performs flow analysis and flow grading activities to filter the suspected flows from the normal flows. Keeping in mind the effect of false positives, the analyzer performs multiple pass detection on the suspected flows before declaring on the attack flows. The information of the flows that are detected as attack flows is passed to the filter module for further monitoring and quarantining activities.

5.3.3 Filtering Module

This is the final module, which consists of a quarantining subsystem and a deterministic packet mark control list (DCL). The DCL structure is a one to many mapping as shown in Figure 5.5.

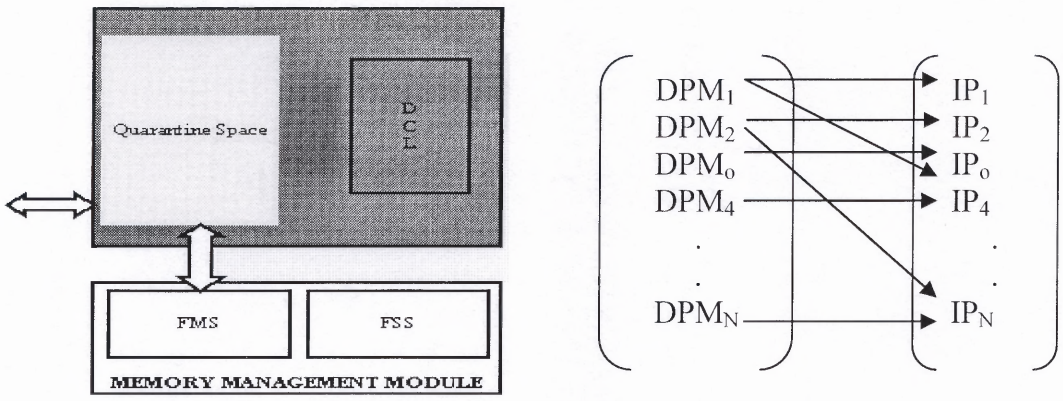


Figure 5.5 Proposed Filtering Module

The list is a mapping of DPM marks to destination IP addresses. Once the attack flow ids are identified, any ingress packet with a DPM mark listed in the DCL intended to the corresponding destination is quarantined. The packet flow id is checked for its grade as a well-established or un-established flow to decide if the packet has to be released for routing or if the packet should be dropped. If the packet is an un-established flow, then the packet is quarantined and dropped. If the packet belongs to an established connection, the packet is released. Thus, even under an attack, the established flows do not suffer throughput degradation. If the packet is an un-established flow, then it is quarantined till the attack is moderated.

5.3.4 Data Structure

In this section, the lightweight data structure designed to achieve the above objectives is explained. Figure 5.6 shows the schematic of the data structure. It is designed as a 2-Dimensional doubly linked list. The colored nodes in the figure represent the flow objects. The nodes in a column are the attributes of the object that stores the history of the flow. The data structure is very lightweight and holds all the necessary flow information for monitoring and detecting an attack. Details of the definition of the data structure and

its implementation in C++ are available in Appendix. The data structure is stored in the flow management segment, and refreshed from time to time with the recent activities of the flows. If a flow terminates, then the flow id is deleted from the list and is analogous to deletion of a node in a list. The new attributes are appended to the flow object at the beginning of the list, thus reducing the access time for scanning the flows. Hence, the list traversal time is very small and hence reduces the complexity of scanning operations.

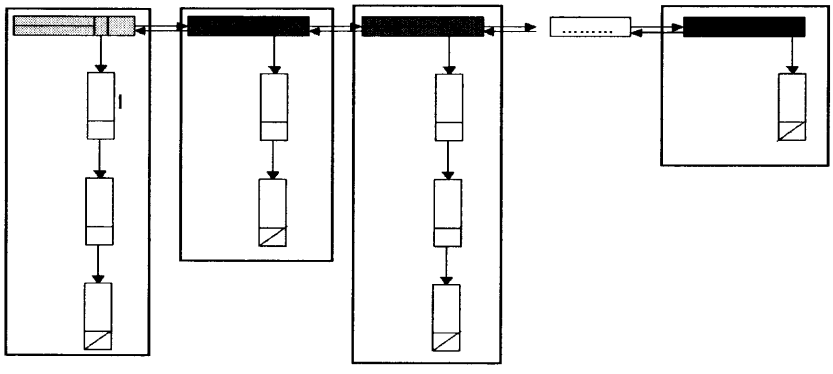


Figure 5.6 Proposed data structure model

The data structure shows a number of rectangular blocks. Each block represents a flow object with the colored node being the head node, and has all the flow connection information. The vertical nodes in the block represent the attributes of the flow. The number of attributes depend on the number packets each flow encounters in a particular interval of time. All the un-established flows are temporarily considered under one generic flow before moving them to their actual object upon flow establishment confirmation. The choice of such a structure over a tree model is debatable. Consider n flows and m attributes recorded per flow, then the time complexity to perform an operation on this list structure would be of the order O(n) * O(m).The complexity of the tree model, however, would be of the order of O(log n) * O(log m). This complexity is

though far lower as compared to the list structure model in the time perspective; it consumes more memory for implementation. The space complexity of the tree model would be $2^n * 2^m = 2^{n+m}$ bytes while that of the list structure is n*m bytes. Since the detection process runs as a separate thread, time complexity is not a concern. The only overhead considered at this point of time is the packet loop back overhead, which is assumed to be small and negligible. Space overhead still remains a concern for line card memory of border routers. It is assumed that flow aggregation happens at the border routers or gateways, and hence the solution needs to accommodate several thousand flows. In this work, we consider only TCP flows, as these flows are the target for the attack. The data structure elements are carefully chosen, and space is allocated optimally. Considering all flows are monitored all the time, i.e., the worst-case scenario, the entire solution would consume at most 3% of the line card memory. Since the TCP flows are graded, trusted flows are not monitored after a certain point of time. This releases a lot of memory for attack detection operations and the consumption goes well below 1%. The estimates of memory consumption mentioned here are based on the assumption that the model can support up to 10000 TCP flows. With the increase in number of flows per line card, the queuing complexities arise that is beyond the scope of this discussion.

5.4 Low Rate DDoS Attack Detection

The attack pattern as described in Chapter 4 is a periodic stream with a burst length l greater than or equal to the maximum RTT of all the TCP flows in the attack path, and the inter-burst period T is equal to the fixed minimum RTO. An attack flow satisfying

these two conditions can cause denial of service to the TCP flows. The data structure explained in the previous section maintains the arrival times of these packets at the router. This pattern analyzer computes the time difference of the consecutive packets of each flow. The average high and average low values are recorded. The scheme assumes that packet path deviation is very rare and negligible, and this assumption is based on an independent study conducted in [11] which shows the path deviations over several months of observation were found negligible. Simulation results showed that these average values repeat periodically only for the attack flows and the averages vary randomly for legitimate flows. From these values, an approximate burst length and the inter-burst period can be estimated and is compared to the relative RTTs (RTT from the perspective of the router), and the estimated inter-burst period is compared to the fixed minimum RTO. A flow that meets the two criteria of inter-burst period T and the burst length l as mentioned in [1] is marked malicious. This approach is autonomous in its operation and does not produce any overhead on the throughput of the legitimate flows where there is no attack. The solution architecture also provisions a better service for legitimate flows in times of attack. From simulations, we see that the scheme is very responsive even for burst lengths lesser than 1 msec. Thus, this scheme can tackle low profile and multi-rate attacks. The next chapter provides the simulation results of the analysis and detection of the attack for different attack burst rates.

CHAPTER 6

SIMULATION RESULTS

Internet simulations are very useful to understand the dynamics, to illustrate a concept and, to explore unexpected behaviors of test networks. Simulating the Internet and analysis of the simulation results is a very challenging task while analysis provides the possibility of exploring an analytical or mathematical model over which one has a larger control; many key issues are neglected to simplify the analysis, and hence the results of such an analysis may not be very accurate and useful in real conditions. On the contrary, simulations not only confirm the theoretical deductions but also allow a researcher to explore many complicated scenarios that are very difficult to analyze theoretically. Moreover, simulations aid the research in developing intuition and also give hands on experience and deeper understanding of the concepts. In this thesis work, the ns2 simulator was extensively used and modified to analyze, model, test, and implement the proposed solution.

6.1 Introduction to NS2

NS is a discrete event simulator used widely in the networking research community. The simulator provides extensive support for simulating TCP, routing, multicast and unicast protocols over wired and wireless networks. The simulator project originated as a real network simulator in the 80s, and is now developed and maintained by DARPA, Xerox PARC, UCB and USC/ISI. This project is now popularly known as the VINT project. In the following sections, the simulation results of the network under attack and no attack

conditions are presented along with the flow control parameters. Results of the proposed detection system are also presented at the end of this chapter. A detailed explanation of the detection system implementation is presented in the Appendix.

6.2 Simulations Results

Figure 6.1 shows the network topologies considered for the simulations. Different networks were used to investigate different issues of the attack and to test the proposed solution.

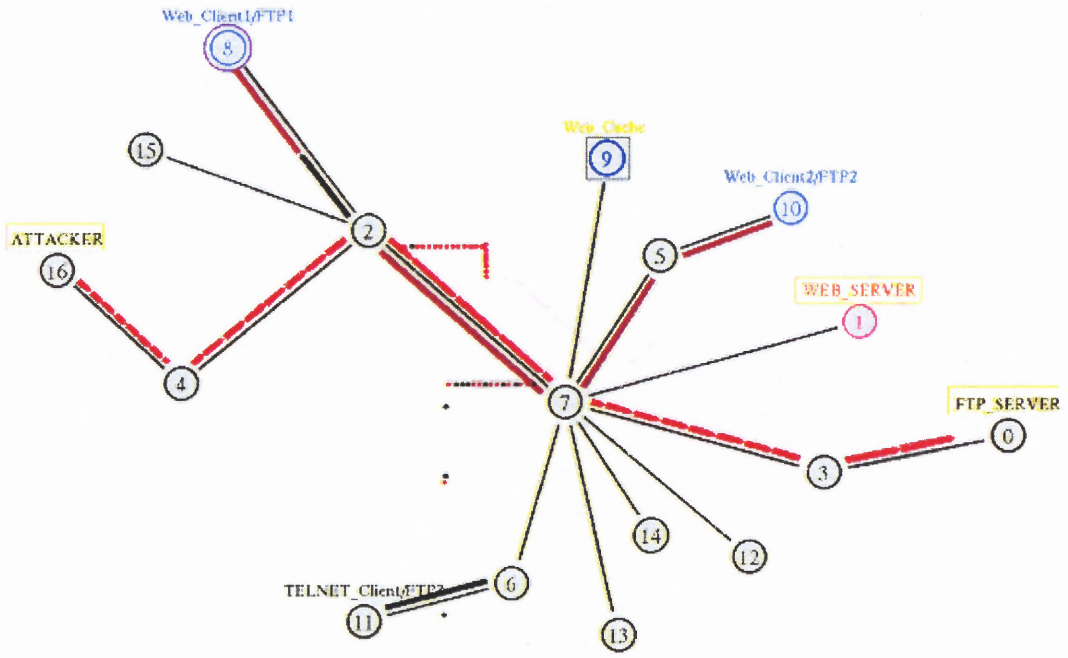


Figure 6.1 Simulation Network

In the network shown in Figure 6.1, only the bottleneck is shown. Nodes 2 and 7 are the nodes connecting the bottleneck link. The attacker traffic is assumed to aggregate at node 16 and is intended for node 0 which happens to be a FTP server. Nodes 11 and 8 show the FTP client flow aggregation. The figure depicts a scenario where the bottleneck is

now congested by the low rate attack burst and all the incoming traffic is dropped. The RTT trace of 9 TCP flows used for the simulation is shown in Figures 6.2. The test network consists of end nodes, which represent not just a node but traffic aggregation from a set of smaller nodes. The traffic agents consists of TCP, UDP and web traffic. The applications used for the simulation were CBR flows, FTP flows, and Telnet flows. Attack patterns were simulated using timed UDP and CBR flows. In order to create a successful attack, knowledge of the existing flows and the nature (long lived and short lived flows) are needed, upon which the attack patterns are designed. Thus the normal network operations were simulated to observe the network parameters. Some of them are the RTT trace of the flows over a period of time and the congestion window. The traces of these values are plotted in Figure 6.2

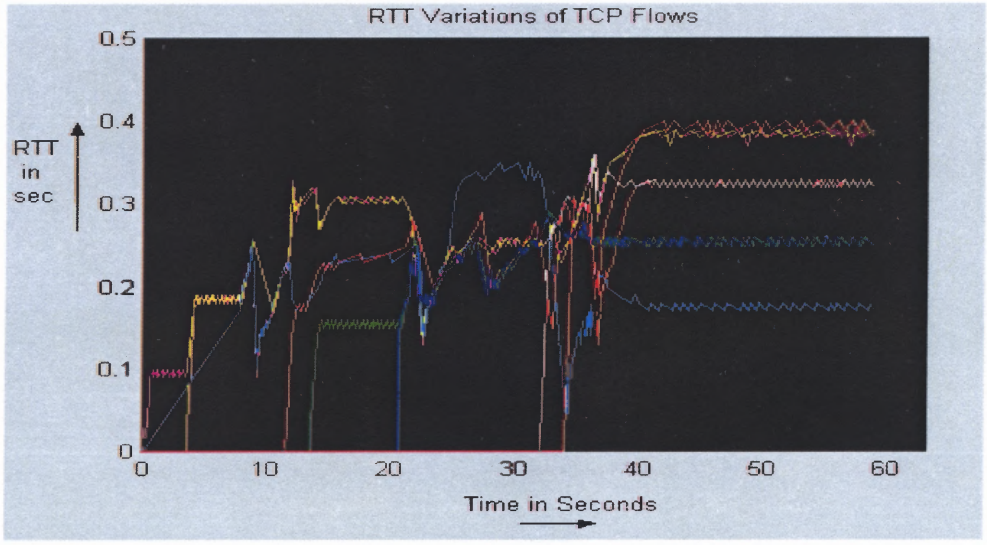


Figure 6.2 RTT Trace

Figure 6.2 shows the RTT trace of 9 TCP flows considered for analysis. The RTT patterns of flows 1, 2 and 10 settle at 400ms at the end of 1 minute. Flows 5 & 6 settle at 250 ms; 3 & 7 at 180 ms; 4 & 9 at 330ms. Due to difficulties mentioned in [12] and [13],

we adopt the model described in [1]. Depending on the flow or the set of flows to be targeted, a suitable attack pattern is designed using the modeling techniques from [1].

6.3 Implementation Results

The scheme proposed in Chapter 5 uses the data from the packet objects stored for attack pattern analysis. Figure 6.3 shows the time difference pattern of the TCP flows from 1 through 9 under a no attack condition. A fraction of this figure is seen by the detection system at regular intervals, and it can be observed that the periodic behavior is absent and none of the flows satisfy the attack conditions.

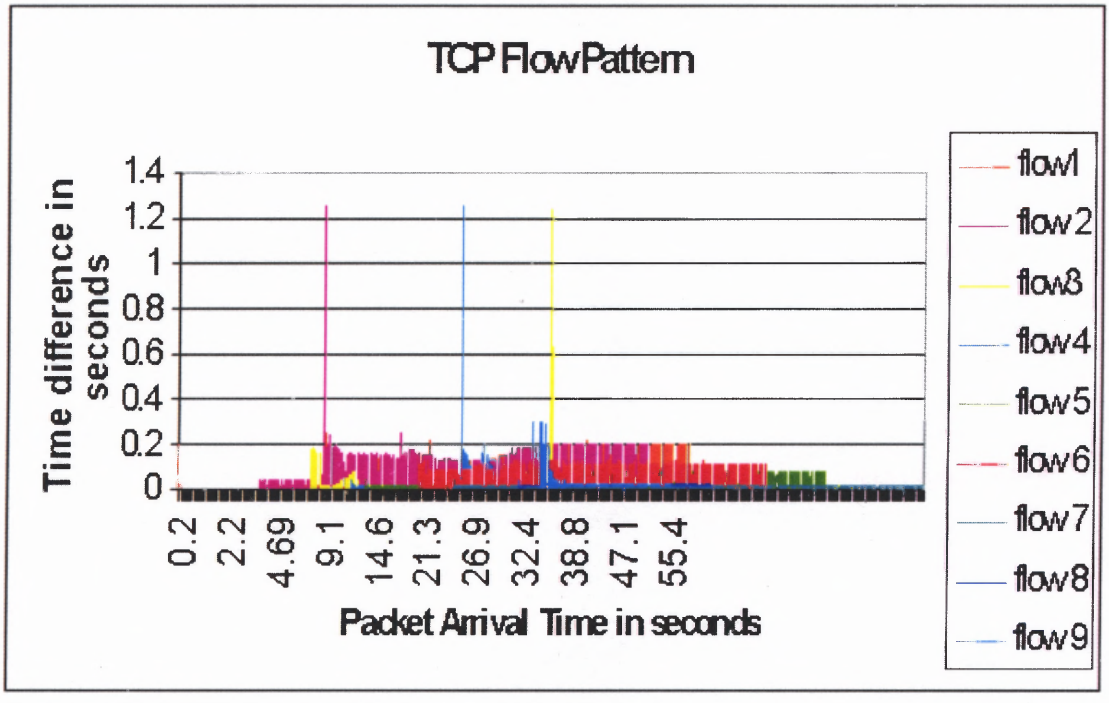


Figure 6.3 TCP flow pattern

Similarly, Figure 6.4 shows the time difference pattern of the web traffic observed over a small interval of time. The web traffic consists of mostly HTTP requests and replies and the size of the packet varies randomly. Since the web requests and replies are random in

nature, the pattern's distribution does not satisfy any attack conditions and the flow is not detected as an attack.

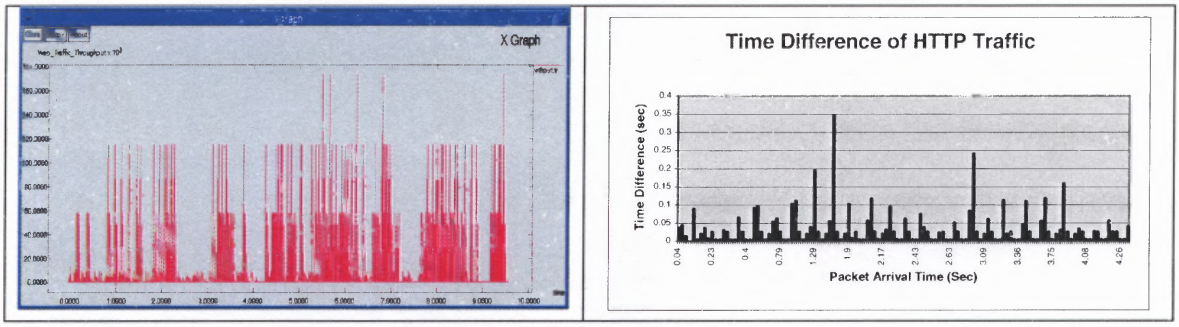


Figure 6.4 Web Traffic Pattern

The FTP flow, which represents the category of bulk data traffic, is observed, and we plot the time difference of the ftp flow in Figure 6.5. The distribution is uniform but the inter-burst period is too small when compared to the fixed minRTO time period, and is thus not considered as an attack. Moreover, the time difference (as shown in Figure 6.3) also varies when the flow is observed for a larger interval of time, thus clearing the flows from the false positive list.

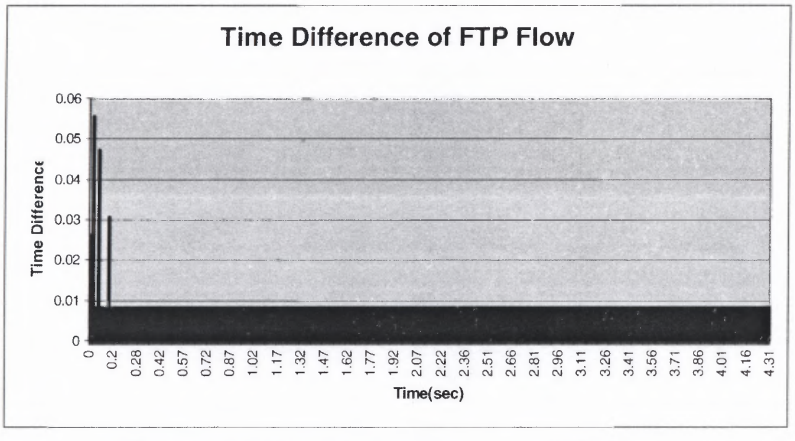
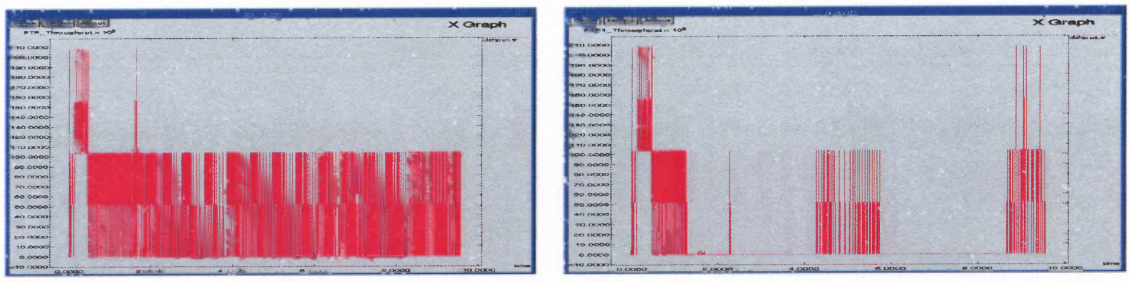


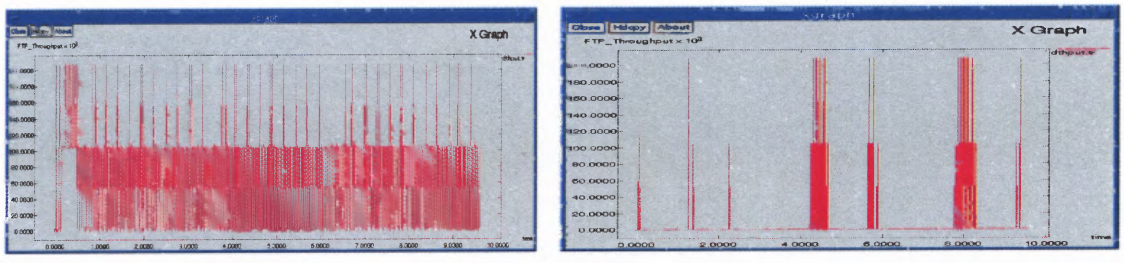
Figure 6.5 Time Difference of an FTP Flow

Figure 6.6a depicts another FTP simulation under attack and no attack scenarios. The right hand side of the figure shows the FTP throughput pattern under attack. From the

figure, we can see a growth in the RTO value after every timeout. The flow decays to near zero throughput in a very short period of time. In Figure 6.6b, another FTP simulation result is depicted. In this case, although there are repetitive timeouts, it is seen that the RTO window does not grow exponentially to throttle the flow. These two simulations describe the effect of RTO value on a flow as described in Chapter 4.



(a)



(b)

Figure 6.6 Bulk Traffic Patterns affected by low rate DDoS Attack

6.3.1 Attack detection results

In the previous chapters, the limitations and existing mechanisms were described and a novel approach was proposed. In this section, the results obtained from the proposed detection system are presented. Figure 6.7 shows the attack pattern as detected by the detection system with no data traffic passing through the edge router. The scenario is hypothetical and is used for simulation purposes only. Figure 6.7b shows an attack pulse

of 250 ms width; 6.7c an attack pulse of width 350 ms, and finally Figure 6.7d shows an attack pulse of width 500 ms.

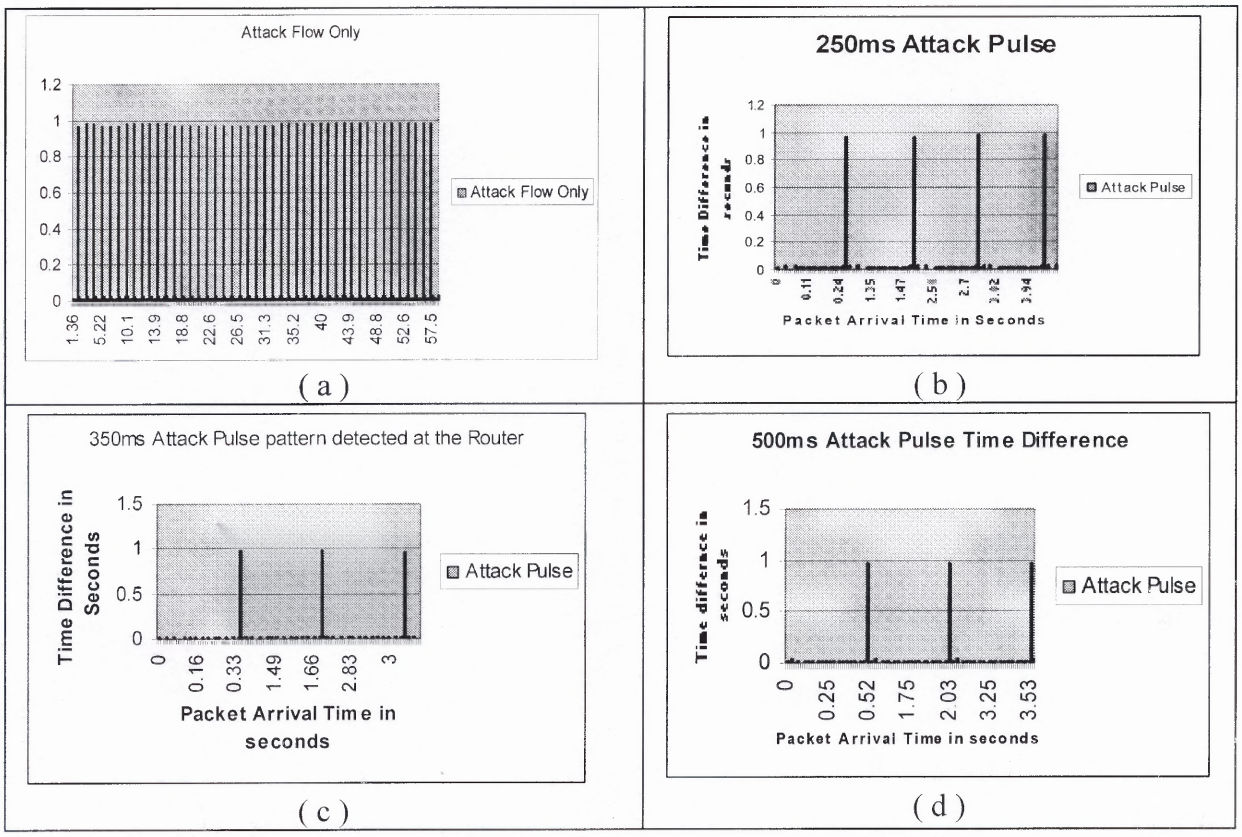


Figure 6.7 Attack Patterns with different burst lengths

As described in the previous chapters, the time difference or the inter-arrival time of the attack packets is effectively captured, and when analyzed, the attack pattern's T and L value are as defined in [1]. This is a clear indication that the pattern is malicious and should be blocked to avoid further damage to the network. The distribution provides results as expected by the scheme, and over repeated observations the flow(s) can be classified as attack flow(s). From these simulation results, we can conclude that the scheme can effectively detect distributed low rate attacks and multi-rate attacks. This effective framework provides a very effective way to counter low rate attacks, and is also scalable to provide extensive applications to online flow monitoring operations.

CHAPTER 7

CONCLUSION

In this work, a new detection scheme has been proposed to detect and defend against the low rate TCP DoS attacks. The focus of this thesis is to analyze the low rate attack by modeling the network under attacks and to propose a detection scheme to defend against such attacks. This is achieved by incorporating an effective, scalable lightweight data structure in the edge-router to build the flow history and to analyze the flows. An effective memory model is suggested to perform memory management operations and prevent a memory overflow in a router under an attack. All the schemes used today have to be preempted by the victim or by the source to start a detection process to mitigate a DDoS attack. A novel approach, which aims at providing an autonomous solution and does not require any premption from any source, has been proposed in this work. This is the first proposal to detect the low rate TCP DDoS attack at edge routers using a router-based approach. The detection scheme is scalable, and can also be deployed easily. This work can also fuel more relevant follow up research activities to build a fool proof autonomous system to guard against such attacks.

APPENDIX

CODE DOCUMENTATION

A.1 Data Structure Design

The data structure design is accomplished by the design of classes and structures used to house the data of each flow object. As described in Chapter 5, each flow is assigned an object, and each packet's information is stored as an attribute under the object to build the history for each flow. These flow attributes are refreshed according to a memory management model discussed in Chapter 5. The following sections give a detailed overview of the object structure and the attributes and the methods exhibited by these objects.

A.2 Object Hierarchy

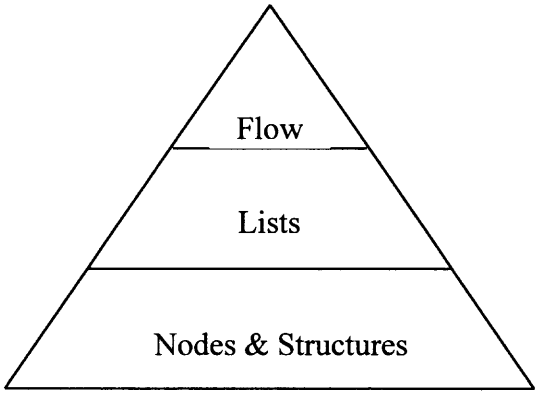


Figure A.1 Object hierarchy of the data structure

A.2.1 Structures: The lowest level of this data structure is a user-defined data-type called a structure. Structures are specially designed to hold data of every passing packet

for performing attack analysis. Structure ***flwdata*** and ***packethistory*** are defined for this purpose.

Flwdata Structure: This structure stores the basic information pertaining to a flow, and then stores the head pointer of the attribute list. Details of the structure members are given in the structure definition shown below.

```
struct flwdata
{
      int flowid; // flowid -> flowid of the tcp flow
      int ssn,ean;// ssn = source sequence num; ean = expected ack num
      int rsn,esn;  //rsn = reciever sequence num; esn = expected
                    //  sequence number from source;
      int sa, da;  // sa-> source address for simulation ; da ->
                   //   destination address for simulation
      float sendtime, recievetime;  //sendtime stores the time the
                                    //packet left the source/attacker
                              //and recievetime stores the time the
                              // packet left the destination/victim
      int datactr,ackctr ; //datactr -> counts data packets and ackctr
                           // counts ackpackets
      int rto;     // rto variable used to track the flow's rto;
      bool blocked; // blocked flag used to indicate block
      bool trigger;     //flow's packets and trigger flag used to
                    // initiate network management alert
};
```

Packet History Structure: This is an element of the attribute list of every flow object and of every flow. The structure stores all the data required by the monitoring system to analyze the flows. The highlight of this structure is the packet arrival time field which plays a vital role in deriving the time difference distribution and then estimating the attack parameters. Each attribute node stores all the packet information from the router's perspective. The object value of the same attribute object is different at different routers as all the values stored in the attribute is from that particular router's perspective. This

relative information is useful in providing an autonomous solution. The definition of the structure is as shown below.

```
struct packethistory {

  int psegno;  // Stores the packet segment number
  int pdatasize;  // Stores packet data size;
  double parrivaltime;  // Stores packet arrival time
  double erto;     // Stores the expected RTO
  int segrto;      // Stores the Segement RTO
  double td; // Time Diff.to check between prev packet and present one
  int rttperflow; // Verifying C1 - Amey -under test..
  //  bool acked, packed; // Stores if the packet was acked or not
 // int segpacked;

};
```

A.2.2 Nodes: A node is a basic unit of information in a list. It is divided into a data portion and an address portion. The data portion stores data in standard data types or user defined data types like structures. Nodes in this implementation have either Flwdata as the data portion or the packethistory structure for the data portion, and the address portion is divided into two. One address corresponds to the first node of the attribute list and the other address portion points to the next node of the flow object list. The abstract class definition of the flow node is shown below.

```
class fnode
{
public:
      flwdata *flowobj;  // Data portion of the Node
      fnode *next;        // Nextflow Object pointer
      plist *phlist;// Head ptr of the packet hist. list of every flow
      fnode();            // Constructor
      flwdata* getflow(); // Method used to send out the flowdata
                          // outside the flow list.
      bool setflow(flwdata *flowobject);  // Method used to populate
                                          //the flow object in the list
      bool UpdateNodeHistory(packethistory *pobj); // Function to
      //update the attribute list or append new atriutes to the list
      ~fnode();                  // Default Destructor
};
```

A.2.3 History Node: As described before, a history node is used to store the history of the flow. Each history node points to another in the second dimension. The history nodes are referenced by a packet list pointer whose head pointer is housed in the flow nodes discussed earlier. The abstract class definition of the history node is shown below.

```
class hnode
{
public:
      packethistory *pobj;  // Attribute values stored in the structure
      hnode *next;          // packet list pointer for history node.
      hnode();              // Default constructor
      packethistory* gethistorydata(); // Method used to send out the
                  // Packet history data  outside the history list.
      bool sethistorydata(packethistory *pobj);  // Method used to
                        // populate the history object in the list
      ~hnode();         // Default destructor
};
```

The history node is a simple node with 2 functionalities apart from its default constructor and destructor. The hnode class implements the "gethistorydata" method whch is used to acquire history data from the node, and the "sethistorydata" method which is used to append or add new attributes to the history list structure. These operations are performed during the routine data monitoring and during the attack detection phases.

A.2.4 Lists: The design involves list structures in 2 dimensional structure of linked lists. One dimension of the linked list is used to maintain the object flows, and the other dimension of the list structure is used to maintain the attributes of the flows. The main list holds the flow ids and the connection information of each flow. This list class is called *list.* The second list which maintains the attributes is defined by the class *plist.* The following section provides a more detailed description of the list classes.

List *list*: The class list, which is defined to perform the flow operations, houses many methods and attributes. It basically performs addition, deletion, traversing and printing operations on the list based on the flow ids. An object of this list class is instantiated by its inherited class *flow*, which will be discussed later. The list maintains all the active flow information and performs the analysis over these flows. The abstract definition of the class list is shown below.

```
class list
{
private:
      fnode *head;  // Head pointer of the flow list

public:
      list();           // Default Constructor
      void addnode(int flowid_);    //    Adds a new flow to the list
        bool addnode(flwdata *flowobj); //Adds a new flow to the list
      bool delnode(int flowid_);    //    DELETES A FLOW FROM THE LIST
//    bool travnode(int flowid_);// Traverses a list based on flow id
      flwdata* travnode (int flowid_);// Trav a list based on flow id
      fnode* find_previous(int flowid_); // Finds the previous node
      void printnodes();   // Prints the list structure - For Debugging
};
```

List *plist*: The packet list, which is linked to each flow object, is defined by the class *plist*. An object of this class is instantiated in the flow node class used as the data part of the flow list. This object houses the head pointer for the attribute list, and hence every flow node has an access to the attributes of its flow. The abstract class definition of the class plist is shown below.

```
class plist {
private:
      hnode *head;

public:
        plist();
      void addnode(int psegno, int pdatasize, double parrivaltime,
double erto, int segrto, int rttperflow);
        bool addnode(packethistory *pobj);
//      bool delnode(int pdatasize);
```

```
//     bool travnode(int pdatasize);
//     packethistory* travnode (int pdatasize);
//     hnode* find_previous(double parrivaltime);
       void printnodes();
       double GetArrivalTime();
       int countnodes();
};
```

The methods exhibited by the class plist are almost the same as the members of the class list. However, these methods apply to the attributes of the flows.

A.2.5 Flows: This is the uppermost data structure used for attack detection operations. The class *flow* defining the flow object is a super structure that has reference to the entire flow list and their attribute structure. This is accomplished by instantiating an object of the flow list as a private member of the flow class. The public member functions use this flow list object to perform the scanning and detection operations. An object of this flow class is instantiated in the dequeue module of the red queues. The implementation is placed in the /../ns-2.26/queues/red.cc file. All the operations are performed on the ingress traffic. The abstract definition of the flow class is shown below.

```
class flow {

private :
       list  *flowlst;
       int ModifySequenceNumber(list *flowlst, Packet *p );
       bool CalculateVirtualAcknowledgement(list *flowlst, Packet *p );

public:

       flow();
       ~flow();
       bool CreateFlowObject(flwdata *flowobj); // Creates a flow object
       bool DeleteFlowObject(list *flowlst, Packet *p ); // Deletes flow
       bool IsListEmpty();      // TO check if list is empty
       bool UpdateFlowObject(list *flowlst, flwdata *obj, packethistory
*pobj);      // Update flow bject
/*     void UpdateFlowHistory(list *flowlst, flwdata *obj, packethistory
*pobj)       // Update flow history

       List* UpdateObject(list *flowlst, Packet *p);
bool ScanIPPacketToList(Packet *p);  // Use to scan Incoming IP Packets
};
```

The scanIPPacket is the entry point to the scheme, and it scans the packets and decides if a flow object has to be created for the packet or not. It also performs monitoring operations, and based on the results calls the other member functions in the class like UpdateFlowObject, UpdateObject, etc. It acts as the flow classifier described in Chapter 5. The complete scheme implementation is scalable, and can be extended for other attack detection schemes.

REFERENCES

1. A. Kuzmanovic and E. W. Knightly, "Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants)", In proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany, August 2003, pp. 75 - 86 .

2. G. Yang, M. Gerla, M. Y. Sanadidi, "Randomization: Defense against Lowrate TCP-targeted Denial-of-Service Attacks", ISCC 2004, pp. 345-350.

3. A. Belenky and N. Ansari, "Tracing Multiple Attackers with Deterministic Packet Marking (DPM)," Proc. 2003 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM '03), Victoria, B.C., Canada, August 28-30, 2003, pp. 49-52.

4. Z. Gao, N. Ansari, and K. Anantharam "A New Marking Scheme to Defend against DDoS Attacks," Proc. IEEE GLOBECOM'04, Dallas, TX, USA, Nov. 29 – Dec. 3, 2004

5. S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network support for IP traceback", IEEE/ACM Transactions on Networking, vol. 9, no. 3,pp 226-237, June 2001.

6. CERT Coordination Center, "Denial of Service Attacks," http://www.cert.org/tech_tips/denial_of_service.html

7. R. Basu, R.K. Cunningham, , S.E. Webster, and Richard P. Lippmann, "Detecting Low-Profile Probes and Novel Denial of Service Attacks", Proceedings of 2001 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, 5-6 June, 2001.

8. A. Hussain, J. Heidemann, C. Papadopulos, " A Framework for Classifying Denial of Service Attacks", Proceedings of SIGCOMM 03 August 25-29 2003, Karsruhe, Germany, pp. 99 - 110.

9. D. Xuan, R. Bettati and W. Zhao, "A Gateway Based Defense System for Distributed Dos Attacks in High Speed Networks", Proceedings of 2001 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY 5-6 June, 2001.

10. M. Allman and V. Paxson, "On estimating end-to-end network path properties," Proceedings of ACM SIGCOMM 1999, Cambridge, MA, August – September 1999 pp. 263-274.

11. M. Fomenkov, K. Keys, D. Moore and K. Claffy, "Longitudinal study of the Internet traffic in 1998 -2003".

12. S. Floyd and V. Paxson, "Difficulties in Simulating the Internet" IEEE/ACM Transactions on Networking (TON), Volume 9 , Issue 4 (August 2001), pp. 392 – 403.

13. S. Floyd, E. Kohler, "Internet research needs better models", SIGCOMM 03,Volume 33, Issue 1(January 2003), pp.29 - 34

14. S. Zhang, P. Dasgupta," Denying Denial of Service Attacks: A router based Solution", The 2003 International Conference on Internet Computing, pp. 301-307, June 2003.

15. Tools for the Analysis of Trace-route Samples, http://klamath.stanford.edu /tools /Traceroute