Mitigation of DDoS Attack using a Probabilistic Approach & End System based Strategy

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

Specialization: Information Security

by

Biswa Ranjan Swain

Department of Computer Science and Engineering
National Institute of Technology
Rourkela-769 008, Orissa, India

May 2009

Mitigation of DDoS Attack using a Probabilistic Approach & End System based Strategy

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering
Specialization: Information Security

by

Biswa Ranjan Swain

under the guidance of

Prof. Bibhudatta Sahoo

Department of Computer Science and Engineering
National Institute of Technology
Rourkela-769 008, Orissa, India

May 2009

To my parents

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India.

Certificate

This is to certify that the work in the thesis entitled ***Mitigation of DDoS Attack using a Probabilistic Approach & End System based Strategy*** submitted by ***Mr.Biswa Ranjan Swain*** in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering, Specialization: Information Security during the session 2008–2009 in the department of Computer Science and Engineering, National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Prof. Bibhudatta Sahoo
Senior Lecturer
Dept. of Computer Science & Engineering
National Institute of Technology
Rourkela-769008 Orissa (India)

Place: NIT Rourkela
Date: 26 May 2009

Acknowledgment

My first thanks are to the Almighty, without whose blessings I wouldn't have been writing this "acknowledgments".

I then would like to express my heartfelt thanks to my supervisor, Sr. Lecturer Bibhudatta Sahoo, for his guidance, support, and encouragement during the course of my master study at the National Institute of Technology, Rourkela. I am especially indebted to him for teaching me both research and writing skills, which have been proven beneficial for my current research and future career. Without his endless efforts, knowledge, patience, and answers to my numerous questions, this research would have never been possible.The experimental methods and results presented in this thesis have been influenced by him in one way or the other. It has been a great honor and pleasure for me to do research under Prof. Bibhudatta's supervision.

I am very much indebted to Dr. B. Majhi, Head of the Department, Computer Science engineering, National Institute of Technology, Rourkela for his support during my work.

I am also thankful to Dr. S. K. Rath, Dr. S. K. Jena, Dr. D. P. Mohapatra, Dr. R. Baliarsingh, Dr. A.K. Turuk, Dr. P. M. Khillar for giving encouragement during my thesis work.

I thank Mr. Pushpendra Kumar Chandra for his helping hand during the preparation of thesis and learning of simulation tools.

I thank my friends & all the members of the Department of Computer Science and Engineering, and the Institute, who helped me by providing the necessary resources, and in various other ways, in the completion of my work.

Finally, I want to dedicate this thesis to my parents,my family members and my beloved one for their unlimited support and strength. Without their dedication and dependability, I could not have pursued my MTech. degree at the National Institute of Technology Rourkela.

Biswa Ranjan Swain

Abstract

From the very begining of Internet Technology, the attacks are the undetatchable element of it. This Distributed Denial-of-Service attack is very much harmful as our systems are vulnerable to this. Many mitigation strategies were proposed but these all need the help of network services for mitigation. Also these need the administrative support to continue the mitigation technique beyond own network if the attacker is outsider of own network. But it is very hard to meet all these constraint at a time. That's why we need the help of "end system" based mitigation strategy. Very rare researches have been carried out on this type of method.

In this thesis, we have followed the end system based mitigation strategy to solve the problem of DDoS attack. Here neither we need the support of network services nor any administration facility. Here we proposed one probabilistic approach to find out the number of packets being malicious among the massive number of packets. Also we have proposed one algorithm to mitigate the attack based on the number of packets being malicious. We also analyzed our approach in a simulation environment against one existing end system based mitigation strategy. The result shows that our approach solves the problem DDoS and saves the computational resource in terms of CPU time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Introduction

DDoS Impact

Problem Addressed

Design Approach

Layout of the Thesis

Chapter 1

Introduction

1.1 Introduction

We are living in the information age. We need to keep the information about every aspect of our life. In other words, information is an asset that has a value like any other asset. As an asset, information needs to be secured from attacks. To be secured information needs to be hidden from unauthorized access (confidentiality), protected from unauthorized changes (integrity), and available to an authorized entity when it is needed (availability) [1]. With the advent of computers, information storage became electronic instead of storing in physical files. The three security requirements, however, did not change. The files are stored in computers require confidentiality, integrity, and availability. The implementation of these requirements, however, is difficult and more challenging.

So Distributed Denial of Service (DDoS) attack will make the system not to be available for the legitimate purpose. So here one of the security goals is affected that is availability. DDoS attack is a threat to the availability. Our legitimate system is being unavailable to serve. This attack may slow down or totally interrupt the service of a system. The attacker can have several strategies to achieve this. It might send so many bogus requests to a server that the server crashes because of the heavy load. The attacker might intercept and delete a server's response to a client, making the client to believe that the server is not responding. The attacker may also intercept requests from the clients, causing the clients to send requests many times and overload the system.

A Denial of Service (DoS) attack can be characterized as an attack with the purpose of preventing legitimate users from using a victim computing system or network resource [2]. A Distributed Denial of Service (DDoS) attack is a large-scale, coordinated attack on the availability of services of a victim system or network resource, launched indirectly through many compromised computers on the Internet. The services under attack are those of the "primary victim", while the compromised systems used to launch the attack are often called the "secondary victims" [2]. The use of secondary victims in performing a DDoS attack provides the attacker with the ability to wage a much larger and more disruptive attack, while making it more difficult to track down the original attacker.

A Distributed Denial of Service (DDoS) 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 [3].

1.2 DDoS Impact

Distributed Denial-of-service (DoS) attacks have been around for a long time. In the computer network arena, DDoS attacks usually take one of two forms [4]: (1) exploiting bugs in network clients or server applications, in an attempt to crash the application (and possibly the host on which it is running) or (2) flooding a network server with fake traffic, making it difficult or impossible for the server to receive and process legitimate traffic. The former are typically carried out by using 'buffer overrun attacks' in which a network application is sent a large amount of data which it fails to handle properly, instead overwriting critical information with the excess data.

Some companies do not take security seriously enough and their systems, in general, are easily compromised and pose a threat not only to the companies themselves but also to anyone else targeted by a hacker through their systems. Defending against DoS attacks involves using secure operating systems such as

Unix which offer process protection (to prevent an application crash from crashing the whole system), keeping up-to-date with security patches and vulnerability alerts (to avoid running applications which are vulnerable to buffer overrun attacks) and monitoring and controlling network traffic (to handle flood attacks). DDoS attacks are a new variation on this old theme. A DDoS attack uses network flooding, but is harder to defend against because the attack is launched from hundreds or even thousands of hosts simultaneously. Rather than appearing as an excess of traffic coming from a single host, a DDoS attack appears instead as normal traffic coming from a large number of hosts. This makes it harder to identify and control.

Even when an attack has been identified, it can be difficult if not impossible to trace back to its origins as so many compromised hosts are involved. When there are so many hosts involved, the logistical problems of stemming the attack and identifying its real origin are enormous. To make matters worse, the programs used by the attackers commonly use 'address spoofing' [4] - that is, they put fake source addresses in the packets that they send out. In such a case, tracing the attack back involves examining the logs of all of the intermediate routers, one by one, to trace the packets back one hop at a time. This renders the task almost impossible, so it is no surprise that no arrests have yet been made for these attacks. Not only are these attacks hard to trace back, but there is no comprehensive defense against them either. The Internet was simply not designed with these vulnerabilities in mind, and a real solution would involve re-engineering the entire network architecture. This means it is critical to take pre-emptive measures to reduce the possibility of these attacks and minimize their impact.

It is vitally important to prevent your systems from being used to launch these attacks. The compromised systems are usually taken over by one of two methods, namely password cracking and buffer overruns [4]. Educating users about choosing good passwords is also necessary. Dictionary-based attack programs are common, so single word passwords are not a good choice. Passwords should contain a mix of letters and digits or punctuation, be hard to guess, kept secure and changed

regularly. Buffer overrun attacks used to compromise systems are a more sophisticated variation of the DDoS buffer overrun attacks, in which the excess data is carefully constructed to be meaningful instructions which are then executed by the host under attack. To protect against these, you need to use servers that have been well written and carefully audited for such bugs. A firewall can often provide a good defense here, if your servers are located behind it. A good firewall will have all buffer operations carefully controlled, and will identify, log and handle overrun attempts. A firewall can also issue alerts, for example by sending SMS messages to mobile phones, which will allow countermeasures to be taken as early as possible. Preventing address spoofing can also help to make the origins of attacks harder to conceal. Ingress and egress packet filtering should be used on firewalls or routers to prevent packets with spoofed addresses from crossing these boundaries. To defend yourself against being the victim of an attack, you need to ensure that you have ample network bandwidth, that your systems are not vulnerable to regular DDoS attacks, and that your systems provide redundancy.

Due to this DDoS attack there are very best possibilities to have an attack on the routing information at the various routers available in the route. The packets are entering to the router may have some codes which will changes the routing table so that the table can be maintained according to the attacker's wish and the ongoing attack can not be caught.

Also this DDoS attack is very much harmful in case of the network components' survivability. It may make busy the network components so that these can not respond to the legitimate requests and also will behave as malfunctioning. So the conclusion of the discussion is the impact of this attack is on three components of the system,

Consumption of computational resource, such as bandwidth, disk space, or CPU time, Disruption of configuration information, such as routing information, Disruption of physical network components.

1.3 Problem Addressed

From the above discussion it was assessed that how much dangerous the DDoS attack is. This attack hampers the server in two different modes. It hampers in terms of resources available at server side and the bandwidth of the channel used to avail the service from the server. This resource depletion attack is described in Chapter 2. Where the resources are likely to be tie up the network resources (i.e. misuse of the protocol execution) and the resources available at the server (i.e. the processing time will be increased). So many solutions to solve the problem to mitigate the attack from hampering the resources at server have been proposed. Amongst them so many are proposed for saving the server from being attacked is of having the support from network. But very rare proposals have been made to solve this problem without using the support from network. In HCF (Hop-count filtering) method they have focused upon the value of TTL in [5] in which an IP packet header is checked to know the TTL value, whether the packet is malicious or not. This checking of the TTL is done at the end system, so that we don't need the help of network services. Detail about this method is discussed in Chapter 3. This approach is used to mitigate the attack as well as it saves some computational time of implementing it in comparison to other solution. As TTL value of a packet can't be changed by any intruder in transit of a packet so this TTL will play a vital role through out this thesis. This HCF method was only mitigating the DDoS attack but saving the time was not in its concern. So we focused upon the mitigation of the DDoS attack as well as saving the computational time.

1.4 Design Approach

Implementation of one algorithm at server side is not much easy. This approach will be implemented at a port where all the packets coming to the server will be analyzed and then will be allowed to enter the server. A probabilistic approach and a filtering function will be maintained in the module to allow the packets into the server. Here we don't need to bother about the queuing analysis as we are not checking the whole packet. We are just checking the header of the packet and that

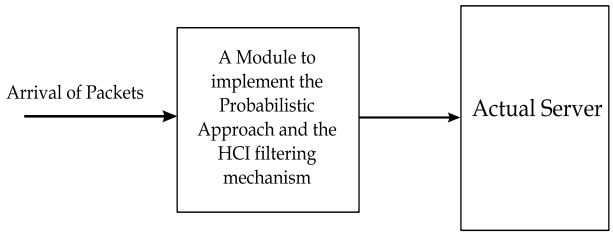

Figure 1.1: Implementation Model Structure

is only the TTL value of the packet. So processing a large packet or processing a small packet doesn't have any effect on the performance of the module for checking. From the probabilistic approach we are having the formula, which depends upon the arrival rate of packets and the error probability of a packet arriving at the server. This probabilistic approach is clearly described in Chapter 3. Then we are checking the TTL values to confirm the packet is spoofed or not using the algorithm described in Chapter 3. Our approach is saving the resource available at the server in terms of time.

1.5 Layout of the Thesis

This thesis is divided into five chapters. The Chapter 1, which is here, we have given an overview of the implementation of algorithms in a victim server and the resource saving analysis at the server. Chapter 2 describes the detailed overview of the related work had been done in this area and also the detailed classification of the attack and various mitigation methods. At the end of the chapter, we have just classified the various mitigation methods on the basis of comprehensive solution with its underlying strategies. Chapter 3 begins with the probabilistic approach to calculate the number of packets being malicious among the massive number of packets. And also the Hop-Count Filtering (HCF) method is discussed over there. Chapter 4 describes the solution to the problem addressed in the introductory part of this thesis. It also contains the experimental results to prove the proposed system. Chapter 5 contains the conclusion of this thesis and the future enhancements can be done to this work.

Chapter 2

Distributed Denial-of-Service Attack & Related Work

Denial-of-Service Attack (DoS)

Distributed Denial-of-Service Attack (DDoS)

DDoS Attack Network

DDoS Taxonomy

DDoS Countermeasures Taxonomy

Related Work

Chapter 2

Distributed Denial-of-Service Attack and Related Work

2.1 Denial-of-Service Attack (DoS)

A DoS attack can be described as an attack designed to render a computer or network incapable of providing normal services. A DoS attack is considered to take place only when access to a computer or network resource is intentionally blocked or degraded as a result of malicious action taken by another user. These attacks don't necessarily damage data directly, or permanently, but they compromise the availability of the resources.

DoS attacks can be classified in [6] as follows:

Network Device Level: DoS attacks in the Network Device Level include attacks that might be caused either by taking advantage of bugs in software or by trying to exhaust the hardware resources of network devices.

OS Level: In the OS Level DoS attacks take advantage of the ways operating systems implement protocols.

Application-based attacks: A great number of attacks try to settle a machine or a service out of order either by taking advantage of specific bugs in network applications that are running on the target host or by using such applications to dram the resources of their victim.

Data Flooding: An attacker may attempt to use the bandwidth available to a network, host or device at its greatest extent, by sending massive quantities of data and so causing it to process extremely large amounts of data.

Attacks based on protocol features: DoS may take advantage of certain standard protocol features, for example several attacks exploit the fact that source addresses can be spoofed.

2.2 Distributed Denial-of-Service Attack (DDoS)

A DDoS 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 DoS significantly by harnessing the resources of multiple unwitting accomplice computers, which serve as attack platforms.
A DDoS attack is composed of four elements in [6]:

1. The real attacker.
2. The handlers or master compromised hosts, who are capable of controlling multiple agents.
3. The attack daemon agents or zombie hosts, who are responsible for generating a stream of packets toward the intended victim.
4. A victim or target host.

A DDoS attack can be described as follows:
Recruitment: The attacker chooses the vulnerable agents, which will he used to perform the attack.
Compromise: The attacker exploits the vulnerabilities of the agents and plants the attack code, protecting it simultaneously from discovery and deactivation.
Communication: The agents inform the attacker via handlers that they are ready.
Attack: The attacker commands the onset of the attack.
Sophisticated and powerful DDoS toolkits are available to potential attackers increasing the danger of becoming a victim in DoS or DDoS attack. Some of the most known DDoS tools are Trinoo, TFN, Stacheldraht, TFN2K, mstream and Shaft [6].

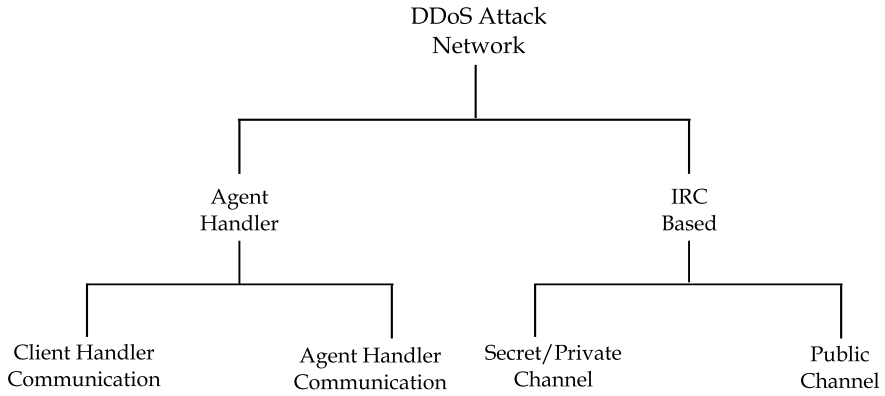

Figure 2.1: DDoS Attack Network

2.3 DDoS Attack Network

There are two main types of DDoS attack networks: The Agent-Handler model and The Internet Relay Chat (IRC-Based) [2] model (See Figure 2.1).

2.3.1 Agent-Handler Model

An Agent-Handler DDoS attack network consists of clients, handlers, and agents (see Figure 2.2). The client platform is where the attacker communicates with the rest of the DDoS attack network. The handlers are software packages located on computing systems throughout the Internet that the attacker uses to communicate indirectly with the agents. The agent software exists in compromised systems that will eventually carry out the attack on the victim system. The attacker communicates with any number of handlers to identify which agents are up and running, when to schedule attacks, or when to upgrade agents.

Depending on how the attacker configures the DDoS attack network, agents can be instructed to communicate with a single handler or multiple handlers. Usually, attackers will try and place the handler software on a compromised router or network server that handles large volumes of traffic. This makes it harder to identify messages between the client and handler and between the handler and agents. The communication between attacker and handler and between the handler and agents can be via TCP, UDP, or ICMP protocols. The owners and users of the agent systems typically have no knowledge that their system has been

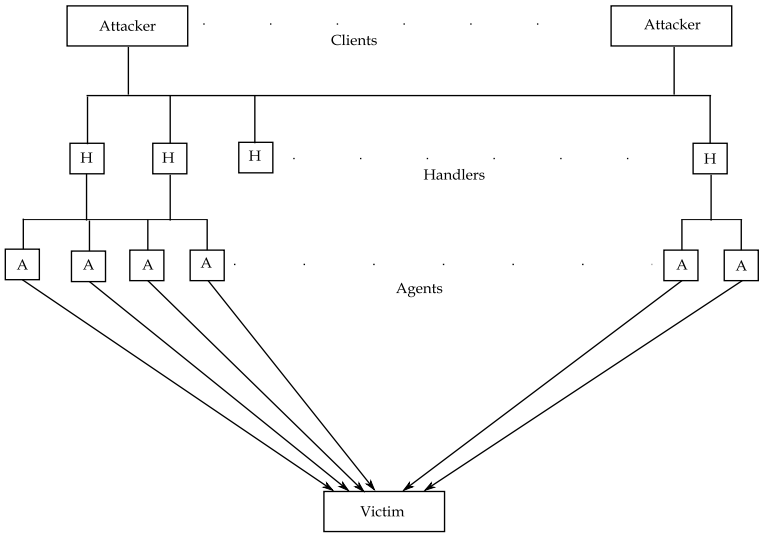

Figure 2.2: DDoS Agent Handler Attack Model

compromised and will be taking part in a DDoS attack. When participating in a DDoS attack, each agent program uses only a small amount of resources (both in memory and bandwidth), so that the users of these computers experience minimal change in performance.

In descriptions of DDoS tools, the terms handler and agents are sometimes replaced with master and daemons respectively. Also, the systems that have been violated to run the agent software are referred to as the secondary victims, while the target of the DDoS attack is called the (primary) victim.

2.3.2 IRC Based DDoS Attack Model

Internet Relay Chat (IRC) is a multi-user, on-line chatting system. It allows computer users to create two-party or multi-party interconnections and type messages in real time to each other. IRC network architectures consist of IRC servers that are located throughout the Internet with channels to communicate with each other across the Internet. IRC chat networks allow their users to create public, private and secret channels. Public channels are channels where multiple users can chat and share messages and files. Public channels allow users of the channel to see all the IRC names and messages of users in the channel. Private and secret channels are set up by users to communicate with only other designated users. Both pri-

vate and secret channels protect the names and messages of users that are logged on from users who do not have access to the channel. Although the content of private channels is hidden, certain channel locator commands will allow users not on the channel to identify its existence, whereas secret channels are much harder to locate unless the user is a member of the channel.

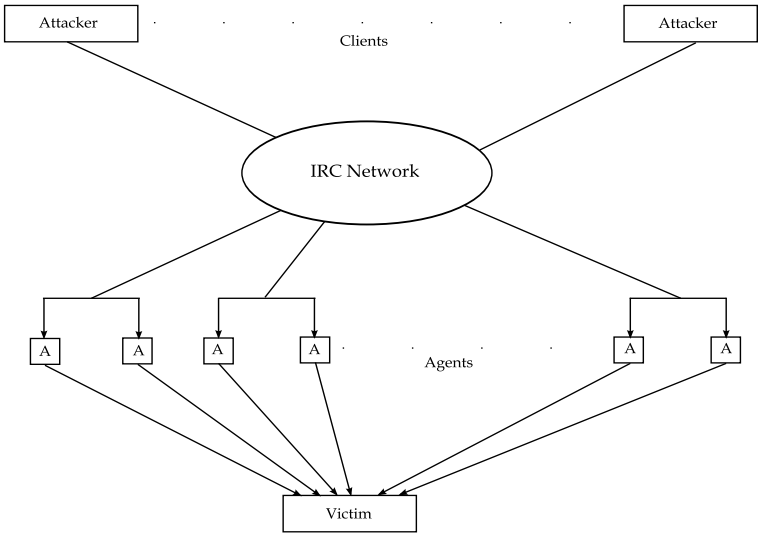

Figure 2.3: DDoS IRC Based Attack Model

An IRC-Based DDoS attack network is similar to the Agent-Handler DDoS attack model except that instead of using a handler program installed on a network server, an IRC communication channel is used to connect the client to the agents. By making use of an IRC channel, attackers using this type of DDoS attack architecture have additional benefits. For example, attackers can use "legitimate" IRC ports for sending commands to the agents. This makes tracking the DDoS command packets much more difficult. Additionally, IRC servers tend to have large volumes of traffic making it easier for the attacker to hide his presence from a network administrator. A third advantage is that the attacker no longer needs to maintain a list of agents, since he can simply log on to the IRC server and see a list of all available agents. The agent software installed in the IRC network usually communicates to the IRC channel and notifies the attacker when the agent is up and running. A fourth advantage is that IRC networks also provide the benefit of easy file sharing. This makes it easier for attackers to secure secondary victims to

participate in their attacks.

In IRC-based DDoS attack architecture, the agents are often referred to as "Zombie Bots" or "Bots". In both IRC-based and Agent-Handler DDoS attack models, we will refer to the agents as "secondary victims" or "zombies".

2.4 DDoS Taxonomy

There are a wide variety of DDoS attack techniques. Taxonomy of the main DDoS attack methods was proposed in [2]. There are two main classes of DDoS attacks: bandwidth depletion and resource depletion attacks in figure 2.4. A bandwidth depletion attack is designed to flood the victim network with unwanted traffic that prevents legitimate traffic from reaching the (primary) victim system. A resource depletion attack is an attack that is designed to tie up the resources of a victim system. This type of attack targets a server or process on the victim system making it unable to process legitimate requests for service.

2.4.1 Bandwidth Depletion Attacks

There are two main classes of DDoS bandwidth depletion attacks [6]. A flood attack involves the zombies sending large volumes of traffic to a victim system, to congest the victim system's bandwidth. An amplification attack involves either the attacker or the zombies sending messages to a broadcast IP address, using this to cause all systems in the subnet reached by the broadcast address to send a message to the victim system. This method amplifies malicious traffic that reduces the victim system's bandwidth.

Flood Attacks: In a DDoS flood attack the zombies flood the victim system with IP traffic. The large volume of packets sent by the zombies to the victim system slows it down, crashes the system or saturates the network bandwidth. This prevents legitimate users from accessing the victim. Figures 2.2 and 2.3 indicate a flood attack for an Agent-Handler attack network and an IRC-based attack network.

UDP flood attack: User Datagram Protocol (UDP) is a connectionless protocol. When data packets are sent via UDP, there is no handshaking required between sender and receiver, and the receiving system will just receive packets it must process. A large number of UDP packets sent to a victim system can saturate the network, depleting the bandwidth available for legitimate service requests to the victim system.

In a DDoS UDP Flood attack, the UDP packets are sent to either random or specified ports on the victim system. Typically, UDP flood attacks are designed to attack random victim ports. This causes the victim system to process the incoming data to try to determine which applications have requested data. If the victim system is not running any applications on the targeted port, then the victim system will send out an ICMP packet to the sending system indicating a "destination port unreachable" message [7].

Often, the attacking DDoS tool will also spoof the source IP address of the attacking packets. This helps hide the identity of the secondary victims and it insures that return packets from the victim system are not sent back to the zombies, but to another computer with the spoofed address.

UDP flood attacks may also fill the bandwidth of connections located around the victim system (depending on the network architecture and line-speed). This can sometimes cause systems connected to a network near a victim system to experience problems with their connectivity.

ICMP Flood Attacks: Internet Control Message Protocol (ICMP) packets are designed for network management features such as locating network equipment and determining the number of hops or round-trip-time to get from the source location to the destination [8]. For instance, ICMP_ECHO_REPLY packets ("ping") allow the user to send a request to a destination system and receive a response with the roundtrip time.

A DDoS ICMP flood attack occurs when the zombies send large volumes of ICMP_ECHO_REPLY packets to the victim system. These packets signal the victim system to reply and the combination of traffic saturates the bandwidth

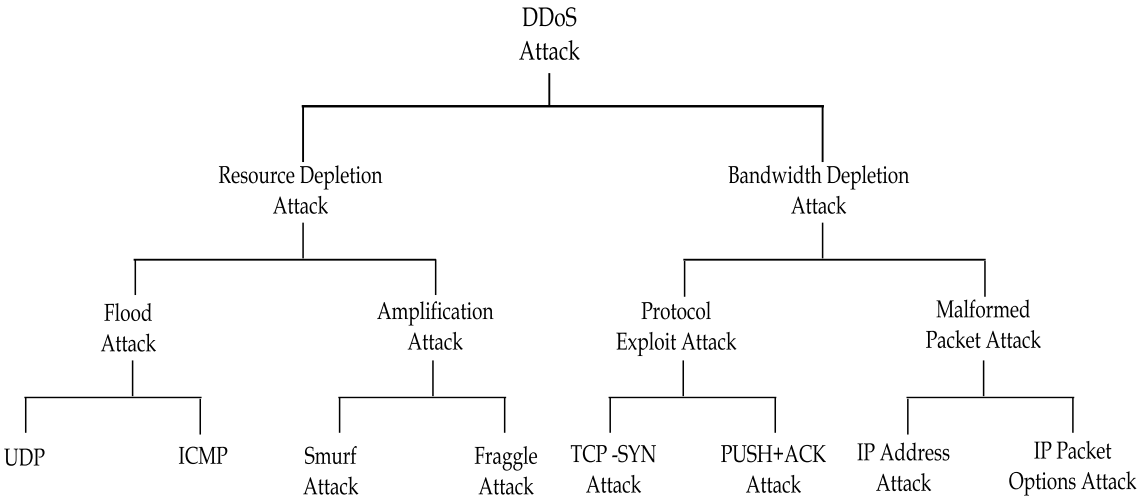

Figure 2.4: DDoS Attack Taxonomy

of the victim's network connection. As for the UDP flood attack, the source IP address may be spoofed.

Amplification Attacks: A DDoS amplification attack is aimed at using the broadcast IP address feature found on most routers to amplify and reflect the attack (see Figure 2.5). This feature allows a sending system to specify a broadcast IP address as the destination address rather than a specific address. This instructs the routers servicing the packets within the network to send them to all the IP addresses within the broadcast address range.

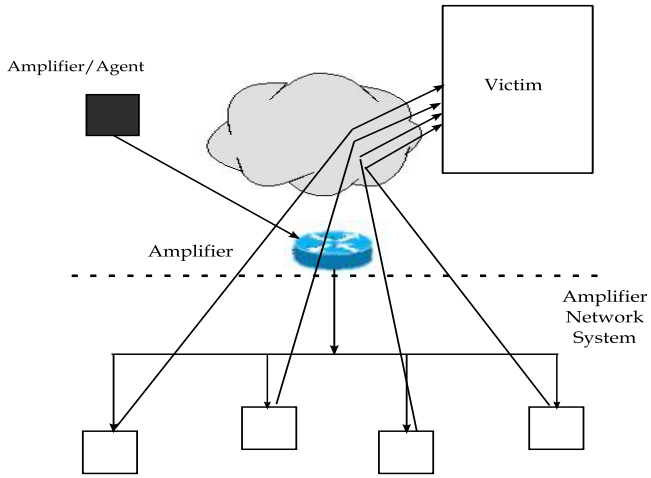

Figure 2.5: Amplification Attack

For this type of DDoS attack, the attacker can send the broadcast message directly, or the attacker can use the agents to send the broadcast message to in-

crease the volume of attacking traffic. If the attacker decides to send the broadcast message directly, this attack provides the attacker with the ability to use the systems within the broadcast network as zombies without needing to infiltrate them or install any agent software. We further distinguish two types of amplification attacks, Smurf and Fraggle attacks [2].

Smurf Attacks: In a DDoS Smurf attack, the attacker sends packets to a network amplifier (a system supporting broadcast addressing), with the return address spoofed to the victim's IP address. The attacking packets are typically ICMP_ECHO_REQUESTs, which are packets (similar to a "ping") that request the receiver to generate an ICMP_ECHO_REPLY packet. The amplifier sends the ICMP_ECHO_REQUEST packets to all of the systems within the broadcast address range, and each of these systems will return an ICMP_ECHO_REPLY to the target victim's IP address. This type of attack amplifies the original packet tens or hundreds of times.

Fraggle Attacks: A DDoS Fraggle attack is similar to a Smurf attack in that the attacker sends packets to a network amplifier. Fraggle is different from Smurf in that Fraggle uses UDP_ECHO packets instead of ICMP_ECHO packets. There is a variation of the Fraggle attack where the UDP ECHO packets are sent to the port that supports character generation (chargen, port 19 in Unix systems), with the return address spoofed to the victim's echo service (echo, port 7 in Unix systems) creating an infinite loop. The UDP Fraggle packet will target the character generator in the systems reached by the broadcast address. These systems each generate a character to send to the echo service in the victim system, which will resend an echo packet back to the character generator, and the process repeats. This attack generates even more bad traffic and can create even more damaging effects than just a Smurf attack.

2.4.2 Resource Depletion Attacks

DDoS resource depletion attacks involve the attacker sending packets that misuse network protocol communications or sending malformed packets that tie up network resources so that none are left for legitimate users.

Protocol Exploit Attacks:

TCP SYN Attacks: The Transfer Control Protocol (TCP) includes a full handshake between sender and receiver, before data packets are sent. The initiating system sends a SYN (Synchronize) request (see Figure 2.6a). The receiving system sends an ACK (acknowledgement) with its own SYN request. The sending system then sends back its own ACK and communication can begin between the two systems. If the receiving system is sent a SYN_X packet but does not receive an ACK_Y+1 to the SYN_Y it sends back to the sender, the receiver will resend a new ACK + SYN_Y after some time has passed (see Figure 2.6b). The processor and memory resources at the receiving system are reserved for this TCP SYN request until a timeout occurs.

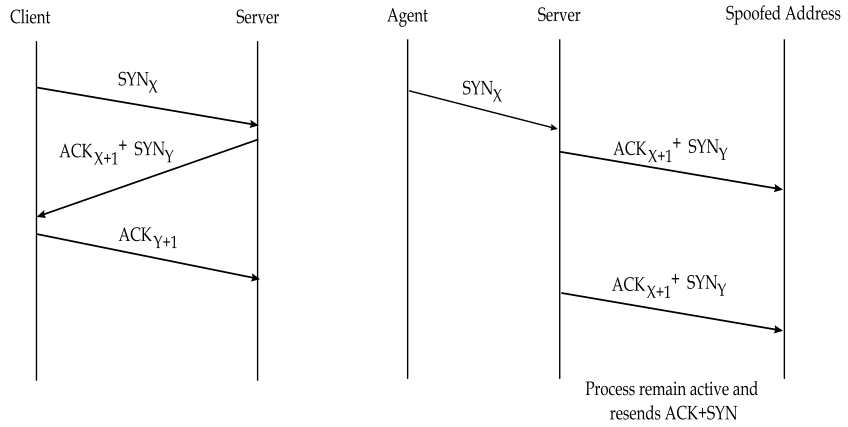

Figure 2.6: (a) TCP Synchronization ,(b) TCP-SYN Attack

In a DDoS TCP SYN attack, the attacker instructs the zombies to send such bogus TCP SYN requests to a victim server in order to tie up the server's processor resources, and hence prevent the server from responding to legitimate requests. The TCP SYN attack exploits the three-way handshake between the sending system and the receiving system by sending large volumes of TCP SYN packets to the victim system with spoofed source IP addresses, so the victim system responds to a non-requesting system with the ACK+SYN. When a large volume of SYN requests are being processed by a server and none of the ACK+SYN responses are returned, the server begins to run out of processor and memory resources. Eventually, if the volume of TCP SYN attack requests is large and they continue

over time, the victim system will run out of resources and be unable to respond to any legitimate users.

PUSH + ACK Attacks: In the TCP protocol, packets that are sent to a destination are buffered within the TCP stack and when the stack is full, the packets get sent on to the receiving system. However, the sender can request the receiving system to unload the contents of the buffer before the buffer becomes full by sending a packet with the PUSH bit set to one. PUSH is a one-bit flag within the TCP header. TCP stores incoming data in large blocks for passage on to the receiving system in order to minimize the processing overhead required by the receiving system each time it must unload a non-empty buffer.

The PUSH + ACK attack is similar to a TCP SYN attack in that its goal is to deplete the resources of the victim system. The attacking agents send TCP packets with the PUSH and ACK bits set to one. These packets instruct the victim system to unload all data in the TCP buffer (regardless of whether or not the buffer is full) and send an acknowledgement when complete. If this process is repeated with multiple agents, the receiving system cannot process the large volume of incoming packets and it will crash.

Malformed Packet Attacks:

A malformed packet attack is an attack where the attacker instructs the zombies to send incorrectly formed IP packets to the victim system in order to crash the victim system. There are two types of malformed packet attacks. In an IP address attack, the packet contains the same source and destination IP addresses. This can confuse the operating system of the victim system and cause the victim system to crash. In an IP packet options attack, a malformed packet may randomize the optional fields within an IP packet and set all quality of service bits to one so that the victim system must use additional processing time to analyze the traffic. If this attack is multiplied using enough agents, it can shut down the processing ability of the victim system.

2.5 DDoS Countermeasures Taxonomy

There are a number of proposals and partial solutions available today for mitigating the effects of a DDoS attack. Many of these solutions and ideas assist in preventing certain aspects of a DDoS attack. However, there is no comprehensive solution to protect against all known forms of DDoS attacks. Also, many derivative DDoS attacks are continually being developed by attackers to bypass each new countermeasure employed. More research is needed to develop more effective and encompassing countermeasures and solutions.

There are three essential components to DDoS countermeasures. There is the component for preventing the DDoS attack which includes preventing secondary victims and detecting and neutralizing handlers. There is the component for dealing with a DDoS attack while it is in progress, including detecting or preventing the attack, mitigating or stopping the attack, and deflecting the attack. Lastly, there is the post-attack component which involves network forensics.

2.5.1 Prevent Secondary Victims

Individual Users: One of the best methods to prevent DDoS attacks is for the secondary victim systems to prevent themselves from participating in the attack. This requires a heightened awareness of security issues and prevention techniques from all Internet users. If attackers are unable to break into and make use of secondary victim systems, then the attackers will have no "DDoS attack network" from which to launch their DDoS attacks.

In order for secondary victims to not become infected with the DDoS agent software, users of these systems must continually monitor their own security. They must check to make sure that no agent programs have been installed on their systems and that they are not sending DDoS agent traffic into the network. The Internet is so de-centralized, and since there are so many different hardware and software platforms, it is quite difficult for typical users to implement the right protective measures. Typically this would include installing anti-virus and anti-Trojan software and keeping these up to date. Also, all software patches for

discovered vulnerabilities must be installed.

Since these tasks can be viewed as daunting for the average "web-surfer", recent work has proposed built-in mechanisms in the core hardware and software of computing systems that can provide defenses against malicious code insertion, for example through exploiting buffer overflow vulnerabilities [9]. This can significantly reduce the probability of a system being compromised as a secondary victim in setting up a DDoS attack network.

Network Service Providers: One strategy currently being discussed is for providers and network administrators to add dynamic pricing to their network usage, to encourage secondary victims to become more active in preventing themselves from becoming part of a DDoS attack. Dynamic pricing policies allow the network service providers to charge a cost per time unit depending on the availability of network resources; hence it regulates the arrival rate of calls of service to the network. This implies that network service requirements such as availability, reliability, security, bandwidth, congestion, routing, stability, delays, etc are maintained at an optimum level. These are the parameters that define the network QoS both within the network and at the edge access points where customer services are offered, leading to significant improvement in the network management [10]. If providers chose to charge differently for the use of different resources, they could charge for access to certain services within their networks. This would allow the providers to only allow legitimate customers on to their networks. This system would make it easier to prevent attackers from entering the network [11]. By altering the pricing of services, secondary victims who would be charged for accessing the Internet may become more conscious of the traffic they send into the network and hence may do a better job of policing themselves to verify that they are not participating in a DDoS attack.

2.5.2 Detect and Neutralize Handlers

One important method for stopping DDoS attacks is to detect and neutralize handlers. Since the agent-handler DDoS attack tools require the handler as an intermediary for the attacker to initiate attacks, finding and stopping the handlers

is a quick method to disrupt the DDoS attack network. This can possibly be done by studying the communication protocols and traffic patterns between handlers and clients or handlers and agents in order to identify network nodes that might be infected with a handler. Also, there are usually far fewer DDoS handlers deployed than there are agents, so neutralizing a few handlers can possibly render multiple agents useless, thus thwarting DDoS attacks.

2.5.3 Detect Potential Attacks

Egress Filtering: One method for detecting potential attacks is to use egress filtering. Egress filtering refers to the practice of scanning the packet headers of IP packets leaving a network (egress packets) and checking to see if they meet certain criteria. If the packets pass the criteria, they are routed outside of the sub-network from which they originated. If the filter criteria are not met, the packets will not be sent to the intended target. Since one of the features of DDoS attacks is spoofed IP addresses, there is a good probability that the spoofed source address of DDoS attack packets will not represent a valid source address of the specific sub-network. If the network administrator places a firewall or packet sniffer in the sub-network that filters out any traffic without an originating IP address from this subnet, many DDoS packets with spoofed IP source addresses will be discarded, and hence neutralized.

MIB Statistics: Another method currently being looked at to identify when a DDoS attack is occurring uses the Management Information Base (MIB) data from routers. The MIB data from a router includes parameters that indicate different packet and routing statistics. Current research has focused on identifying statistical patterns in different parameters during a DDoS attack [12]. It looks promising for possibly mapping ICMP, UDP, and TCP packet statistical abnormalities to specific DDoS attacks.

Accurate statistical models based on the MIB parameters from routers are still being studied to understand how accurately they can monitor DDoS attack traffic and predict when a DDoS attack is happening. Work in this area could provide important information and methods for identifying when a DDoS attack is starting

and how to filter or adjust the network to compensate for the attacking traffic.

2.5.4 Mitigate or Stop the Effects of DDoS Attacks

Load Balancing: For network providers, there are a number of techniques used to mitigate the effects of a DDoS attack. Providers can increase bandwidth on critical connections to prevent them from going down in the event of an attack. Replicating servers can help provide additional failsafe protection in the event some go down during a DDoS attack. Balancing the load to each server in a multiple-server architecture can improve both normal performance as well as mitigate the effect of a DDoS attack.

Throttling: One proposed method to prevent servers from going down is to use Max-min Fair server-centric router throttles [13]. This method sets up routers that access a server with logic to adjust (throttle) incoming traffic to levels that will be safe for the server to process. This will prevent flood damage to servers. Additionally, this method can be extended to throttle DDoS attacking traffic versus legitimate user traffic for better results. This method is still in the experimental stage, however similar techniques to throttling are being implemented by network operators. The difficulty with implementing throttling is that it is still hard to decipher legitimate traffic from malicious traffic. In the process of throttling, legitimate traffic may sometimes be dropped or delayed and malicious traffic may be allowed to pass to the servers.

Drop Requests: Another method is to simply drop requests when the load increases. This can be done by the router or the server. Alternatively, the requester may be induced to drop the request by making the requester system solve a hard puzzle that takes a lot of compute power or memory space, before continuing with the request. This causes the users of zombie systems to detect performance degradation, and could possibly stop their participation in sending DDoS attack traffic.

2.5.5 Deflect Attacks

Honeypots: Another area being researched is Honeypots. Honeypots are systems

that are set up with limited security to be an enticement for an attacker so that the attacker will attack the Honeypot and not the actual system. Honeypots typically have value not only in deflecting attacks from hitting the systems they are protecting, but also in serving as a means for gaining information about attackers by storing a record of their activity and learning what types of attacks and software tools the attacker is using. Current research discusses the use of honeypots that mimic all aspects of a legitimate network (such as web servers, mail servers, clients, etc.) in order to attract potential DDoS attackers [14]. The goal of this type of honeypot is to attract a DDoS attacker and get him to install either handler or agent code within the honeypot. This prevents some legitimate systems from getting compromised and allows the honeypot owner to track the handler or agent behavior and better understand how to defend against future DDoS installation attacks.

2.5.6 Post-Attack Forensics

Traffic Pattern Analysis: If traffic pattern data is stored during a DDoS attack, this data can be analyzed post-attack to look for specific characteristics within the attacking traffic. This characteristic data can be used for updating load balancing and throttling countermeasures to increase their efficiency and protection ability. Additionally, DDoS attack traffic patterns can help network administrators develop new filtering techniques for preventing DDoS attack traffic from entering or leaving their networks.

Packet Traceback: Another set of techniques assist in identifying the attackers using packet traces [15]. The concept of tracing is that Internet traffic could be traced back to the true source (rather than that of a potentially spoofed source IP address). This allows back tracing the attacker's traffic and possibly identifying the attacker. Additionally, when the attacker sends vastly different types of attacking traffic, this method assists in providing the victim system with information that might help develop filters to block the attack.

A model for developing a Network Traffic Tracking System that would identify and track user traffic throughout a network has been proposed. This type of system

has been identified as being very successful within a closed network environment, such as a corporate network where internal client systems can be fully managed by a central network administrator who can track individual end-user actions. This method begins to break down as the network becomes widely distributed. Traffic tracking on the Internet or large extranets is difficult to implement. Since different network administrators control different sections of the Internet, it would be difficult to determine who would be responsible for monitoring the traffic. Also, the loss of privacy on the Internet would meet with an unfavorable response from most network users.

Event Logs: Network administrators can keep logs of the DDoS attack information in order to do a forensic analysis and to assist law enforcement in the event the attacker does severe financial damage. Using both Honeypots as well as other network equipment such as firewalls, packet sniffers, and server logs, providers can store all the events that occurred during the setup and execution of the attack. This will allow the network administrators to discover what type of DDoS attack (or combination of attacks) was used.

Also based on the underlying strategies, we can categorize current DDoS detection and defense approaches into three categories: Proactive Mechanisms, Reactive Mechanisms and Post Attack Analysis [16].

Proactive defense mechanisms *(Keromytis, Misra and Rubenstein, 2002)* The motivation for these approaches 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, these approaches 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 *(Ioannidis and Bellovin, 2002)* These mechanisms typically deploy third-party Intrusion Detection Systems (IDS) to obtain attack information and take action based on this information.

	Detect & Neutralize the Handlers	Detect/ Prevent Secondary Victims	Detect/ Prevent Potential Attack	Mitigate/ Stop Attack	Deflect Attack	Post Attack Forensics
Proactive	✓	✓	×	✓	✓	×
Reactive	×	×	✓	✓	✓	✓
Post Attack	✓	×	✓	×	×	✓

Table 2.1: Allocation of mitigation methods into underlying strategies

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 can not 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 *(Song and Perrig, 2001)* 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 in some situations; 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.

2.6 Related Work

Our approach is related to reactive defense mechanism. Reactive mechanisms strive to alleviate the impact of an attack on the victim. To attain this goal they need to detect the attack and respond to it. The goal is to detect every attempted DDoS attack as early as possible and to have a low degree of false positives. Steps then can be taken to characterize the attack packets and provide this characterization to the response mechanism.

This reactive defense mechanism is classified into three mechanisms based on the attack detection strategies that deploy pattern detection, anomaly detection and third party detection. Mechanisms that deploy pattern detection store the signatures of known attacks in a database and monitor each communication for the presence of these patterns. The obvious drawback is that only known attacks can be detected, whereas new attacks or even slight variations of old attacks go unnoticed. On the other hand, known attacks are easily and reliably detected, and no false positives are encountered. A similar approach has been helpful in controlling computer viruses. Like in virus detection programs, signature databases must be regularly updated to account for new attacks.

Mechanisms that deploy anomaly detection have a model of normal system behavior, such as normal traffic dynamics or expected system performance. The current state of the system is periodically compared with the models to detect anomalies. Approaches presented in [17–19] provide examples of anomaly detection approaches. The advantage of anomaly detection over pattern detection is that previously unknown attacks can be discovered. The disadvantage is that anomaly detectors must trade on their ability to detect all attacks against their tendency to misidentify normal behavior as an attack.

Mechanisms that deploy third-party detection do not handle the detection process themselves, but rely on an external message that signals the occurrence of the attack and provides attack characterization. Examples of third-party detection are easily found among trace back mechanisms [20, 21].

Reactive defense mechanism is also classified into four more mechanisms based

on the attack response strategies. The goal of the attack response is to relieve the impact of the attack on the victim while imposing minimal collateral damage to legitimate clients. The four strategies are like agent identification, rate limiting, filtering and reconfiguration.

Agent identification mechanisms provide the victim with (somewhat accurate) information about the identity of the machines that are performing the attack. This information can be used by other approaches to alleviate the impact of the attack. Agent identification examples include numerous trace back techniques [20, 21].

Rate-limiting mechanisms impose a rate limit on a set of packets that have been characterized as malicious by the detection mechanism. It is a lenient response technique that is usually deployed when the detection mechanism has many false positives or cannot precisely characterize the attack stream. The disadvantage is that rate limiting will allow some attack traffic through, so extremely high-scale attacks might still be effective even if all traffic streams are rate limited. Examples of rate-limiting mechanisms are found in [18, 22].

Filtering mechanisms use the characterization provided by detection mechanisms to filter out the attack streams completely. Examples include dynamically deployed firewalls, and some commercial systems. Here the disadvantage is unless the characterization is very accurate, filtering mechanisms run the risk of accidentally denying service to legitimate traffic. Worse, clever attackers might leverage them as denial-of-service tools.

Reconfiguration mechanisms change the topology of the victim or the intermediate network to either add more resources to the victim or to isolate the attack machines. Examples include reconfigurable overlay networks [19], resource replication services, and attack isolation strategies [23]. In each and every case we have mentioned the demerit of mitigation mechanisms.

So we have proposed a probabilistic approach and a filtering method to mitigate the attack. As we are not depending upon the outside world to mitigate the attack we assume to solve this problem in place.

Chapter 3

Probabilistic Approach & HCF Method

Probabilistic Approach For Uncertainty Analysis

The Probabilistic Approach

Time-to-Live (TTL)

The Hop Count Filtering Method

Construction of HCF Table

Running States of HCF

Chapter 3

Probabilistic Approach & HCF Method

3.1 Probabilistic Approach For Uncertainty Analysis

Probabilistic approach is the most widely used technique for uncertainty analysis of mathematical models. There are a number of text books that describe the concepts and application of probabilistic analysis in detail. Tsokos (1972) presents excellent introductory material for probabilistic analysis, Johnson (1972) explains the multivariate random variables, and Papoulis (1991) presents an excellent description on probability and random variables from a mathematical view point. Additionally, Gardiner (1983) presents the applications of probabilistic analysis in modeling. This appendix attempts to merely summarize some basic information on probability and random variables, which can be found in more detail in the abovementioned texts.

In the probabilistic approach, uncertainties are characterized by the *probabilities* associated with *events* [24]. An event corresponds to any of the possible states a physical system can assume, or any of the possible predictions of a model describing the system. In the study of environmental pollution, the situation where the contaminant concentration exceeds a regulatory level can be an event. Similarly, in case of mathematical modeling, the situation where the model outputs fall in a certain range, is also an event. In short, any possible outcome of a given problem, such as the tossing of a coin, experimental measurement of contaminant concen-

tration, as and mathematical modeling of a physical system, is an event. The *probability* of an event can be interpreted in terms of the frequency of occurrence of that event. When a large number of *samples* or *experiments* are considered, the probability of an event is defined as the ratio of the number of times the event occurs to the total number of samples or experiments. A probability of 0 for an event means that the event will never occur, and a probability of 1 indicates that the event will always occur. Examples of experiments or samples include repeated coin tossing, a large number of independent measurements of contaminant concentration, or a large number of simulations using a mathematical model with randomly varying parameters.

The following examples illustrate the concept of probability based on a large number of samples: (a) if a coin is tossed, the probability of heads turning up is 0.5 and that of the tails is 0.5. This means that if the coin is tossed a large number of times, heads will turn up roughly half the time and tails half the time. (b) The statement that the probability that a pollutant concentration c lies between c_1 and c_2 equals p means the following: from a large number of independent measurements of the concentration c, under identical conditions, the number of times the value of c lies between c_1 and c_2 is roughly equal to the fraction p of the total number of samples.

3.2 The Probabilistic Approach

As knowing the number of packet being malicious is a very uncertain problem, we have applied the probability theory to calculate that easily.

Suppose the number of packets arriving at the server with a poisson's distribution λ. Suppose further that each packet which reaches at server being malicious with probability '*p*' or non-malicious with probability '*1-p*'.

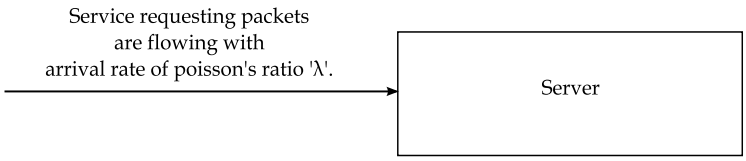

Figure 3.1: Model Configuration of Proposed System

So now the joint probability of 'n' packets among the total traffic are malicious and 'm' packets among the total traffic are non-malicious is as follows,

Let N_1 denotes the number of malicious packets.
N_2 denotes the number of non-malicious packets. Also let $N{=}N_1{+}N_2$ be the total number of traffic or packets which arrives at server.
Now conditioning on N gives,

$$P\{N_1 = n, N_2 = m\} = \sum_{i=0}^{\infty} P\{N_1 = n, N_2 = m \mid N = i\}\, P\{N = i\} \tag{1}$$

Because $P\{N_1 = n, N_2 = m \mid N = i\} = 0$, when $i \neq n{+}m$, the preceding equation yields that,

$$P\{N_1 = n, N_2 = m\} = P\{N_1 = n, N_2 = m \mid N = n + m\}\, e^{-\lambda} \frac{\lambda^{n+m}}{(n+m)!} \tag{2}$$

Also if it is clear that among that *'n+m'* packets each packet of being malicious has the probability of *'p'* and the conditional probability that *'n'* of them are malicious is nothing but just the *binomial probability* of *'n'* successes out of *'n+m'* trials. Therefore,

$$\begin{aligned} P\{N_1 = n, N_2 = m\} &= \left(\binom{m+n}{n} p^n (1-p)^m \right) e^{-\lambda} \frac{\lambda^{n+m}}{(n+m)!} \\ &= \left(\frac{(m+n)!}{m!n!} p^n (1-p)^m \right) e^{-\lambda} \frac{\lambda^{n+m}}{(n+m)!} \\ &= \frac{p^n (1-p)^m \left(e^{-\lambda p} e^{-\lambda(1-p)} \right) \lambda^m \lambda^n}{m!n!} \\ &= e^{-\lambda p} \frac{(\lambda p)^n}{n!} e^{-\lambda(1-p)} \frac{(\lambda(1-p))^m}{m!} \end{aligned} \tag{3}$$

Here equation (3) shows the probability of *'n'* packets being malicious and *'m'* packets being non-malicious. So by putting the value of probability to 1, by specifying the value of rate of arrival *'λ'* in *'number of packets per time unit'* and probability of error in a packet *'p'* we can get the value of *'n'* and *'m'*.

3.3 Time-to-Live (TTL)

The *time-to-live* (TTL) is the number of hops that a packet is permitted to travel before being discarded by a *router* [25].

A packet is the fundamental unit of information transport in all modern computer networks, and increasingly in other communications networks as well. A router is a *network layer* electronic device and/or software that connects at least two networks, such as two LANs (local area networks) or WANs (wide area networks), and forwards packets between them. A hop is the trip that a packet takes from one router to another as it traverses a network on the way to its destination.

The TTL is set in an eight binary digit field in the packet header by the sending host and is used to prevent packets from endlessly circulating on the Internet or other network. When forwarding an IP packet, routers are required to decrease the TTL by at least one. If a packet's TTL field reached zero, the router detecting it discards the packet and sends an ICMP (Internet control message protocol) message back to the originating host.

The *ping* and the *traceroute* utilities both make use of the TTL [25]. The latter intentionally sends packets with low TTL values so that they will be discarded by each successive router in the destination path. The time between sending a packet and receiving the ICMP message that it was discarded is used to calculate the travel time for each successive hop.

A specific TTL number can indicate the maximum range for a packet. For example, 0 restricts it to the same host, 1 to the same subnet, 32 to the same site, 64 to the same region and 128 to the same continent; 255 is unrestricted.

3.4 The Hop Count Filtering Method

To thwart DDoS attacks, researchers have taken two distinct approaches. The first approach improves the routing infrastructure, while the second approach enhances the resilience of Internet servers against attacks. The first approach performs either off-line analysis of flooding traffic traces or on-line filtering of spoofed IP packets inside routers. Offline IP traceback [21] attempts to estab-

lish procedures to track down flooding sources *after* occurrences of DDoS attacks. While it does help pinpoint locations of flooding sources, off-line IP traceback does not help sustain service availability during an attack. On-line filtering mechanisms rely on IP router enhancements [26] to detect abnormal traffic patterns and foil DDoS attacks. However, these solutions require not only router support, but also coordination among different routers and wide-spread deployment. The end-system approach protects Internet servers with sophisticated resource management to servers. This approach provides more accurate resource accounting, and fine-grained service isolation and differentiation.

End-system-based filtering that does not require router support is necessary to detect and discard spoofed traffic. We only utilize the information contained in the IP header for packet filtering. Although an attacker can forge any field in the IP header, he cannot falsify the number of hops an IP packet takes to reach its destination, which is solely determined by the Internet routing infrastructure. The hop-count information is indirectly reflected in the TTL field of the IP header, since each intermediate router decrements the TTL value by one before forwarding it to the next hop. The difference between the initial TTL (at the source) and the final TTL value (at the destination) is the hop-count between the source and the destination. By examining the TTL field of each arriving packet, the destination can infer its initial TTL value, and hence the hop-count from the source. Here we assume that attackers cannot sabotage routers to alter TTL values of IP packets that traverse them.

The rationale behind hop count filtering is that most spoofed IP packets, when arriving at victims, do not carry hop-count values that are consistent with legitimate IP packets from the sources that have been spoofed. *Hop-Count Filtering* (HCF) builds an accurate HCI (IP to hop-count) mapping table, while using a moderate amount of storage, by clustering address prefixes based on hop-count. To capture hop-count changes under dynamic network conditions, we also devise a "safe" update procedure for the HCI mapping table that prevents pollution by HCF aware attackers.

```
for each packet:
    Extract the final TTL T and IP address S;
    Infer the initial TTL T0;
    Compute the hop-count Hc=T-T0;
    Index S to get the stored hop-count Hs;
    if (Hc ≠ Hs)
        Packet is Spoofed;
    else
        Packet is Legitimate;
```

Table 3.1: Hop-Count Inspection Algorithm

Two running states, *alert* and *action*, within HCF use this mapping to inspect the IP header of each IP packet. Under normal condition, HCF resides in *alert* state, watching for abnormal TTL behaviors without discarding packets. Upon detection of an attack, HCF switches to *action* state, in which the HCF discards those IP packets with mismatching hop counts. Besides the HCI, several efficient mechanisms in [27] are available to detect DDoS attacks. Through analysis using network measurement data, we show that the HCF can recognize close to 90% of spoofed IP packets. Then, since our hop-count-based clustering significantly reduces the percentage of false positives, we can discard spoofed IP packets with little collateral damage.

3.4.1 Hop-Count Inspection

Central to HCF is the validation of the source IP address of each packet via hop-count inspection. In this section, we first discuss the hop-count computation, then present the inspection algorithm in detail.

TTL based Hop-Count Computation Since hop-count information is not directly stored in the IP header, one has to compute it based on the TTL field of the IP header. TTL is an 8-bit field in the IP header, originally introduced to specify the maximum lifetime of IP packets in the Internet. During transit, each intermediate router decrements the TTL value of an IP packet by one before forwarding it to the next-hop router. The final TTL value when a packet reaches its destination is therefore the initial TTL subtracted by the number of intermediate hops (or simply hop-count). The challenge in hop-count computation is that a

destination only sees the final TTL. It would have been simple if all operating systems (OSs) use the same initial TTL, but in practice, there is no consensus on the initial TTL value.

Fortunately, however, most modern OSs use only a few selected initial TTL values, 30, 32, 60, 64, 128, and 255, according to [28]. This set of initial TTL values cover most of the popular OSs, such as Microsoft Windows, Linux, variants of BSD, and many commercial Unix systems. We observe that these initial TTL values differ from each other by more than 30, except between 30 and 32, and between 60 and 64. Since it is generally believed that few Internet hosts are apart by more than 30 hops, which is also confirmed by our own observations, one can determine the initial TTL value of a packet by selecting the smallest initial value in the set that is larger than its final TTL. For example, if the final TTL value is 112, the initial TTL value is 128, the smallest of the two possible initial values, 128 and 255. To resolve ambiguities in the cases of 30 and 32, and 60 and 64, we will compute a hop-count value for each of the two possible initial TTL values, and accept the packet if either hop-count matches.

Inspection Algorithm Assuming that an accurate HCI mapping table is present, the above algorithm outlines the hop-count filtering mechanism uses to identify spoofed packets. The inspection algorithm extracts the source IP address and the final TTL value from each IP packet. The algorithm infers the initial TTL value and subtracts it from the final TTL value to obtain the hop-count. Then, the source IP address serves as the index into the table to retrieve the correct hop-count for this IP address. If the computed hop-count matches the stored hop-count, the packet has been "authenticated" otherwise, the packet is classified as spoofed. Note that a spoofed IP address may happen to have the same hop-count as the one from a zombie (flooding source) to the victim. In this case, HCF will not be able to identify every spoofed packet.

3.5 Construction of HCF Table

Building an accurate HCF table (i.e., HCI mapping table) is critical to detecting the maximum number of spoofed IP packets. In this section, we detail our approach to constructing HCF tables. Our objectives in building an HCF table are: (1) accurate HCI mapping, (2) up-to-date HCI mapping, and (3) moderate storage requirement. By clustering address prefixes based on hop-counts, we can build accurate HCI mapping tables and maximize HCF's effectiveness without storing the hop-count for each IP address. Moreover, a pollution-proof update procedure that captures legitimate hop-count changes while foiling attackers' attempt to pollute HCF tables is also described.

3.5.1 IP Address Aggregation

It is highly unlikely that an Internet server will receive legitimate requests from all live IP addresses in the Internet. Also, the entire IP address space is not fully utilized in the current Internet. By aggregating IP address, we can reduce the space requirement of HCI mapping significantly. More importantly, IP address aggregation covers those unseen IP addresses that are co-located with those IP addresses that are already in an HCF table. Grouping hosts according to the first 24 bits of IP addresses is a common aggregation method. However, hosts whose network prefixes are longer than 24 bits, may reside in different physical networks in spite of having the same first 24 bits. Thus, these hosts are not necessarily co-located and have identical hop-counts. To obtain an accurate HCI mapping, we must refine the 24-bit aggregation. Instead of simply aggregating into 24-bit address prefixes, we further divide IP addresses within each 24-bit prefix into smaller clusters based on hop-counts. To understand whether this refined clustering improves HCF over the simple 24-bit aggregation, we compare the filtering accuracies of HCF tables under both aggregations - the simple 24-bit aggregation (without hop-count clustering) and the 24-bit aggregation with hop-count clustering.

Aggregation into 24 bit Address Prefixes: For each web server, we build an HCF table by grouping its IP addresses according to the first 24 bits. We use the minimum hop-count of all IP addresses inside a 24-bit network address as the hop-count of the network. After the table is constructed, each IP address is converted into a 24-bit address prefix, and the actual hop-count of the IP address is compared to the one stored in the aggregate HCF table. Since 24-bit aggregation does not preserve the correct hop-counts for all IP addresses, we examine the performance of three types of filters: "Strict Filtering," "+1 Filtering," and "+2 Filtering." "Strict Filtering" drops packets whose hop-counts do not match those stored in the table. "+1 Filtering" drops packets whose hop-counts differ by greater than 1 compared to those in the table, and "+2 Filtering" drops packets whose hop-counts differ by greater than two.

24-bit aggregation is straightforward to implement and can offer fast lookup with an efficient implementation. Assuming a one-byte entry per network prefix for hop-count, the storage requirement is 224 bytes or 16 MB. The memory requirement is modest compared to contemporary servers which are typically equipped with multi-gigabytes of memory. Under this setup, the lookup operation consists of computing a 24-bit address prefix from the source IP address in each packet and indexing it into the HCF table to find the right hop-count value. For systems with limited memory, the aggregation table can be implemented as a much smaller hash-table. While 24-bit aggregation may not be the most accurate, at present it is a good and deployable solution.

3.5.2 Pollution proof Initialization and Update

To populate the HCF table initially, an Internet server should collect traces of its clients that contain both IP addresses and the corresponding TTL values. The initial collection period should be commensurate with the amount of traffic the server is receiving. For a very busy site, a collection period of a few days could be sufficient, while for a lightly-loaded site, a few weeks might be more appropriate.

Keeping the HCI mapping up-to-date is necessary for our filter to work in the Internet where hop-counts may change. The hop-count, or distance from a client

to a server can change as a result of relocation of networks, routing instability, or temporary network failures. Some of these events are transient, but changes in hop-count due to permanent events need to be captured.

While adding new HCI mapping entries or capturing legitimate hop-count changes, we must foil attackers' attempt to slowly pollute HCF tables by dropping spoofed packets. One way to ensure that only legitimate packets are used during initialization and dynamic update is through TCP connection establishment, an HCF table should be updated only by those TCP connections in the established state. The three-way TCP handshake for connection setup requires the active-open party to send an ACK (the last packet in the three-way handshake) to acknowledge the passive party's initial sequence number. The host that sends the SYN packet with a spoofed IP address will not receive the server's SYN/ACK packet and thus cannot complete the three-way handshake. Using packets from established TCP connections ensures that an attacker cannot slowly pollute an HCF table by spoofing source IP addresses.

While our dynamic update provides safety, it may be too expensive to inspect and update an HCF table with each newly-established TCP connection, since our update function is on the critical path of TCP processing. We provide a user-configurable parameter to adjust the frequency of update. The simplest solution would be to maintain a counter p that records the number of established TCP connections since the last reset of p. We will update the HCF table using packets belonging to every k-th TCP connection and reset p to zero after the update. p can also be a function of system load and hence, updates are made more frequently when the system is lightly-loaded.

3.6 Running States of HCF

Since HCF causes delay in the critical path of packet processing, it should not be active at all time. We therefore introduce two running states inside HCF: the *alert* state to detect the presence of spoofed packets and the *action* state to discard spoofed packets. By default, HCF stays in alert state and monitors the

trend of hop-count changes without discarding packets. Upon detection of a flux of spoofed packets, HCF switches to action state to examine each packet and discards spoofed IP packets. In this section, we discuss the details of each state and show that having two states can better protect servers against different forms of DDoS attacks.

In *alert* state:
for each **sampled** packet p:
spoof=HCI(p);
t=Average(*spoof*);
if(*spoof*)
if($t > T_1$)
Switch2Action();
Accept(p);
for the k-th TCP control block *tcb*:
UpdateTable(*tcb*);

In *action* state:
for each packet p:
spoof=HCI(p);
t=Average(*spoof*);
if(*spoof*)
Drop(p);
else
Accept(p);
if($t \leq T_2$)
Switch2Alert();

Table 3.2: Operations of States of Hop-Count Filtering Method

3.6.1 Task in Two States

HCF performs the following tasks: sample incoming packets for hop-count inspection, calculate the spoofed packet counter, and update the HCI mapping table in case of legitimate hop-count changes. Packets are sampled at exponentially-distributed intervals with mean m in either time or the number of packets. The exponential distribution can be precomputed and made into a lookup table for fast on-line access. For each sampled packet, HCI() returns a binary number *spoof*, depending on whether the packet is judged as spoofed or not. This is then used by Average() to compute an average spoof counter t per unit time. When t is greater

than a threshold *T1*, the HCF enters the action state. The HCF in alert state will also update the HCF table using the TCP control block of every k-th established TCP connection.

The HCF in *action* state performs per-packet hop-count inspection and discards spoofed packets, if any. The HCF in action state performs a similar set of tasks as in alert state. The main differences are that HCF must examine every packet (instead of sampling only a subset of packets) and discards spoofed packets. The HCF stays in action state as long as spoofed IP packets are detected. When the on going spoofing ceases, the HCF switches back to alert state. This is accomplished by checking the spoof counter t against another threshold *T2*, which should be smaller than *T1* for better stability. HCF should not alternate between alert and action states when t fluctuates around *T1*. Making the second threshold *T2*$<$*T1* avoids this instability.

Chapter 4

Proposed System & Simulation Results

Introduction

Proposed System

Detailed Structure Overview

Proposed Algorithm

Simulation Results

Chapter 4

Proposed System & Simulation Results

4.1 Introduction

It is discussed earlier that to thwart DDoS attackers the researchers have taken two approaches. The first approach improves the routing infrastructures whereas the second approach protects Internet servers with sophisticated resource management to the servers. This end system approach provides more accurate resource accounting, and fine-grained service isolation and differentiation [29], for example, to shield interactive video traffic from FTP traffic. However, without a mechanism to detect spoofed traffic, spoofed packets will share the same resource principals and code paths as legitimate requests. While a resource manager can confine the scope of damage to the particular service under attack, it cannot sustain the availability of that service. In stark contrast, the server's ability to filter most, if not all, spoofed IP packets can help sustain service availability even under DDoS attacks. Since filtering spoofed IP packets is orthogonal to resource management, it can also be used in conjunction with advanced resource-management schemes.

As discussed earlier, end-system-based filtering that does not require router support is necessary to detect and discard spoofed traffic. We only utilize the information contained in the IP header for packet filtering. Although an attacker can forge any field in the IP header, he cannot falsify the number of hops an IP packet takes to reach its destination, which is solely determined by the Internet routing infrastructure. The hop-count information is indirectly reflected in the

TTL field of the IP header, since each intermediate router decrements the TTL value by one before forwarding it to the next hop. The difference between the initial TTL (at the source) and the final TTL value (at the destination) is the hop-count between the source and the destination. By examining the TTL field of each arriving packet, the destination can infer its initial TTL value, and hence the hop-count from the source. Here we assume that attackers cannot sabotage routers to alter TTL values of IP packets that traverse them.

The rationale behind hop-count filtering is that most spoofed IP packets, when arriving at victims, do not carry hop-count values that are consistent with legitimate IP packets from the sources that have been spoofed. Hop-Count Filtering (HCF) builds an accurate HCI (IP to hop-count) mapping table, while using a moderate amount of storage, by clustering address prefixes based on hop-count. To capture hop-count changes under dynamic network conditions, we also devise a "safe" update procedure for the HCI mapping table that prevents pollution by HCF aware attackers.

Two running states, alert and action, within HCF use this mapping to inspect the IP header of each IP packet. Under normal condition, HCF resides in alert state, watching for abnormal TTL behaviors without discarding packets. Upon detection of an attack, HCF switches to action state, in which the HCF discards those IP packets with mismatching hop-counts. Besides the HCI inspection, several efficient mechanisms [27] are available to detect DDoS attacks. Through analysis using network measurement data, we show that the HCF can recognize close to 90% of spoofed IP packets. Then, since our hop-count-based clustering significantly reduces the percentage of false positives, we can discard spoofed IP packets with little collateral damage. To ensure that the filtering mechanism itself withstands attacks, our design is light-weight and requires only a moderate amount of storage. We implement a test module of HCF in the Linux kernel, at the network layer as the first step of IP-packet processing. Then, we evaluate its benefits with real experiments and show that HCF is indeed effective in countering IP spoofing while producing significant resource savings.

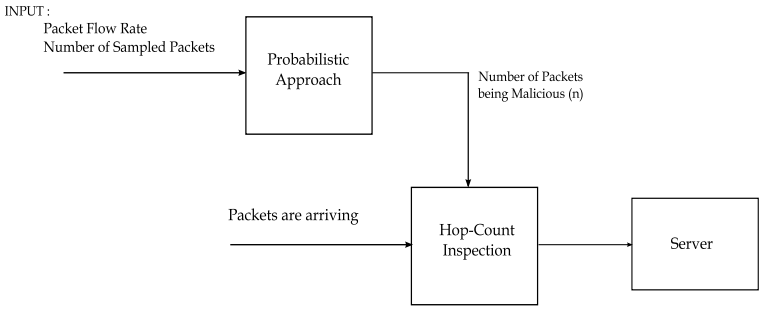

Figure 4.1: The Detailed design of Proposed System

4.2 Proposed System

There already exist commercial solutions that block the propagation of DDoS traffic with router support. However, the main difference between our scheme and the existing approaches is that our approach & HCF are end-system-based mechanisms that do not require any network support. So we have compared the results of both the schemes. Our proposed approach saves the resource as the computational time needed by the system to mitigate in large extend, as the HCF method works in two phases.

In our approach we are calculating the number of malicious packets from a given number of packets using a probabilistic approach. We first sample the number of packets on the basis of arrival at the server per a time unit. By taking that number of packets, the average arrival rate of the packets and the error probability of a packet we calculate the number of packets being malicious. Then we apply the simple HCI (Hop-Count Inspection) Algorithm (discussed in Chapter 3) to filter out first that many number of packet how much was found out using the probabilistic approach. When we will reach at the exact number of packets, we simply release all the rest packets towards the server assuming that these are not malicious. It may also happen that we may loose some malicious packets being undetected but we save the computational time in a great extent than the actual HCF algorithm.

4.3 Detailed Structure Overview

In our proposed view we are just showing the structure for only one server. This can be implemented on individual server in this manner. Here in our structure we are having one victim server. In our structure we are having two important sections, i.e. the section which will implement the probabilistic approach and the section which will implement the Hop-Count Inspection (HCI) method. When packets are arrived, those will be sampled according to the willing of the victim. Then number of packet is taken for the sampling, their average arrival rate, and their probability of error in a packet will be taken into consideration for the calculation of number of packet being malicious by the probabilistic approach section. Then the probabilistic approach section will result out the value. Based on that value the Hop-Count Inspection section will filter out the packets which will be found malicious and dropped from being served by the server. Simultaneously a counter checks whether the number of found malicious packet has reached the number found by the probabilistic approach. If it will touch the limit very soon then all the packets remaining in the sampled packets will be released to the server without checking, whether those are malicious or not, assuming those are non-malicious in nature. By applying this approach we need not to check all the packets for the maliciousness. Hence we are saving the computational time with some extent and also mitigating the DDoS attack.

4.4 Proposed Algorithm

This algorithm is proposed to implement our approach at attacked end. Here initial input to the algorithm are the rate of arrival of the packets at the server, the error probability of each packet arrived at the server, and the number of packet was taken as a sample. Then these information were used to calculate the number of packets being malicious. That value is identified as *'n'*. This value is input to the HCI section. In HCI section the packet will be checked whether the packet is *'spoofed'* or not using the discussed TTL value of each packet. Depending on the result of HCI, the packet will either be allowed or dropped by the algorithm. If

```
For given 'λ','p','m+n':
Calculate 'n' such that P(N1=n,N2=m)=1;
integer count=0;
for each packet 'i':
        if(count≠n)
          status=HCI(i);
          if status is 'spoofed'
            count++;
            drop the packet;
          else
            allow the packet;
          end if;
        end if;
end for;
```

Table 4.1: Proposed algorithm for mitigation of DDoS Attack

the packet will be found *'spoofed'* then the counter value will be incremented and the packet will be dropped. The counter value is maintained because the counter will run up to *'n'* so that the first spoofed *'n'* packets will be dropped and rest of the sampled packet will be considered as non-malicious and allowed to the server. Each time a spoofed packet will be identified the *count* value will be incremented. Hence by using this approach we are saving the computational time which was not saved in classical HCF mechanism. In classical HCF method each and every packet were checked for the maliciousness and using two sections of execution and using threshold value.

4.5 Simulation Results

Some portion of the analysis of this approach was simulated using ns2 and some portion using matlab. We first analyze the approach in ns2 by taking some nodes. Some of them were attackers and it was intending to attack on one victim. So we forward packets from those malicious nodes by spoofing the source address. And others were pretending them as non-malicious node and they were sending packets using their own source address. By applying the probabilistic approach and the HCI method at the victim side we caught some packets as malicious and some as non-malicious. So from there we have collected 16 set of data and maintained into

an array. We have collected the dataset about the HCF method and use them for the analysis of the performance. Then we have just compared the results of our approach and the results of classical HCF method. Our approach took less time than the HCF method.

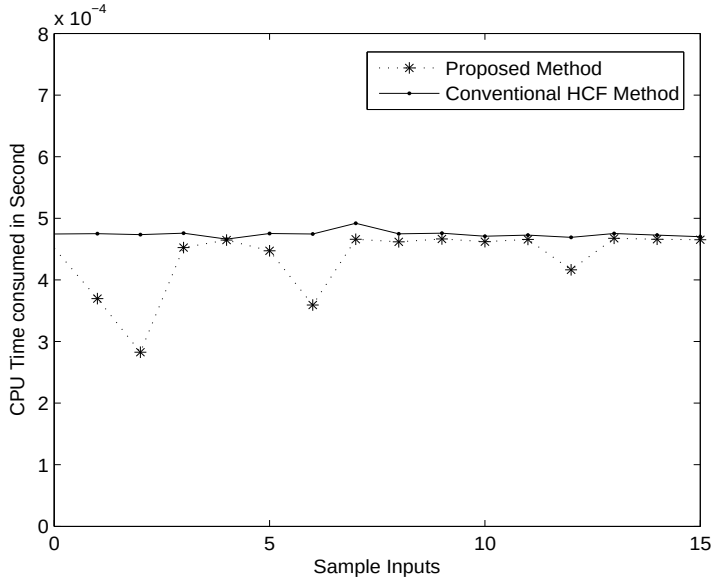

Figure 4.2: Comparison between time taken by Proposed Algorithm and HCF method (Packet flow rate=8000 packets/second)

We have collected the dataset of different flow rate of packets. And in our simulation we have used that flow rate to validate our approach. For our simulation we have taken the flow rate of packet as 8000 packets/second. From the above we can see the resource as time is saved by our approach. It takes less time than the HCF method because of the complexity of the execution of HCF method, which takes two steps in execution (*alert* and *action* states). Here we compared the sixteen data we got from our result.

In [5] it was given that the 90% of DDoS traffic was caught using the HCF method. Here also averagely 85% to 90% of DDoS traffic was caught. And that is because of that we are not using any queue for keeping the packets when they are reaching at the server side. We are using all those packets as they are coming to the server. That's why sometime the packet detection is less than 100% then it will come to 100% and again it will go less than 100%. This cycle will continue. This

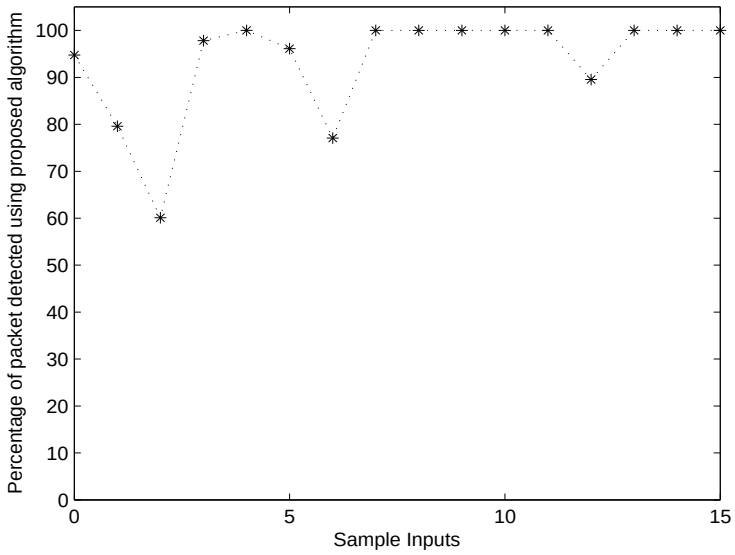

Figure 4.3: Percentage of packets detected using Proposed Algorithm(Packet flow rate=8000 packets/second)

is the future enhancement to our proposed model. Very less amount of packets got undetected using this approach as we are using the probabilistic approach to solve that whereas the HCF method used the threshold for considering the packets to be malicious.

From the following dataset we have compared our result with the existing one and simulated the same to show our proposed approach's performance. Figure 4.2 shows the comparison between the time taken by the HCF method and the proposed method. Figure 4.3 shows the packets being detected by the proposed method, and it shows averagely our approach shows about to 90% of DDoS packets are being detected.

We have collected so many dataset of HCF Method. Some of them are as follows,

Sample	HCF Method ($\times 10^{-3}$ sec)
1	0.1783
2	0.1767
3	0.2724
4	0.1761
5	0.1763
6	0.1764
7	0.1782
8	0.1832
9	0.1748
10	0.1770
11	0.1752
12	0.1758
13	0.4845
14	0.2753
15	0.2619
16	0.2623

Table 4.2: Packet flow rate(λ)=3000 packets/second

Sample	HCF Method ($\times 10^{-3}$ sec)
1	0.3539
2	0.4458
3	0.3562
4	0.3523
5	0.3581
6	0.3529
7	0.3559
8	0.4187
9	0.3482
10	0.7400
11	0.5709
12	0.5440
13	0.5433
14	0.4057
15	0.3519
16	0.3581

Table 4.3: Packet flow rate(λ)=6000 packets/second

Sample	HCF Method ($\times 10^{-3}$ sec)
1	0.5946
2	0.9060
3	0.9490
4	0.9608
5	0.9232
6	0.9656
7	0.9735
8	0.9619
9	0.9364
10	0.8407
11	0.5996
12	0.5822
13	0.7272
14	0.5948
15	0.5839
16	0.5840

Table 4.4: Packet flow rate(λ)=10,000 packets/second

Chapter 5

Conclusion and Future Work

Conclusion

Future Enhancement

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis shows that Distributed Denial-of-Service (DDoS) attack is very harmful attack. Many organisations have faced a lot of problems to manage. This attack will harm your system within no time or without any prior knowledge. As this attack is very much harmful, so many defense mechanisms were developed to mitigate the attack to make the systems free form harm. The attack strategies are also of different types. They may need the help of network services or may not need to mitigate the attack. Till now so many network supported solution were proposed but proposals without the support of network were very rare. That had motivated us to work upon this. Previously HCF method was there to mitigate the attack.

From Chapter 2 we got the idea that how much harmful this attack is and how much uncertain this attack is. As this attack is very much uncertain we proposed one probabilistic approach to work this out. Then that probabilistic approach shows us the way to solve the problem. Then simple HCI (Hop-Count Inspection) method made it possible to solve it out. As here we have compared the result of our approach with the result of HCF method, we have collected the dataset required to solve the problem. The simulation results shows that our approach is taking less amount of time than the HCF method and also it could detect about to 90% of the DDoS packets from all the packets.

5.2 Future Enhancement

Chapter 4 discusses the total system design for the mitigation of attack using our probabilistic approach and the simple HCI method. The packets are flowing towards the server but for the better sampling we should use one queue there to maintain the flow of packets through the server. As our probabilistic approach takes very little amount of time to calculate the number of packet being malicious, we need to maintain the sampled packet in that queue for that much amount of time so that we should not loss any packet and a better sampling can be done. By doing that we can have a better result than existing result as we can exactly filter up to 90% to 100% of DDoS packets.

This thesis shows another enhancement in the related field. This opens a way to solve the DDoS attack problems without the help of network support. Many researches can be done to enhance the proposed work to perform better.

Bibliography

[1] Behrouz A. Forouzan. Cryptography and network security. *Tata McGraw Hill Publication*, 4th Edition,2008.

[2] S. Specht and R. Lee. Taxonomies of distributed denial of service networks, attacks, tools, and countermeasures. *Technical Report CE-L2003-03*, 164, May 2003.

[3] http://www.w3.org/security/faq/wwwsf6.html.

[4] Graham Wheeler. Denial-of-service:courting disaster. *Technical Report,CEQURUX Technologies.*

[5] Haining Wang , Cheng Jin and Kang G. Shin. Hop-count filtering: An effective defense against spoofed traffic. *Technical Report,The Pennsylvania State University*, 2003.

[6] Christos Douligeris and Aikaterini Mitrokotsa. Ddos attacks and defense mechanisms: A classifications. *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology*, pages 190–193, December 2003.

[7] Valon Sejdini Li Xiaoming and Hasan Chowdhury. Denial of service (dos) attack with udp flood. *School of Computer Science, University of Windsor, Windsor, Ontario, Canada.*

[8] Juniper Networks. firewall/vpn feature brief - a documental survey. June 2006.

[9] Patrick McGregor Ruby Lee, David Karig and Zhijie Shi. Enlisting hardware architecture to thwart malicious code injection. *Proceedings of the International Conference on Security in Pervasive Computing (SPC-2003)*, March 2003.

[10] http://www.wired.com/news/business/0,1367,34221,00.html. *Yahoo on Trail of Site Hackers*, February 2000.

[11] Ceilyn Boyd John Zao David Mankins, Rajesh Krishnan and Michael Frentz. Mitigating distributed denial of service attacks with dynamic resource pricing. *Proceedings of 17th Annual Conference on Computer Security Applications , 2001. ACSAC 2001*, pages 411–421, 2001.

[12] Xinzhou Qin Wenke Lee Ravi K. Prasanth B. Ravichandran Joao B. D. Cabrera, Lundy Lewis and Ramon K. Mehra. Proactive detection of distributed denial of service attacks using mib traffic variables - a feasibility study. *Integrated Network Management Proceedings*, pages 609–622, 2001.

[13] John C. S. Lui David K. Yau and Feng Liang. Defending against distributed denial of service attacks with max-min fair server-centric router throttles. *2002 Tenth IEEE International Workshop on Quality of Service*, pages 35–44, 2002.

[14] Nathalie Weiler. Honeypots for distributed denial of service. *Proceedings of Eleventh IEEE International Workshops*, pages 109–114, 2002.

[15] Howard F. Lipson. Tracking and tracing cyber attacks: Technical challenges and global policy issues. *CERT Coordination Center*, November 2002.

[16] Guangsen Zhang and Manish Parashar. Cooperative defence against ddos attacks. *Department of Electrical and Computer Engineering, The State University of New Jersey*, February 2006.

[17] S. Early J. Yan and R. Anderson. The xenoservice - a distributed defeat for distributed denial of service. *Proceedings of ISW 2000*, October 2000.

[18] S. Floyd V. Paxson R. Mahajan, S. Bellovin and S. Shenker. Controlling high bandwidth aggregates in the network. *ACM Computer Communications Review*, 32(3), July 2002.

[19] Information Sciences Institute. Dynabone. http://www.isi.edu/dynabone/.

[20] M. Leech S. Bellovin and T. Taylor. Icmp traceback messages. internet draft, work in progress. October 2001.

[21] A. Karlin S. Savage, D. Wetherall and T. Anderson. Practical network support for ip traceback. *In Proceedings of ACM SIGCOMM 2000*, August 2000.

[22] T. M. Gil and M. Poletto. Multops: a data-structure for bandwidth attack detection. *In Proceedings of 10th Usenix Security Symposium*, August 2001.

[23] BBN Technologies. http://www.bbn.com/infosec/apod.html. *Applications that participate in their own defense.*

[24] http://www.ccl.rutgers.edu/ ssi/thesis/thesis-node51.html.

[25] http://searchnetworking.techtarget.com/sdefinition/0,,sid7_gci214184,00.html.

[26] V. Misra A. D. Keromytis and D. Rubenstein. *SoS: Secure overlay services.* In Proceedings of ACM SIGCOMM' 2002, August 2002.

[27] M. Fullmer and S. Romig. The osu flow-tools package and cisco netflow logs. *In Proceedings of USENIX LISA'2000, New Orleans, LA*, December 2000.

[28] Swiss Academy and Research Network. http://www.map.meteoswiss.ch/map-doc/ftp-probleme.htm.

[29] P. Druschel G. Banga and J. Mogul. Resource containers: A new facility for resource management in server systems. *In Proceedings of USENIX OSDI'99, New Orleans, LA*, February 1999.

Dissemination of Work

1. Biswa Ranjan Swain & Bibhudatta Sahoo, "Mitigating DDoS Attack & Saving Computational Time using a Probabilistic Approach & HCF Method ", *IEEE International Advance Computing Conference*, pp 1170-1172, 6-7 March 2009, Patiala, Punjab, India.