PERFORMANCE ANALYSIS OF DEFENSE MECHANISMS AGAINST UDP FLOOD ATTACKS

KIATTIKUL TRESEANGRAT

2014

ABSTRACT

A Distributed Denial of Service (DDoS) attack remains one of the most common and devastating security threats to the Internet world. The main purpose of an attack is to disable the use of services on the Internet or the victim network by sending a large number of IP packets to the targeted system. Since no single solution for a DDoS attack has been found, these attacks have managed to prevail on the Internet for a decade. Therefore, it is necessary and important to evaluate such an attack in a real testbed environment to find the most suitable defense mechanism.

In this thesis, the different types of DDoS attacks are discussed followed by a focus on UDP flood attacks. Tests were conducted and new results obtained on the impact of a UDP flood attack on computers using the latest versions of Windows and Linux platforms, e.g., Windows Server 2012, Windows 8, and Linux Ubuntu 13. This research also produced new evaluation results on various defense mechanisms such as Network Load Balancing, Software Firewall, Access Control Lists, Threshold Limit, Hybrid Method, and IP Verify.

Unlike simulation studies, this project lays down the steps involved in implementing the attack in a real testbed environment. In this study, the victim network is based on an Intranet network environment that provides several services (e.g., a web service and file transfer service) to legitimate clients. An attacker in the testbed, on the other hand, will launch the attack from outside the local subnet. Several metrics such as round-trip time, user throughput, packet loss, and CPU utilization of the victim computer were gathered in order to investigate the impact of an attack.

The findings of this study concluded that Linux Ubuntu 13 withstood UDP flood attacks better than Windows Server 2012 while the Hybrid Method and Threshold Limit were the most effective defenses against UDP flood attacks for both Windows and Linux platforms. Both defenses significantly increased throughputs, and reduced the RTT, packet loss, and CPU utilization of a victim computer. On the other hand, the Software Firewall was the least effective defense in all studies.

ACKNOWLEDGEMENTS

First of all I would like to acknowledge my principal supervisor, Dr Samad Kolahi, and my associate supervisor, Bahman Sarrafpour, for giving me the opportunity to carry out this research project and for giving me their valuable time and assistance from the beginning through to the completion of this study. I would also like to acknowledge my postgraduate program director, Dr Chandimal Jayawardena. This thesis would not have been successfully completed without their guidance.

Secondly, I would like to give special thanks to my previous lecturers in Department of Computing at Unitec and at Mahanakorn University of Technology who have provided me with valuable knowledge and skills during the course. Their knowledge and experience have empowered me to reach the level required to complete this study.

Lastly, I would like to express my appreciation and thanks to my family members who have facilitated and provided me with this opportunity and supported me during this thesis as well as the other requirements to fulfill my master’s degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACL	Access Control List
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DETER	Defense Technology Experimental Research
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DoS	Denial of Service
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IPAD	Internet Protocol Address Database
IP	Internet Protocol
IPS	Intrusion Prevention System
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IRC	Internet Relay Chat
ISP	Internet Service Provider
LAN	Local Area Network
MAC Address	Media Access Control Address
Mbps	Megabits per Second
MS	Millisecond
NFS	Network File System
NIC	Network Interface Card
NLB	Network Load Balancing
NTP	Network Time Protocol
OS	Operating System
PC	Personal Computer
PPS	Packets per Second
QoS	Quality of Service
RA	Router Advertisement
RAM	Random Access Memory
RPF	Reverse Path Forwarding
RTT	Round Trip Time
SLA	Service-Level Agreement
TCP	Transmission Control Protocol
TCP/IP	Transport Communication Protocol and Internet Protocol
TFN	Tribe Flood Network
UDP	Connectionless User Datagram Protocol
VoIP	Voice Over Internet Protocol

CHAPTER 1

INTRODUCTION

1.1 Internet Security

A revolution occurred in the world of computers and communication with the advent of the Internet. Nowadays, such a technology has become increasingly important to current society; it has changed our way of communication, business modes, and made information publicly accessible quickly and easily manner, and within easy reach. Many organizations use such technology to provide various services to their customers, for example, online banking, shopping online, and video conferencing.

However, along with the advantages of the Internet, there are also some disadvantages. There is no absolute security in the Internet world, and hackers can use the Internet to launch many different types of attacks on a targeted network, one of which is known as a Distributed Denial of Service attack (DDoS attack).

A DDoS attack is one of the most common and major threats to the Internet in which the goal of the attacker is to consume resources of the victim, usually by sending a high volume of seemingly legitimate traffic requesting services from the victim. As a result, it creates network congestion on the targeted computer, thus disrupting its normal Internet operation (Gupta, Joshi, & Misra, 2010).

1.2 Event of DDoS Attacks

The first reported DDoS attack appeared in late 1999 against the University of Minnesota, Canada. The attack, which was launched by 227 Zombies (compromised computers) shut down the university's network for more than two days (Garber, 2000). DDoS attacks received further attention in February 2000 when a hacker using a DDoS attack crippled several major Internet companies including *Amazon.com, Yahoo!, eBay*, and *CNN Interactive*, significantly slowing them down or rendering their websites inaccessible (Singh & Juneja, 2010).

Experts estimated that during the three hours *Yahoo!* was down, the company's loss of advertising revenue and e-commerce was approximately $500,000. The resulting down time for *Amazon.com* cost them an estimated $600,000. During the attack, the number of

CNN.com's users dropped to five per cent of the normal volume, while *Buy.com* went from 100% availability to 9.4% (Hutt, Hoyt, & Bosworth, 2002).

Despite their impact, there are no effective strategies to deal with these attacks which have been in existence for almost a decade. According to a worldwide infrastructure security report in 2012, half of the survey respondents (130 respondents in total) indicated they have experienced DDoS attacks against their infrastructures (e.g., routers, firewalls, and load balancers), and one-quarter encountered DDoS attacks against services used by their customers and partners during the survey period. The result also showed that about 10% of the attacks were launched by malicious insiders (Anstee, Bussiere, & Sockrider, 2012).

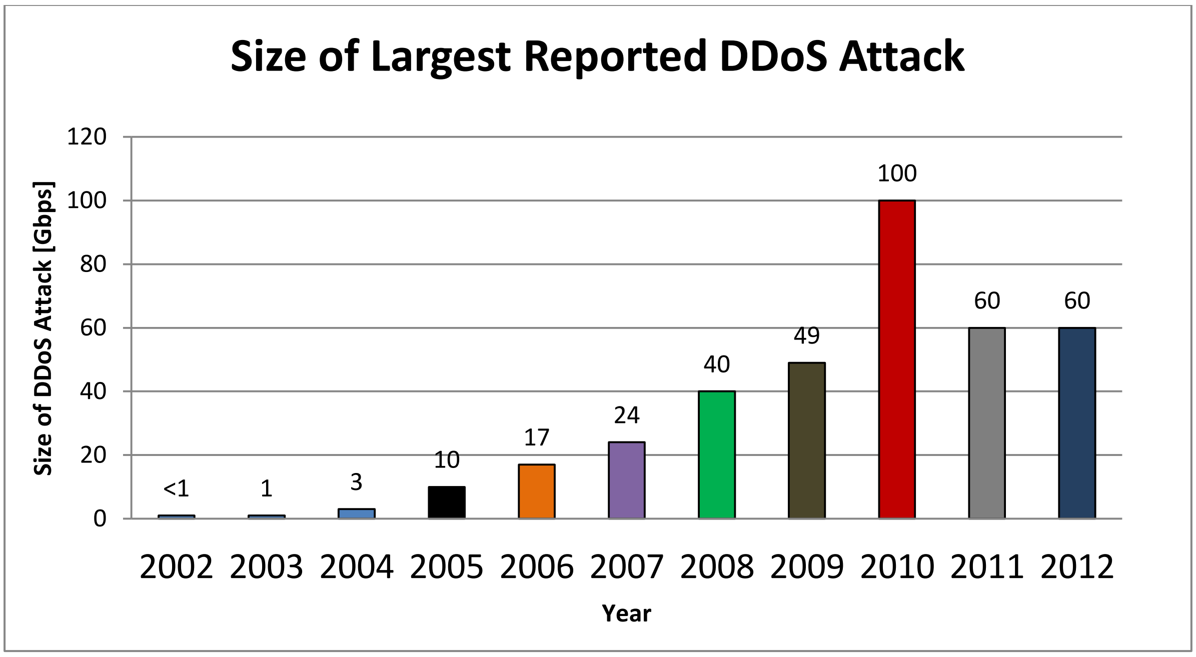


Figure 1.1: Size of Largest Reported DDoS Attack in Gbps from 2002 to 2012 (Anstee et al., 2012)

Figure 1.1 illustrates the size of DDoS attacks from 2002 to 2012 (Anstee et al., 2012). The result shows that the size of the attack at the beginning was only 500Mbps. This is no surprise as there were not many DDoS attack tools available on the Internet and the speed of the Internet in early 2000 was limited.

However, in 2012, the number of attacks significantly increased to 60Gbps, which was over 120 times higher than the figure in 2002. The most noticeable feature of this graph is that the largest attack reported was 100Gbps (in 2010). "This was a very significant volume of traffic and was more bandwidth than some Internet operators had, let alone their customers. This can be concluded that the attackers are shifting to more advanced threat approaches" (Anstee et al., 2012).

Although the number of DDoS attacks is growing every year, only 37% of the respondents (130 respondents in total) had developed an increased awareness of the DDoS threat in their organization, whereas, 63% retained the same level of awareness. Moreover, the report reveals that more than 10% of the respondents did not have DDoS mitigation capabilities in their networks (Anstee et al., 2012).

1.3 Impact of DDoS on Business

With the current trends toward data center consolidation and cloud computing, it is essential to keep up with developments relating to traffic monitoring techniques, DDoS defenses, and other points of interest regarding data centers. This section discusses the business impact caused by DDoS attacks in data centers around the world from October 2011 through to the end of September 2012 (Anstee et al., 2012).

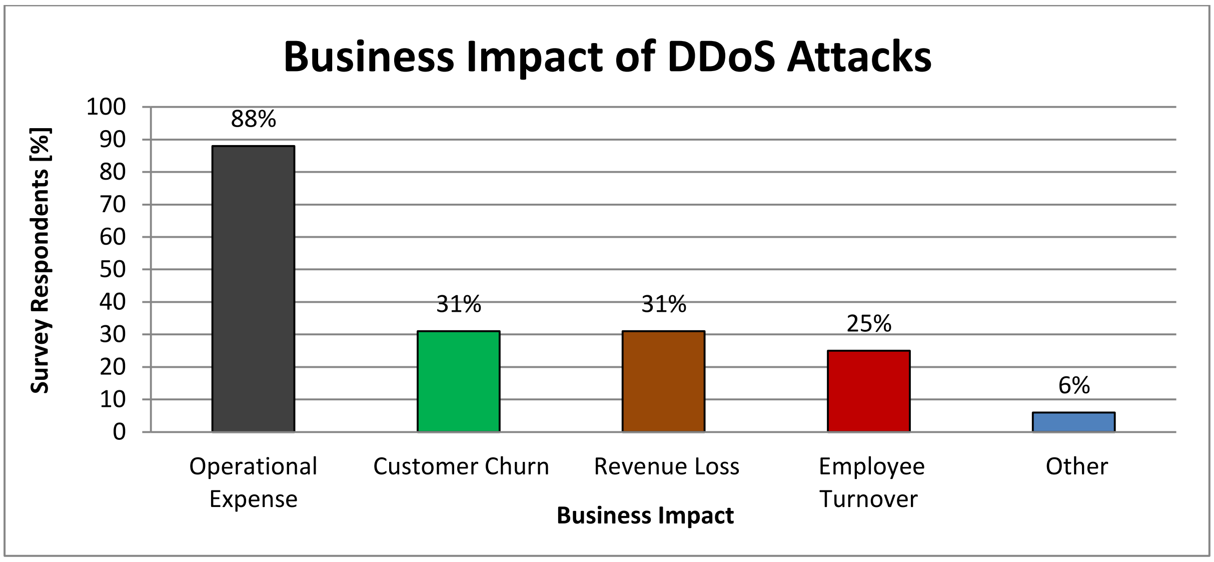


Figure1.2: Business Impact of DDoS Attacks in Data Center (Anstee et al., 2012)

Figure 1.2 illustrates the impact of DDoS attacks on the business in data centers. The survey reveals that most data center operators (about 90% of server respondents) reported operational expenses as a business impact from DDoS attacks. This should come as no surprise since bandwidth comes at a cost, the failure to meet service-level-agreements (SLAs) can result in hefty penalties, and attacks can be time-consuming to deal with and waste valuable resources.

One-third of data center operators reported customer churn as a business impact. This also should come as no surprise since customer confidence in the availability of data center services can be shaken when the services are not available. As a result, the customers may feel that it is better to change to a datacenter that has better DDoS attack prevention.

Revenue loss was reported by approximately one-third of the survey responders. According to the survey, the reasons were because the customers were not required to pay for services when they were unavailable, and the primary business of the organization is affected by an attack (Anstee et al., 2012).

1.4 Related Work

In 2006, Pack, et all. evaluated the efficiency of Access Control List (ACL) against DDoS attacks. The primary tool used to gather legitimate and attacking traffic values was Netflow. The result showed that the number of ACL rules affected the collateral damage (legitimate traffic was dropped unintentionally). By using 5 ACL rules, the collateral damage was 45%. This number significantly reduced to 15% if authors used 50 ACL rules. Interestingly, if the number of ACL rules used was more than 50, it would increase the collateral damage values again (Pack, Yoon, Collins, & Estan, 2006).

In 2007, Chen, et all. proposed a unique technique to detect DDoS attacks on networks. The technique was called Distributed Change Point Detection. This technique detected attacks by using CAT (Change Aggregation Tree) to monitor the propagation of abrupt traffic changes inside the network. If traffic exceeded a preset threshold, an attack was identified. The testbed ran over multiple network domains, which had about 90 routers. There were four types of DDoS attacks used in this study, which were TCP, UDP, ICMP, and Smurf attack. The primary tool used to generate the attack traffic was Stacheldraht (Chen, Hwang, & Ku, 2007).

In 2009, Lu, et all. evaluated the impact of a UDP flood attack on a network using a testbed method. The performance metrics used in this study were the packet loss rate, delay and jitter. The network consisted of 9 routers and 14 computers with Intel Celeron 2.1 and 512 memory running Linux. Three routers were connected with a 100M switch while the bandwidth of cables was between 10Mbps to 100Mbps. Iperf was the primary tool used to generate UDP flood traffic. In terms of experimental results, it showed that the average of packet loss before the attack was 0% while the delay jitter was 32.3%. However, the packet loss value during the attack went up to 14.08% while the jitter value slightly decreased from 32.3% to 29.7% (Lu, Gu, & Yu, 2009).

In 2009, Rui, et all. proposed a UDP flood attack prevention method using Negative Selection Algorithm. To achieve this, the traffic threshold was set. If the UDP requests were greater than the threshold, the intrusion detection system would drop packets. The algorithm also had the ability to check whether the IP addresses came from the same region or not. If they were from different regions, they were likely to be spoofed IP addresses, and therefore, there was no reason to forward them to the next process. In addition, the simulation program in this study was built by .net 2005 and ran in a Windows Server 2003 system, and the total number of IP addresses tested was 12,960,000 IP addresses (Rui, Li, & Ling, 2009).

In 2011, Rao conducted DDoS experiments based on TCP and UDP flood attacks. The network environment in this experiment consisted of 4 components: an attacker computer, legitimate computer, router, and victim computer. A traffic generator called Hyenae was used to generate the attack traffic, while Netflow and Wireshark were used to monitor the impact on the server during the attack. There were two defense mechanisms used in this study: Access Control Lists and Rate Limiting. The results showed that the average RTT of the server before the attack was 0.834ms, while, during the attack, this number increased to 8.782ms. After using Access Control Lists and Rate Limiting, the average RTT went down to 1.093ms and 6.985ms, respectively. In terms of traffic utilization, the result showed the average network traffic rate before the attack was constant at 241 Kbps, while, during the attack, this number significantly went up to 3216 Kbps (Rao, 2011).

In 2011, Mohd, et all. proposed a UDP flood attack defense called Protocol Share Based Traffic Rate Analysis (PSBTRA). The result from the experiment showed that TCP was the main protocol used during the normal network operation (at 96.47%). During the UDP flood attack, however, the main protocol was changed from TCP to UDP (at 85.36%). Based on this result, the authors developed the UDP flood detection by comparing the proportion of incoming UDP traffic with the proportion of TCP and ICMP traffic. The detector would drop UDP packets if the proportion of UDP was greater than the proportion of TCP and ICMP. The advantage of this solution is that it has low computational overheads, and it can be applied in the real network environment (Z. Mohd et al., 2011).

In 2011, Arora, et all. evaluated the impact of a UDP flood attack on the network using a simulation method. The network consisted of 19 nodes in total. Ten nodes were used as authorized nodes, 5 nodes as routers, 2 nodes for an attacker and victim, and 2 nodes were used as a server and a control node. UDP was used as an attacking protocol. To evaluate the impact of the attack, the authors ran legitimate traffic for 40 seconds, then launched the attack for 10 seconds during that time (Arora, Kumar, & Sachdeva, 2011).

In 2012, Kaur, et all. conducted an experiment on DDoS attacks using simulation software called Bonesi. The network in this study consisted of 3 computers: an attacker computer, legitimate computer, and FTP server. The purpose of this research was to study the impact of the user throughput between computer nodes before and during a UDP flood attack. The result showed that the average bandwidth before the attack was around 75 Kbps, while, during the attacks, the average bandwidth significantly went up to 130 Kbps (Kaur, Sachdeva, & Kumar, 2012).

1.5 Motivation

The main motivation of this research is to investigate the impact of a UDP flood attack and provide results using the new generation of Windows and Linux platforms, namely, Windows Server 2012 and Linux Ubuntu 13. The related works show that little work has been done to date on the testbed evaluation of these attacks on modern operating systems.

Another motivation is the need to develop suitable solutions to address the rising cases of DDoS attacks on the computer network. Despite its impacts, there are no effective strategies to deal with such attacks although they have been in existence for almost a decade (Singh & Juneja, 2010). As York (2010) observed, many organizations, and government institutions have fallen victim to DDoS attacks. These organizations have lost consumers, important data, and some of them have gone out of business because of attacks. Governments have also lost important national data, impairing their ability to deliver services to the public. Consequently, an understanding of the characteristics of DDoS can assist in the development of appropriate security measures and could be used to prevent attacks in the future.

The recent attacks on numerous banks in America in 2012 also indicated the need to come up with robust defense mechanisms against DDoS attacks. In that incident, a malicious hacker circumvented security mechanisms on servers in the financial sector. The attacker turned them into Zombies to launch attacks on bank websites including *Bank of America*, *PNC Bank*, and *U.S. Bank*. As a result, the attacks caused outages of websites, online services, and the operations of the banks for several hours.

The attacks could have been prevented if appropriate security systems had been in place. All of these examples demonstrate the ineffectiveness of existing security mechanisms to detect and prevent DDoS attacks, and, as a consequence, are the motivation behind this research.

1.6 Research Contribution

This research aims to produce new results that no other researchers have found before. Previous studies showed there were many researchers who used a simulation method to carry out their DDoS experiments. However, their methods usually relied on simulation software to simulate a switch, router, and attacking traffic, which are not realistic environments and may not be applied on the real network. Many researchers also tried to propose their DDoS defenses against the attack, however, many did not compare the efficiency of their solutions with other defenses.

As a result, the purpose of this study is to evaluate the impact of a UDP flood attack on the new generation of Windows and Linux platforms using a testbed method. This method allows a researcher to set up a DDoS experiment using real traffic, real hardware, and real network environment. In this study, many comprehensive parameters were used, e.g., the user throughput, round-trip time, packet loss, CPU utilization, and jitter. This research also evaluates and compares the strengths and weaknesses of existing defense mechanisms such as Access Control Lists, Threshold Limit, Hybrid Method, IP Verify, Network Load Balancing, and Software Firewall.

1.7 Structure of the Thesis

This thesis consists of eight chapters. Chapter one contains an introduction, which briefly mentions the history of DDoS attacks, the motivation behind this study, followed by the contribution of this research. Chapter two provides an overall view of distributed denial of service attacks including the characteristics of DDoS attacks, different types of DDoS attacks, and methods of studying DDoS attacks.

The third chapter covers the details of a UDP flood attack, which is the type of DDoS attack used in this study. This chapter explains the method of a UDP flood attack, and how the attack exploits the design of UDP protocol. The chapter also reviews UDP flood attack tools, and the background of the two protocols involved in a UDP flood attack, which are User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP).

The fourth chapter covers the hypotheses and the methodology employed for this research. This chapter also presents the procedures of data collection and the steps involved in the process of the literature review and the experiment for this research. Chapter five includes a detailed explanation of the network diagram and the specification of all hardware and software used in the test lab. This chapter also contains configurations, commands and monitoring tools used during the test.

Chapter six covers the evaluation of a UDP flood attack on the webserver using Windows Server 2012. The metrics used in this study were user throughput, round-trip time, packet loss, CPU utilization, and jitter. At the end of this chapter, six defense mechanisms results are described along with comparisons between them. Chapter seven covers the performance analysis of a UDP flood attack and defenses on Linux Ubuntu 13. Chapter eight is the final chapter, which covers the conclusion, discussion, and directions for future works.

1.8 Chapter Summary

This chapter described the advantages of the Internet and the security perspective. This chapter also discussed the impacts of DDoS attacks in past years using the survey conducted by Arbor networks incorporation in 2012. The final part in this chapter described related works, the motivation behind this research, and outlined the structure of this report.

The next chapter provides knowledge about Distributed Denial of Service attacks in detail.

CHAPTER 2

DISTRIBUTED DENIAL OF SERVICE ATTACKS

This chapter covers the details of distributed denial of service attacks. Section 2.1 explains the definition of DDoS attacks and the attack methods. Section 2.2 describes the characteristics and the purpose of DDoS attacks. Section 2.3 covers three categories of DDoS attacks. Section 2.4 gives the overview of the different types of DDoS attacks. Section 2.5 explains the existing DDoS prevention and defense mechanisms. Section 2.6 describes five common parameters that can be used to evaluate the impact of DDoS attacks.

2.1 Definition of Distributed Denial of Service Attacks

A Denial of Service (DoS) attack is one of the network attacks which makes resources and services unavailable to its legitimate users. The main purpose of the attack is to disable the uses of services on the Internet or the targeted network. Typically, the attack can be launched either from a local or remote location (Singh & Juneja, 2010).

“Denial of Service” and “Denial of Service Attack” are two completely different concepts; the former refers to an event or a situation, and the latter refers to an intention-driven illegal act (Z. Mohd, Idris, Hussain, Stiawan, & Awan, 2011).

A Distributed Denial of Service (DDoS) attack is part of a DoS attack; an attacker launches a massive number of attack packets from multiple distributed sources to the targeted machine, which makes it more dangerous than a traditional DoS attack. There are four components involved in the DDoS attacks:

1. Real Attacker
2. Host Computer or Victim
3. Master Control Program
4. Zombies

The victim is the computer that is chosen for the attack, whereas, the real attacker is the person who is working behind the method for the attack. Normally, he/she works behind the shield of the master control program, which makes it harder to trace the attack back to its true source. The master control program is the interface between the attacker and Zombies.

Zombies are computers that have been compromised and used to attack the victim computer (Singh & Juneja, 2010).

2.2 Characteristics of DDoS Attacks

The DDoS attack is an advanced and upgraded version of the DoS attack both of which have the same goal, that is, to shut a system down by exhausting the resources of the targeted network (Oriyano & Gregg, 2010). The following list describes some of the characteristics of DDoS attacks:

- Attacks of this type are characterized by being very large using hundreds or even thousands of computers to conduct the attack.
- DDoS attacks have two types of victim, namely, primary and secondary. The former is the recipient of the actual attack, while the latter are the computers used to launch the attack itself.
- DDoS attacks can be very difficult to track back to the true source of the attacker because there are many computers involved, and the attacker can also use spoofed IP addresses to hide his/her IP addresses.
- Defense is very difficult due to the number of attackers involved. Configuring a firewall or router to black few IP addresses is possible. However, with the large number of malicious IP addresses created by thousands of Zombie computers, it is nearly impossible to identify and block the source of the attack (Ciampa, 2012).
- The impact of the DDoS attack is increased over a standard DoS attack because many hosts are involved, which multiplies the attack’s strength and power (Oriyano & Gregg, 2010).

2.3 Categories of DDoS Attacks

DDoS attacks are not all the same. DDoS attacks can be separated into three broad categories, depending on how they carry out their goal of denying the service to legitimate users. The following is a list of DDoS attack categories:

1. Consumption of bandwidth
2. Consumption of resources
3. Exploitation of programming defects

2.3.1 Consumption of Bandwidth

Consumption of bandwidth is one of the most common DDoS attacks in computer networks. The purpose of this type of attack is to exhaust the network bandwidth flowing to and from a targeted machine. Adding more bandwidth is not a good solution because the attacker does not have to exhaust the bandwidth of the targeted system entirely, but rather use up so much of it that performance becomes unacceptable to users (Oriyano & Gregg, 2010). The following list is an example of well-known forms of attacks in this category. In addition, the details of these attacks are discussed in Section 2.4.

- Smurf Attack
- UDP Flood Attack

2.3.2 Consumption of Resources

This type of attack is similar to bandwidth consumption. The purpose of the resource consumption-based attacks is to consume a limited resource such as CPU and RAM. Unlike bandwidth consumption, the intruder focuses on the resources on a single system, which causes the service or the entire system to become overloaded (Oriyano & Gregg, 2010). The following list contains examples of the common forms of this attack. In addition, the details of these attacks are discussed in Section 2.4.

- TCP and UDP Flood Attack
- Smurf Attack
- Ping Flood Attack

2.3.3 Exploitation of Programming Defects

This type of attack exploits known weaknesses in the system’s design. Vulnerabilities of this type may have been exposed due to flaws in the system’s design that were unintentionally put in place by the programmers or developers of the system (Oriyano & Gregg, 2010). The following list is an example of the well-known forms of attack in this category. In addition, the details of these attacks are discussed in Section 2.4.

- Ping of Death
- ARP Request Attack

2.4 Common Types of DDoS Attacks

This section discusses the most common types of DDoS attacks, their characteristics, and mitigation techniques. These attacks are TCP SYN flood attack (Section 2.4.1), Smurf attack (Section 2.4.2), Application-Level flood attack (Section 2.4.3), ARP request attack (Section 2.4.4), ICMP flood attack (Section 2.4.5), Ping of Death (Section 2.4.6), and UDP flood attack (Section 2.4.7).

2.4.1 TCP SYN Flood Attack

The TCP SYN flood attack exploits a vulnerability of TCP 3-way handshake (SYN, SYN-ACK, and ACK). During the attack, the attacker sends SYN packets with source IP addresses that do not exist to the targeted computer (e.g., server). When the server stores the request information into the memory stack, it will wait for confirmation from the client that sends the request. While the request is waiting to be confirmed, it will remain in the memory stack.

Because these source IP addresses are spoofed IP addresses, the server will not receive confirmation packets. Eventually, the requests will accumulate and fill up the entire memory stack. Figure 2.1 shows the method of a TCP flood attack (Kavisankar & Chellappan, 2011).

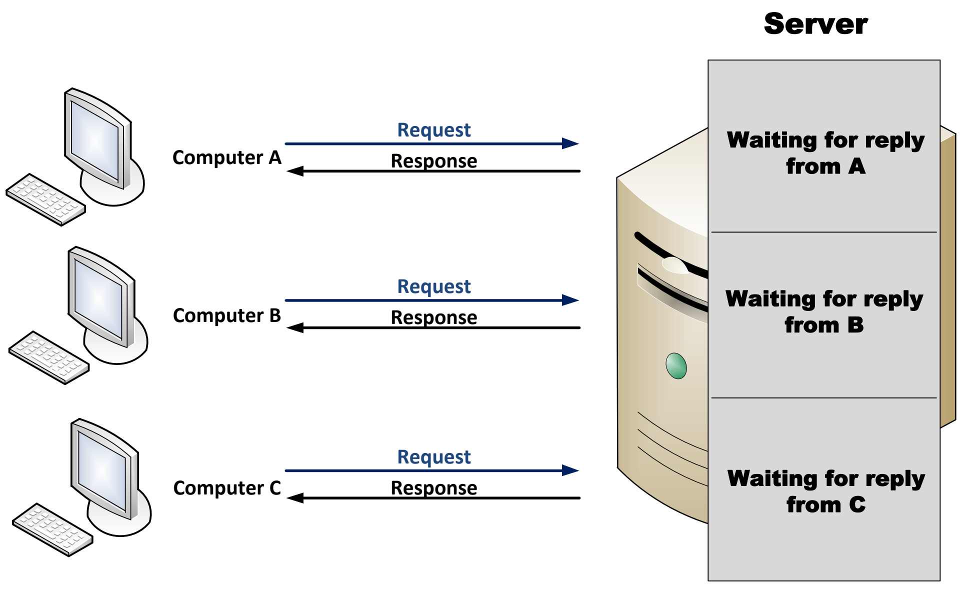


Figure 2.1: Vulnerability of TCP 3-way Handshake (Ciampa, 2012)

2.4.1.1 IP Spoofing in TCP Flood Attack

IP Spoofing is used to hide the original IP address of the attacker. This technique allows the attacker to gain unauthorized access to the network by changing his original IP address to the IP address of the trusted machine in that network. A TCP flood attack can also be launched from compromised computers using spoofed IP address, or genuine source IP addresses, given that the compromised computers used are configured to ignore the SYN/ACK packets from the target (Kavisankar & Chellappan, 2011).

2.4.1.2 Mitigation Techniques for TCP Flood Attack

A TCP flood attack can be prevented by using the server as the attack detector. To detect the attack, the SYN request sent by the client will be stored in the database and will wait until the client sends the acknowledgement (ACK signal) to the server. Information on the client that is stored in the database consists of the IP address and the SYN count. If the number of SYN requests exceeds the threshold, it drops the packets from that source IP address (Kavisankar & Chellappan, 2011).

The network administrator can also use SYN cookies to prevent a TCP flood attack based on spoofed IP addresses. SYN cookies will not allocate resources until the 3-way handshake completes. However, one of the disadvantages of this method is that this solution uses huge computing resources such as the CPU and RAM of the server (Lemon, 2002).

2.4.2 Smurf Attack

The smurf attack is a type of DDoS attack that is designed to exhaust the computer resource and network bandwidth. According to the ICMP protocol, when a device on the network receives a “ping” request, it replies to the source IP address with a “pong” message informing the status of the receiver. The smurf attack exploits the design of ICMP and TCP/IP. During the attack, the attacker crafts large numbers of ICMP packets with the intended victim’s spoofed source IP address, and broadcasts to the computer network (Wang, 2009).

All devices on the network that have received these broadcast messages will respond with “pong” messages back to the victim computer. This will exhaust the computing resources of the victim and associated computers. In addition, the degree of the attack depends on the number of machines on the network (that receive the broadcast messages) and the attack packet rates sent by the attacker (Wang, 2009). Figure 2.2 illustrates the smurf attack methods.

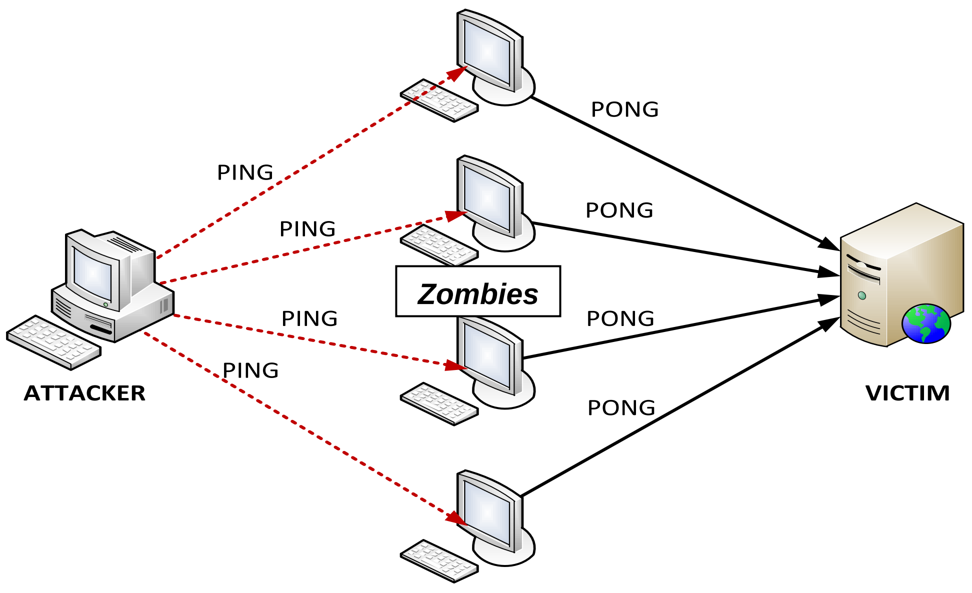


Figure 2.2: Smurf Attack Methods (Wang, 2009)

2.4.2.1 Mitigation Techniques for Smurf Attacks

The smurf attack can be prevented by using a firewall or a router to filter ICMP packets. According to Figure 2.2, there are three components involved in the smurf attack: the attack network, intermediary network, and the victim network. In order to prevent smurf attacks, all three components must do their part in keeping the attacker out. First, the attacker's network should not allow its users to craft spoofed IP addresses. To achieve this, the firewall needs to be trained to inspect both ingress and egress traffic. Similarly, the intermediary network must not be exploited by the attacker for amplifying the malicious traffic (Kumar, 2007).

Another way to protect the network against a smurf attack is to disable the IP directed broadcast, which is often not needed. In addition, many operating systems can be configured to prevent the machine from responding to ICMP messages such as the security capabilities of Windows Service Pack 2 (Kumar, Azad, Gomez, & Valdez).

2.4.3 Application-Level Flood Attack

Traditionally, DDoS attacks are carried out at network layer level, e.g., TCP flood attack, and UDP flood attack (known as Net-DDoS attacks). The purpose of these attacks is to consume the network bandwidth and deny service to the legitimate users of the victim systems. However, when the Net-DDoS attacks fail, the attacker shifts the offensive strategies to application layer attacks (known as App-DDoS attacks).

App-DDoS attacks are designed to exhaust the server resources such as CPU, memory, bandwidth, and disk utilization. This attack occurs when the attacker sends a large number of queries to the victim’s database to bring the server down, or attack the webserver by sending a massive number of HTTP GET requests. App-DDoS attacks on the webserver are similar to a “flash crowd” which is the situation when many legitimate users access a popular website simultaneously. As a result, it is difficult to distinguish between legitimate and attacking traffic (Xie & Yu, 2009).

2.4.3.1 Mitigation Techniques for Application-Level Flood Attack

Client Puzzle Protocol (CPP) is an algorithm that will not allow any abuse of the server resources. According to this algorithm, any client that wants to establish a connection with the server has to first correctly solve a mathematical puzzle. After solving the mathematical puzzle, the client returns the solution to the server and the server will quickly confirm, reject or drop the connection, based on the client’s solution (Rajesh, 2013).

2.4.4 ARP Request Attack

In computer networks, media access control addresses (MAC addresses) are used in data transfer. It is important to convert an IP address to a MAC address in order to communicate in a LAN. Address Resolution Protocol (ARP) is used for this purpose. When a source wants to know the MAC address of the destination IP node, it broadcasts the ARP request to every host in the network. The destination node sends a reply to the source with a MAC address through ARP reply in a unicast mode. Figure 2.3 shows an example of communication in LAN using ARP (Vidya & Bhaskaran, 2011).

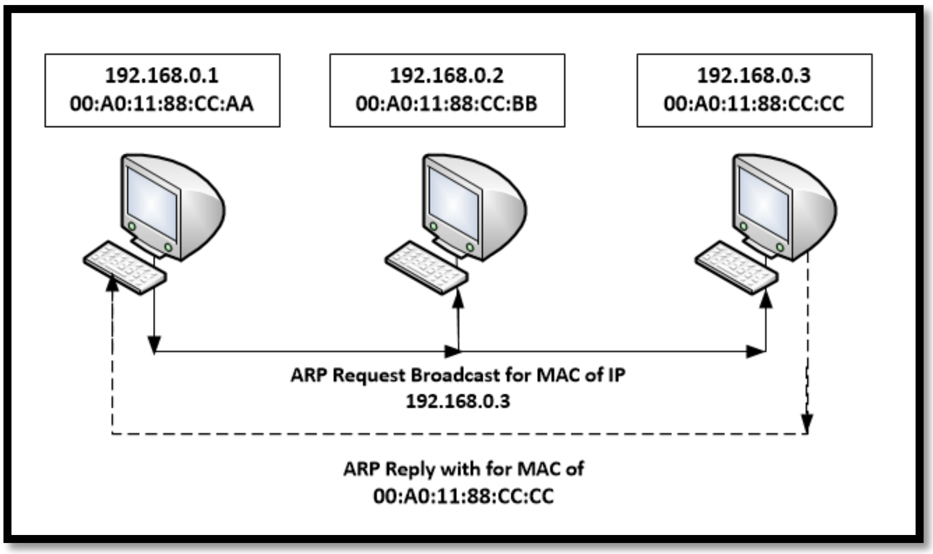


Figure 2.3: Communication in LAN using ARP (Vidya & Bhaskaran, 2011)

In terms of an ARP request attack, it is an attack situation intentionally created by an attacker from within the local network. During the attack, the attacker keeps sending a large number of broadcast packets, with IP addresses within a subnet range or even to IP addresses not present in the local subnet. The purpose of this attack is to reduce the bandwidth with unwanted traffic or collect the IP/MAC address details of all machines for later attacks. In addition, when a network administrator monitors the network traffic using a network analyzer, he/she will detect a large series of Address Resolution Protocols (Vidya & Bhaskaran, 2011).

2.4.4.1 Mitigation Techniques for ARP Request Attack

ARP request attacks can be prevented by monitoring the ARP traffic on each LAN segment. Programs such as SNMP and ARPwatch can be used to monitor changes in ARP tables in a router, and can switch to raise the alarm for the onset of such ARP attacks. Another way to prevent these attacks is to set up the traffic threshold for broadcast/multicast traffic on a per-port basis. In addition, these thresholds on a per-port basis should be set up by limiting the bandwidth consumed by ARP broadcasts on a switch port (Kumar & Gomez, 2010).

2.4.5 ICMP Flood Attack

ICMP is a part of the TCP/IP suite that is designed to handle error and control messages. One of the best known examples in practice is the ping utility. It uses ICMP to check remote hosts for responsiveness, display the round-trip time, or detect the communication failures between hosts (Bogdanoski & Risteski, 2011).

In terms of an ICMP flood attack, the attack overwhelms the targeted computers by sending a large number of “ICMP Echo Requests” to the targeted computer. The victim receiving these packets will reply “ICMP Echo Reply” back to the source IP address. As a result, it will consume both incoming and outgoing bandwidth, and in extreme cases it can disable the network connectivity (Bogdanoski & Risteski, 2011).

2.4.5.1 Mitigation Techniques for ICMP Ping Flood Attack

ICMP flood attacks can be prevented by blocking the ICMP packets from both incoming and outgoing traffic. One of the disadvantages is that it may cause issues when using a ping command to diagnose network problems. Another solution is to block certain types of ICMP packets or limit the speed of sending ICMP messages from the source. As a result, if registered ICMP traffic is higher than the threshold, the packets are dropped (Bogdanoski & Risteski, 2011).

2.4.6 Ping of Death

A ping of death is a type of attack on a computer that involves sending an oversized ping packet to a computer. According to RFC 791, a correctly formed ping message is typically 56 Bytes in size, or 84 Bytes when the IP header is considered (Exist & Postel, 1981). In the past, many computer systems could not properly handle a ping packet larger than the maximum IPv4 packet size of 65,535 Bytes. Larger packets could crash the targeted computer. Nevertheless, this type of attack would not be harmful to the computer system today since modern operating systems are all patched against this vulnerability (Erickson, 2008).

2.4.6.1 Mitigation Techniques for Ping of Death

Ping of death can be prevented by adding “checks” in the reassembly process. The check is used to ensure that the sum of the “Total length” and “Fragment Offset” fields of each IP fragment is not bigger than the size specified (65,536 Bytes). If the sum is bigger, then the IP fragment is ignored, and the packet is deemed invalid. In addition, the check can be performed by firewalls in order to protect the computer systems that have not been patched against this attack. Another solution for this problem is to use a memory buffer to reassemble the packet so that it is larger than 65,535 Bytes (Erickson, 2008).

2.4.7 UDP Flood Attack

A UDP flood attack in DDoS attacks is a method causing host-based denial of service. It occurs when an attacker crafts a large number of UDP packets to random destination ports on a victim computer. The victim computer, on receipt of the UDP packet, would respond with appropriate ICMP packets if its computer ports are closed. Numerous packet responses would slow down the system or crash it (Rui, Li, & Ling, 2009). In addition, a UDP flood attack is a type of DDoS attack used in this study. The details of this attack, including its characteristics, the attack methods, and attack tools, are discussed in Chapter 3.

2.4.7.1 Mitigation Techniques for UDP Flood Attack

A UDP flood attack can be prevented by identifying the validity of source IP address. One of the effective solutions is Unicast RPF (Unicast Reverse Path Forwarding). When a router receives the IP packet, it checks the route table, and confirms whether the route information of the IP source address for the packet exists. If there is no route information for data return in the table, it is very likely that the IP source address is forged by the attacker, and then the router will drop this packet (Rui, Li, & Ling, 2009). In addition, the details of UDP flood attack defenses are discussed in Section 5.2.

2.5 Classification of DDoS Prevention Mechanisms

DDoS prevention mechanisms can be separated into two categories: general techniques and filtering techniques. The former involves the system protection of common preventive measures, while the latter utilizes the router to filter and drop attack packets (Gupta et al., 2010). The next section discusses general prevention techniques.

2.5.1 General Prevention Techniques

This section details five general prevention techniques that can be implemented against DDoS attacks on the network. These prevention techniques are disabling unused services (Section 2.5.1.1), installing latest security patches (Section 2.5.1.2), disabling IP broadcast (Section 2.5.1.3), firewalls (Section 2.5.1.4), and IP hopping (Section 2.5.1.5).

2.5.1.1 Disabling Unused Services

The concept of this technique is to disable network services that are not needed or are currently unused. Since there are over 60,000 usable ports in a computer, it means, the fewer the applications and open ports in the hosts, the less likely there is a chance to exploit vulnerabilities by attackers. This technique is mainly used to prevent attacks such as UDP flood attacks, TCP flood attacks, and ICMP flood attacks (Gupta et al., 2010).

2.5.1.2 Installing Latest Security Patches

Before attacking the network, the attacker needs to find the vulnerabilities of the targeted system. These vulnerabilities may be from improper network design, a lack of network security, or applications being used which have security holes (especially operating systems). Therefore, it is important that all the relevant latest security patches must be up to date. Due to the quantity of patches, organizations may use an automated patch update service to ensure that patches are installed in a timely fashion (Ciampa, 2012).

2.5.1.3 Disabling IP Broadcast

Another way to protect a network from DDoS attacks is to disable IP broadcasts. Some DDoS attacks such as the smurf attack, exploit this design by sending a broadcast address with the victim’s spoofed source IP address to the computer network. All devices on the network that have received these broadcast messages will respond with “pong” messages back to the victim computer. This problem can be prevented by disabling the broadcast function so that an attacker cannot use IP broadcasts as the amplifiers (Chaba, Singh, & Aneja, 2009).

2.5.1.4 Firewalls

The firewall can be used to protect the machines within the network from the simple flooding type of attacks. The firewall has simple rules such as to allow or deny protocols, IP addresses and computer ports. However, the disadvantage of the firewall is that it may not be able to prevent some complex attacks or prevent attacks on the web service (e.g., the TCP SYN flood attack). This is because it cannot distinguish between legitimate and attacking traffic (Gupta et al., 2010).

2.5.1.5 IP Hopping

DDoS attacks can be prevented by changing the location or IP address of the active server proactively within a pool of homogeneous servers or with a pre-specified set of IP address ranges (Gupta et al., 2010). By doing this, the victim computer's IP address is invalidated by changing it with a new one. Once the IP address change is completed, all Internet routers will be informed and edge routers will drop the attacking packets. However, the disadvantage of this solution is that the attacker can launch the attack at the new IP address (Gupta et al., 2010).

2.5.2 Filtering Defenses Techniques

This section details five filtering defense techniques that can be implemented against DDoS attacks on the network. These prevention techniques are ingress and egress filtering (Section 2.5.2.1), history-based IP filtering (Section 2.5.2.2), router packet-based filtering (Section 2.5.2.3), capability-based method (Section 2.5.2.4), and a source address validity enforcement (Section 2.5.2.5).

2.5.2.1 Ingress and Egress Filtering

Ingress filtering is an approach to set up an edge router to disallow incoming packets with illegitimate source addresses into the network. In terms of egress filtering, it is an outbound filter, which ensures that only valid IP addresses can leave the network otherwise they are dropped. The key requirement for the filtering techniques is information on the expected IP addresses at a particular port (Gupta et al., 2010).

Both ingress and egress filtering can be applied not only to IP addresses, but also port number, protocol type, or any other criteria of importance. However, it is difficult to deploy ingress/egress filtering universally. If the attacker carefully chooses a network without ingress/egress filtering to launch the attack, or the attacker spoofs IP addresses from within the subnet, the attack can go undetected (Gupta et al., 2010).

2.5.2.2 History-Based IP Filtering

History-based IP filtering technique is based on the principle that the set of source IP addresses during the normal operation tends to remain stable, whereas, during DDoS attacks, most IP addresses have not been seen before. Based on this idea, this technique utilizes IP Address Database (IAD) to record frequent source IP addresses. Therefore, if the source IP address is not in the database, the filter determines that the IP address has been spoofed and drops this packet. One of the disadvantages of this solution is that it cannot prevent DDoS attacks based on real IP addresses. It also requires an offline database to keep track of IP addresses. As a result, the cost of storage and information sharing is high (Gupta et al., 2010).

2.5.2.3 Router Packet-Based Filtering

Router packet-based filtering is an improvement on the ingress filtering technique. It is based on the principle that each link in the core of the Internet has a limited set of source addresses. Therefore, if an unexpected source address appears in an IP packet on a link, it determines that the source address has been spoofed, hence the packet is dropped (Gupta et al., 2010). However, this scheme has several limitations. The first limitation is that the router-based filtering technique may drop legitimate packets if a routing table has been updated or changed. Another limitation is that this technique relies on valid BGP messages to configure the filter. If an attacker can hijack a BGP session and change BGP messages, it is possible to mislead routers to update filtering rules in favor of the attacker (Gupta et al., 2010).

2.5.2.4 Capability-Based Method

The Capability-based method allows the destination to control the traffic directed towards itself. To achieve this, a router inserts “a mark” in the source packet before sending it to the destination. The destination can determine whether the packet should be allowed or dropped. If permission is granted the packet can pass through the network via the router, otherwise the packet is rejected. Packets that do not have a “mark” are considered as suspicious packets. One of the benefits of this architecture is that it gives the destination the ability to control the traffic according to its own policy (Anderson, Roscoe, & Wetherall, 2004).

2.5.2.5 Source Address Validity Enforcement

The Source Address Validity Enforcement (SAVE) is a new protocol that allows routers to store information on expected source IP packets, and drop incoming IP addresses that are not in the list. The purpose of the protocol is to inform routers about the scope of the incoming IP address that should be expected at each interface. To achieve this, SAVE broadcasts a message that contains valid source address information to all destinations. By doing this, it allows each router to create an incoming table that associates each link of the router (Gupta et al., 2010). Nevertheless, the efficiency of this solution depends on the number of SAVE deployments in the networks associated because the attacker may spoof the IP addresses of the network that do not deploy this protocol (Gupta et al., 2010).

2.6 DDoS Impact Metrics

This section details the five common parameters that can be used to evaluate the impact of a DDoS attack on the network. In addition, the details of these parameters and an example of the outputs will be presented in Section 4.3.2.1.

- Throughput
- Round-trip Time
- Packet Loss
- CPU Utilization
- Jitter

2.6.1 Throughput

Throughput is defined as the number of bytes transferred from the source to the destination. Performance of the throughput between networks can be impacted or affected by some activities such as the network design, LAN cards, switches, and routers. This metric should not be used to measure the impact of a DDoS attack in some applications, e.g., video conferencing, and voice over IP applications, because a high throughput value may still not satisfy the quality of service required by the user (Mirkovic, Sonia, et al., 2009).

2.6.2 Round-Trip Time

Round-trip time is defined as the interval between the time when a request is issued and the time when a complete response is received from the destination. This metric is mainly used to measure the service denial of interactive applications such as web applications, FTP, and telnet (McCabe, 2010). RTT should not be used to measure the impact of DDoS attacks on the non-interactive applications (such as email or short message service) since these applications are insensitive to delay, or they have large thresholds for acceptable request and response delay (Mirkovic, Sonia, et al., 2009).

2.6.3 Packet Loss

Packet loss is defined as the number of packets or bytes lost due to the interaction of the legitimate traffic with the attack, or due to collateral damage from a defence's operation (Yaar, Perrig, & Song, 2004). The loss metric primarily measures the presence and extent of congestion in the network due to flooding attacks, but cannot be used for some attacks that do not congest network resources or do not continually create congestion, e.g., Ping of Dead attack (Mirkovic, Sonia, et al., 2009).

2.6.4 CPU Utilization

CPU utilization is the percentage of a computer's CPU resource taken during the processing of an application or a task. Some types of DDoS attacks, such as TCP flood attack, UDP flood attack, and Smurf attack, are designed to exhaust the victim's CPU resource (Oriyano & Gregg, 2010).

2.6.5 Jitter

Jitter or packet delay variation is the variation in the time between packets arriving, which can be caused by timing drift, route changes, or network congestion. Some types of DDoS attacks, such as UDP flood attack and smurf attack, can significantly increase jitter values. In addition, high jitter values can impact the performance of some applications, e.g., voice over IP, online games, and video conference applications (Marti, Fuertes, Fohler, & Ramamritham, 2001).

2.7 Chapter Summary

This chapter reviewed the background of DDoS attacks including the characteristics, and different types of DDoS attacks. The different types of DDoS prevention mechanisms were also discussed. These mechanisms can be separated into two categories: general techniques and filtering techniques. The former involves the system protection of common preventive measures, while the latter utilizes the router to filter and drop attack packets. The final section covered five common parameters used to evaluate the impact of DDoS attacks.

The next chapter provides the details of a UDP flood attack, which is the type of DDoS attack used in this study.

CHAPTER 3

UDP FLOOD ATTACK

This chapter covers the details of a UDP flood attack, which is the type of DDoS attack used in this study. Section 3.1 provides the definition of a UDP flood attack. Section 3.2 describes the characteristics of a UDP flood attack, and how the attack exploits the design of a UDP protocol. Section 3.3 describes a technique that can be used to hide the true source of the attack. Section 3.4 explains the method of a UDP flood attack using Zombies. Section 3.5 covers several types of UDP flood attack tools. Section 3.6 covers a brief description of the two protocols involved in a UDP flood attack.

3.1 Definition of a UDP Flood Attack

A User Datagram Protocol (UDP) flood attack in DDoS attacks is a method causing host-based denial of service. It occurs when an attacker crafts numerous UDP packets to random destination ports on a victim computer. The victim computer, on receipt of the UDP packet, would respond with appropriate ICMP packets if one of these ports is closed. Numerous packet responses would slow down the system or crash it (Rui, Li, & Ling, 2009).

Both TCP and UDP flood attacks are categorized as high rate flood attacks because they are launched by flooding a massive amount of TCP or UDP datagrams to overwhelm the victim computer. However, the difference between them is that the former intends to consume the victim computer's resources (e.g., CPU) while the goal of a UDP flood attack is to exhaust the connection bandwidth. This makes a UDP flood attack more severe than a TCP in terms of its ability to cause more degradation of the legitimate traffic bandwidth (Mirkovic, Hussain, Fahmy, Reiher, & Thomas, 2009).

3.2 Characteristics of a UDP Flood Attack

A UDP flood attack exploits the generic design of UDP protocol behavior. Unlike TCP, UDP is a connectionless protocol and does not involve 3-way handshaking functionality as in TCP. During the attack, a large number of UDP packets are sent to either random or specified ports on the victim computer. Typically, this attack is designed to attack random victim ports. When the victim computer receives a UDP packet, it will determine which application is waiting on the destination port. If there is no application that is waiting on the port, the victim computer will generate an ICMP packet of ‘‘destination unreachable’’ to the forged source address. If enough UDP packets are delivered to ports of the victim, the targeted computer will go down (Singh & Juneja, 2010).

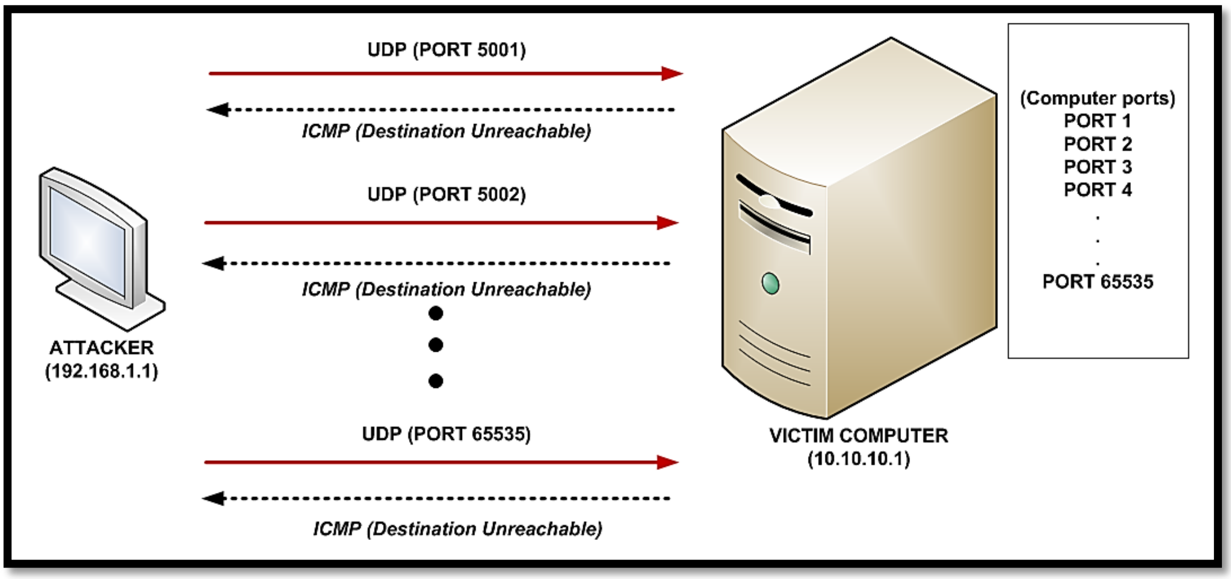


Figure 3.1: UDP Behavior Analysis to Closed Ports (Singh & Juneja, 2010)

Figure 3.1 shows an example of UDP behavior to closed ports. The victim computer will respond with an ICMP message, every time a UDP request is received on a closed port (Singh & Juneja, 2010). For example, the attacker sends the UDP packet to the destination port (port 5001) on the victim computer. If this port is closed, the victim computer will respond by sending an appropriate ICMP packet back to the attacker computer (ICMP destination unreachable). The attacker will keep sending UDP packets to other computer ports of the victim (ranging from 1 – 65535) and the victim system, on receipt of the UDP packets has to respond with ICMP packets if one of these ports is closed.

3.3 UDP Flood Attack using Spoofed IP Address

IP spoofing is a technique that involves replacing the IP address of an IP packet's sender with another machine's IP address. This technique is most often used in distributed denial of service attacks, and especially in a UDP flood attack. In such an attack, the goal is to flood the victim computer with overwhelming UDP packets, and the attacker does not care about receiving responses to the attack packets (Singh & Juneja, 2010).

Packets with spoofed addresses are thus suitable for such an attack. They have many advantages for this purpose. For example, they are more difficult to filter because each spoofed packet appears to come from a different IP address, and these spoofed addresses also hide the true source of the attack. Moreover, when multiple compromised computers are participating in the attack, all sending spoofed traffic, it is very difficult to distinguish the difference between legitimate and attacking traffic (Singh & Juneja, 2010).

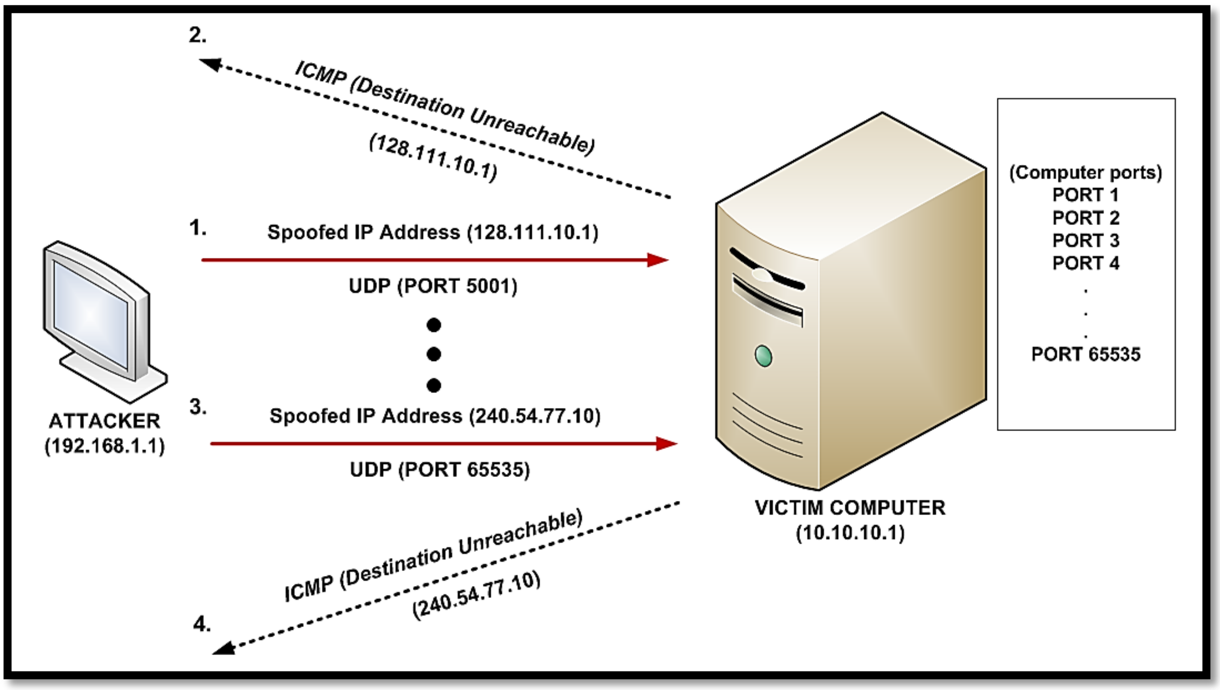


Figure 3.2: UDP Flood Attack using Spoofed IP Addresses (Singh & Juneja, 2010)

Figure 3.2 illustrates an example of a UDP flood attack using spoofed IP addresses. It shows that many UDP packets are sent from random source IP addresses to different port numbers. For example, the attack replaces his/her IP address (192.168.1.1) with a spoofed IP address (128.111.10.1) and sends the UDP packet to the destination port (port 5001) on the victim computer. If this port is closed, the victim computer will respond with an ICMP packet back to the spoofed IP address indicating that the port is closed. The attacker will keep changing his IP address and destination ports, and the victim computer, on receipt of the UDP packets, has to respond by sending ICMP packets back to the forged source addresses.

3.4 Mechanism of UDP Flood Attack using Zombies

There are four components involved in a UDP flood attack using Zombies: a victim computer, a real attacker, a master control program, and Zombies or Daemons. The victim is the computer that is chosen for the attack, whereas, the real attacker is the person who is working behind the method for the attack. Normally, he/she works behind the shield of the master control program, which makes it harder to trace back to it. The master control program is the interface between the attacker and the Zombies. Zombies are computers that have been compromised and used to attack the victim computer.

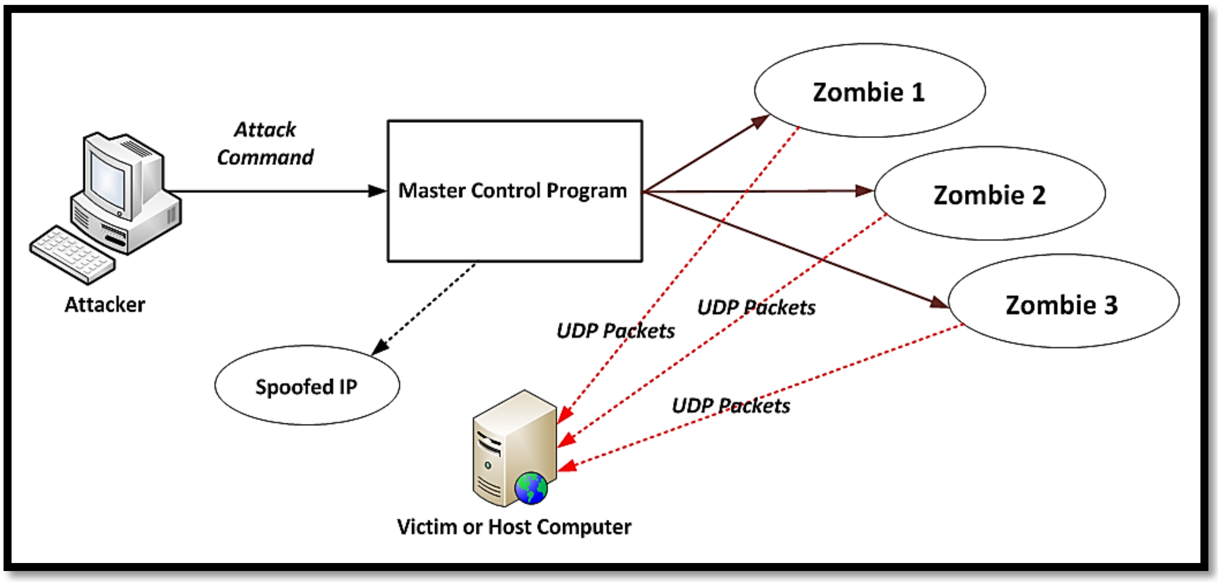


Figure 3.3: Mechanism of UDP Flood Attack using Zombies (Singh & Juneja, 2010)

Figure 3.3 illustrates the mechanism of a UDP flood attack using Zombies. A UDP flood attack occurs when the attacker sends the attack command, the victim’s IP address, attack methods, and the attack duration to a master control program. The master control program then forwards the attack instruction to their agents, which may be either Zombies or Daemons (compromised computers). The difference between Zombies and Daemons is that Daemons will be used in the direct attack method, whereas, Zombies will be used in the reflector method. In addition, the direct attack method is when the attacker arranges to send out a large number of UDP packets directly toward the victim computer. On the other hand, a reflector attack is an indirect attack in that intermediary nodes (e.g., routers) are innocently used as attack launchers (Chang, 2002).

On receiving the attack instruction from the attacker, Zombies start sending UDP packets including port numbers to the victim computer. When the victim computer receives the packets, it sends ICMP packets back to those Zombies indicating that the port is closed. Zombies will keep sending UDP packets to the victim computer until all its resources have been consumed (Singh & Juneja, 2010).

Multiple Zombies are under the control of each master control program and even the masters can be multiple, which leads to a large number of UDP packets being delivered to the victim computer. This ensures flooding the system by consuming the entire bandwidth and other resources (Singh & Juneja, 2010).

3.5 UDP Flood Attack Tools

One of the main reasons that makes DDoS attacks difficult to prevent is the simplicity and power of attack tools (Yaar, Perrig, & Song, 2004). In the past, an attacker needed to have technical knowledge of attack tools before they could be used. Today, however, many attack DDoS tools are freely available and do not require any technical knowledge. The following sections describe two UDP flood attack tools, which are the agent-based and IRC-based types (Douligeris & Mitrokotsa, 2004).

3.5.1 Agent-based Attack Tools

This section covers the agent-based UDP flood attack tools. Examples of the attack tools are Trinoo (Section 3.5.1.1), Tribe Flood Network (Section 3.5.1.2), TFN2K (Section 3.5.1.3), Shaft (Section 3.5.1.4), and Mstream (Section 3.5.1.5).

3.5.1.1 Trinoo

Trinoo is a bandwidth depletion attack tool that can be used to launch a UDP flood attack against one or many IP addresses. The attack uses constant-size UDP packets to target random ports on the victim computer. Typically, Trinoo agents need to be installed on the compromise machines (Zombies) first. This allows the attacker to remotely compile and run the attack via Zombies. The handler is used as an intermediary between the real attacker and Zombies, which is done via UDP packets (Douligeris & Mitrokotsa, 2004).

3.5.1.2 Tribe Flood Network

Tribe Flood Network (TFN) is an attack tool that provides an attacker with the ability to exhaust both the resources and bandwidth of a targeted machine. Communication between the attacker and the control master program is done via a command line interface but with no encryption between them. In addition, TFN uses ICMP packets to communicate between the handler and the Zombies which makes it harder to detect an attack, and attacks can often be overlooked by a firewall. TFN can implement different types of attacks including UDP, and Smurf attacks (Douligeris & Mitrokotsa, 2004).

3.5.1.3 TFN2K

TFN2K is a more advanced version of TFN. TFN2K adds many features including the encrypted messaging between all of the attack components (using CAST-256 algorithm). It also has the ability to hide the detection from intrusion detection systems. TFN2K can generate different types of DDoS attacks including UDP flood attacks, ICMP flood attacks, and Smurf attacks. An advantage of TFN2K is that the protocols used to communicate between attack components can be random during the attack, which makes it harder to detect the attack by scanning the network (Douligeris & Mitrokotsa, 2004).

3.5.1.4 Shaft

Shaft is an advanced version of Trinoo, which enables an attacker to communicate with the handlers via a TCP telnet connection. An advantage of Shaft is the ability to combine different types of attacks into one attack, e.g., to change between TCP, UDP, and ICMP flood attacks. A distinctive feature of Shaft is the ability to change the attacker's IP address and port in real-time during the attack, hence making detection by the firewall more difficult. Besides, Shaft also provides statistics on the flood attack. The statistics are used by the attacker to know when the victim system is completely down and allows him/her to know when to stop adding Zombies to DDoS attacks (Douligeris & Mitrokotsa, 2004).

3.5.1.5 Mstream

Mstream provides a simple point-to-point UDP flood attack that can obstruct the function of tables used by fast routing routines in switches. TCP and UDP packets are used as the main protocols for communication between attack components; however, they do not provide encrypted messaging between them. The advantage of Mstream is that the master machine can be controlled remotely by multiple attackers using a password-protected interactive login. This tool can also inform all connected attacks whether access to the program has been successful or not (Douligeris & Mitrokotsa, 2004).

3.5.2 IRC-based Attack Tools

Internet Relay Chat (IRC) based DDoS tools were developed after the agent handler attack tools and as a result, IRC-based tools are more sophisticated. They combine many of the advantages and distinctive features that can be found in the previous generation of attack tools. Examples of IRC-based tools include Trinity (Section 3.5.2.1) and Knight (Section 3.5.2.2).

3.5.2.1Trinity

Trinity is an IRC-based DDoS attack tool that can generate different types of DDoS attacks such as UDP flood and TCP flood attacks. Before launching an attack, each Trinity compromise machine needs to join a specified IRC channel and waits for commands. By using the legitimate IRC service for communication between the attacker and Zombies, it eliminates the need for a master machine and elevates the level of the threat (Douligeris & Mitrokotsa, 2004).

3.5.2.2 Knight

Knight is a lightweight and powerful IRC-based DDoS attack tool that provides different types of DDoS attacks including UDP flood and TCP flood attacks (Gupta et al., 2010). Knight is designed to run on Windows operating systems, and has features such as a checksum generator, and an automatic updater via HTTP or FTP. In addition, the Knight tool is typically installed by using a Trojan horse program (Douligeris & Mitrokotsa, 2004).

3.6 Protocols involved in a UDP Flood Attack

This section gives a description of two protocols involved in UDP flood attacks, namely, UDP and ICMP. This section also provides a brief description of TCP. In this study, TCP was used as legitimate traffic, and used to compare it with attacking traffic. Therefore, knowledge of these protocols is required.

3.6.1 User Datagram Protocol

The User Datagram Protocol (UDP) was designed by David Patrick Reed in 1980 (Sosinsky, 2009). It is one of the members of the Internet Protocol suite used for sending messages between hosts on the IP network. Unlike TCP, UDP is an unreliable protocol, which uses a simple transmission model with a minimum of protocol mechanisms and it does not attempt to ensure the validity of the data that is sent. However, this protocol provides checksums for data integrity, and port numbers of the source and destination for addressing different functions (Sosinsky, 2009).

UDP does not maintain the overhead, therefore, it can transfer information faster than TCP (Kurose & Ross, 2010). This makes UDP suitable for applications that do not require reliability, e.g., voice, music, and video applications. UDP is often used in both time-sensitive and real-time applications since dropping packets is preferable to waiting for delayed packets in both applications (Kurose & Ross, 2010).

A number of UDP's characteristics make it suitable for certain applications, including:

- It provides datagrams, which is suitable for modeling other protocols such as Network File System (NFS) and IP tunneling.
- It is transaction-oriented, which is suitable for simple query-response protocols such as Network Time Protocol (NTP) and Domain Name System (DNS).
- It is stateless, which makes it suitable for large numbers of clients such as in streaming media applications.
- It is simple, which is suitable for bootstrapping such as Dynamic Host Configuration Protocol (DHCP).
- The lack of retransmission delays makes it suitable for time-sensitive applications such as video conferencing and Voice over IP.
- It is compatible with unidirectional communication which makes it suitable for broadcast and shared information such as service discovery and broadcast time.

3.6.1.1 UDP Packet Structure

UDP datagrams use a simple message format. There are two features found in the datagram: integrity verification, and multiplexing. The former is used to determine the validity of the datagram while the latter is used for transmission of multiple data streams for applications that support this attribute.

The UDP header consists of four fields, each of which has sixteen bits. The use of the fields called "Source port" and "Checksum" is optional when IP version 4 is used, but "Checksum" is required for IP version 6 (Sosinsky, 2009). Table 3.1 shows the structure of a UDP datagram:

Offsets Octet	**0**	**1**	**2**	**3**
0	Source port		Destination port	
4	Length		Checksum	

Table 3.1: Structure of a UDP Datagram (Sosinsky, 2009)

Source port number

A source port number is an optional field which is used to identify the sender’s port or the port number that needs to reply back. In case this field is not used, a value of zero is inserted. In addition, if the source host is a server, the port number is likely to be a common port number, while the port number tends to be an ephemeral port number if the source host is a client (Forouzan, 2000).

Destination port number

A destination port number is used to indicate the receiver’s port. If the source host is a server the port number is likely to be used by common protocols that are in the range of 0 to 1023 (Sosinsky, 2009). If the source host is a client, the port number tends to be an ephemeral port number. In addition, this field is required in both IPv4 and IPv6 (Sosinsky, 2009).

Length

This field specifies the length in octets of the user datagram including the UDP data and UDP header. The practical limit for the data length in IP version 4 is 65,507 Bytes (Sosinsky, 2009). In IP version 6, however, it is possible to have UDP packets of a size greater than this number. In addition, if the UDP data and UDP header is greater than 65,535 Bytes, the length field is set to zero (Sosinsky, 2009).

Checksum

This field is used to check errors in both the header and data. If no checksum is generated, the value is set to all zeros. The checksum is useful when the datagram needs to pass over unreliable links. In addition, this field is optional when IP version 4 is used, but it is required for IP version 6 (Loshin, 2003).

3.6.1.2 Comparison of TCP and UDP

TCP stands for Transmission Control Protocol. It is one of the members of the Internet protocol suite that provides connection-oriented, end-to-end, and reliable inter-process communication between hosts (Loshin, 2003). The following list describes some of the advantages of TCP:

- **Reliable:** TCP is designed to be resilient even when data received is damaged, out of order, or it receives data more than one. In the case of multiple time-outs, the connection is dropped, or if data get lost during transmission, the server will request the lost part again.
- **Ordered:** Unlike UDP that imposes a structure on their data, TCP utilizes a "byte stream service" which allows data to be delivered in order. For example, if two data segments are sent in sequence, the first data segment will reach the destination first. If data segments arrive in the wrong order, it uses a buffer to wait until all data are properly delivered.
- **Heavyweight:** TCP is a reliable protocol, which uses three packets to set up a socket connection before data are transmitted (known as a three-way handshake). The steps involved in a three-way handshake are: synchronize, synchronize – acknowledge, and acknowledge.
- **Streaming:** TCP is a stream-oriented protocol, which allows data to be read in the form of a byte stream. TCP uses a reliable stream known as a "byte stream service" to deliver data to the destination, and it ensures that all bytes received will be identical and in the correct order (Loshin, 2003).

In terms of UDP, it is a connectionless protocol, which does not require a dedicated end-to-end connection. Communication is achieved by sending data from the source to the destination regardless of the state of the receiver. Nevertheless, the main advantage of this protocol is that it is suitable for real-time applications such as video conferencing, VoIP, and online games.

UDP is also suitable for applications in which loss of the packet is not important. For applications that need a high degree of reliability, such as mission-critical applications, a protocol such as TCP may be considered. UDP-based applications do not normally have good congestion avoidance and control mechanisms. In addition, these applications frequently give an inelastic bandwidth load which can consume the available bandwidth on the network, especially on the Internet (Rahman, Saha, & Hasan, 2012). The following list explains the features of UDP:

- **Unreliable:** There is no mechanism of acknowledgement, retransmission, and time-out. When data are sent, it cannot be checked whether it will reach its destination, or whether data was lost during the process.
- **Not ordered:** If two data segments are sent to the same destination in sequence, the order in which they arrive can be different.

- **Datagrams:** UDP provides no congestion control mechanisms. This protocol allows applications to send UDP datagrams at the same rate of the link interface, which is normally much higher than the available path capacity. As a result, it can cause congestion collapse at choke points in the network.
- **Lightweight:** UDP was designed as a “lightweight” protocol. It does not have the ability to track connections between hosts, and it does not allow data to be delivered in order. Table 3.2 summarizes the attributes of TCP and UDP:

Attribute	TCP	UDP
Reliability	Reliable	Unreliable
Connection Management	Connection-oriented	Connectionless
Transmission	Byte-oriented	Message-oriented
Fault Tolerance	No	No
Congestion Control	Yes	No
Flow Control	Yes	No
Security	Yes	Yes
Data Delivery	Strictly Ordered	Unordered

Table 3.2: Comparison Between TCP and UDP

3.6.2 Internet Control Message Protocol

This section describes the Internet Control Message Protocol (ICMP) which is the protocol used during a UDP flood attack process. Therefore, an understanding of the attributes of this protocol is required. ICMP is part of the Internet protocol suite that provides feedback and error messages during network operations. These messages help the network administrators to analyze the current state of the network as well as diagnosing network connectivity problems (Bruce, 2011). The following section discusses the ICMP destination unreachable, which is the type of ICMP message used during a UDP flood attack.

3.6.2.1 ICMP Destination Unreachable

Destination unreachable (Type 3) is generated by the host to inform the client that the destination is unknown. For example, when the router is missing the information that allows a packet to be forwarded, the source host will receive the ICMP message Type 3 (destination unreachable). This is a common occurrence when the router does not have the destination address in a routing table. There are several ICMP codes for destination unreachable but the most common type during a UDP flood attack is code three, which is a destination port unreachable message (Bruce, 2011). During a UDP flood attack, when the attacker sends UDP packets to the destination ports on the victim computer, if one of these ports is closed, the victim computer will respond with ICMP destination unreachable messages back to the attacker computer (Singh & Juneja, 2010).

3.7 Chapter Summary

This chapter described the details of UDP flood attacks, which is the type of DDoS attack used in this study. This attack occurs when an attacker crafts numerous UDP packets to random destination ports on a victim computer, and the victim, on receipt of UDP packets, has to respond with ICMP packets if one of these ports is closed. This chapter also explained the technique called IP spoofing, which is the technique used by the attacker to hide his/her IP address, and to make the attack packets appear to come from different source locations. The method of UDP flood attacks using Zombies was also discussed, which is a technique used by an attacker to multiply the attacks. This chapter also described the functionality of UDP flood attack tools, which are divided into two categories: agent-based and IRC-based. The last section covered a description of the two protocols involved in UDP flood attacks, namely, UDP and ICMP.

The next chapter covers the methodologies and techniques employed in this study.

CHAPTER 4

METHODOLOGY

This chapter covers the methodologies and techniques employed in this study. Section 4.1 describes the hypotheses used in this research. Section 4.2 explains the three broad types of methodologies, then focuses on the quantitative approach, which is the methodology used in this study. This section also covers the evaluation methods that can be used to evaluate the impact of DDoS attacks on a network. Section 4.3 covers two types of data collection processes, the literature review process, and the experimental data gathering process. A brief discussion in regard to the analysis techniques that were used to carry out this study is also provided.

4.1 Research Hypothesis

Hypothesis is the process which gives a clear scope of the elements presented in the research, and it was used as the key driver that forms the experiments in order to produce the result of this study. Following are the research hypotheses that used to manage the research activities and tasks:

- It is expected that the performance of the webserver will reduce during the UDP flood attack.
- It is expected that CPU, RAM, and Disk utilizations on Windows Server 2012 and Linux Ubuntu 13 will increase during the UDP flood attack.
- It is expected that different UDP packet sizes (attack packet) will increase the packet loss and the delay value.
- It is expected that different packet rates (attack packet) will increase the packet loss and the delay value.

4.2 Method Used for Study

There are three broad types of methodologies used to manage a research study: qualitative research, quantitative research, and mixed method research (Case & Light, 2011). According to Daniel Muijs, quantitative research involves gathering numerical data and using mathematically-based methods to explain phenomena or research questions (Muijs, 2011). In addition, there are two main types of quantitative research designs:

experimental designs and non-experimental designs. The former involves the test and numerical statistics, while the latter includes the survey-based information (Charvat, 2003).

The main methodology used in this study was based on the quantitative approach in which all data used in the analysis process were from the results of the experiment. There were three main processes used to evaluate the impact of a UDP flood attack on the network. The first process was to identify the performance of the network before the UDP flood attack. The second process was to identify the performance of the network during the attack. The final process was to evaluate the performance of the network after using DDoS defenses. The details of the evaluation process will be explained in Section 4.3.2.2.

4.2.1 DDoS Attack Evaluation Methods

A DDoS attack evaluation method is the methodology used to study the impact of a DDoS attack on a computer network. The most common DDoS evaluation methods are broken into three categories: Simulation, Testbed, and Theory (Mirkovic, Sonia, Reiher, & Thomas, 2009). The following section describes the advantages and disadvantages of each method. In addition, Testbed was an evaluation method used in this study.

4.2.1.1 Simulation

Simulation is an attempt to model a hypothetical situation on a computer so that it can be studied to see how the system works. This method is very popular for analyzing network performance questions. Most simulation software provides simple router models but some of them (such as OPNET) provide several models of switches, routers, and servers based on vendor specifications (Mirkovic, Sonia, et al., 2009). Nevertheless, one of the disadvantages of a simulation method is that it may unintentionally increase CPU and RAM utilization of the testing computer.

4.2.1.2 Testbed

Testbed is one of the evaluation methods used in this study. Testbed gives the most realistic environment, which allows researchers to set up DDoS experiments using real traffic and attack methods. Testbed gives access to a desired number of computers, located at a central facility and isolated from the Internet, and it also offers realistic parameters such as delays, packet loss, and user throughput (Mirkovic, Sonia, et al., 2009). Testbed provides a more realistic evaluation environment than simulation for many reasons, including:

- Real operating system, applications, and hardware are used in testing.
- DDoS and legitimate traffic can be generated in several ways.
- Many router choices exist in Testbed, including Cisco and Juniper routers which allow realistic forwarding behavior.

4.2.1.3 Theory

Theory method is suitable for answering questions about situations that can be accurately represented by existing models. In the DDoS experiment, theory may be useful to evaluate the robustness of a given defense to cheating or a direct attack. Both require some guesswork on the attacker’s part, the success of which can be evaluated via theory.

This approach is also useful in answering specific questions about a defence’s scalability, cost and delay. However, these attributes depend on the defence’s design, and whether or not parts of it can be accurately represented by simple models. Generally, the theory approach is not suitable for effectiveness evaluation (Mirkovic, Sonia, et al., 2009).

4.3 Data Collection Process

This section describes the data collection process, which was used to obtain the data for this study. There were two types of data collection methods in this research: the literature review and an experimental data gathering process. The former provides both the knowledge and information needed for the research. Searching for the literature was mainly based on journals, conference papers, books and other credible sources from the Internet. In terms of the experimental data gathering process, it was carried out using multiple tests in the computer laboratory.

4.3.1 Literature Review Process

The literature review is the initial process that enhances the researcher intellectually while reading and analyzing other’s relevant works. It also provides the knowledge base, background, and information needed for this study. In this research, all literature was gathered from different credible resources such as academic databases, academic peer-reviewed journals, library references, and appropriate credible association websites. Table 4.1 is an example of the credible resources, where information for this research was retrieved.

Resources	From
Academic Database	IEEEXplore, ACM, and EBSCO HOST
Books	Unitec library, and Google Book
Web Search Engine	Google scholar

Table 4.1: Credible Resources

The resources mentioned in Table 4.1 led to research papers relevant to the study, as they are well-known and credible for information technology-based research. After the literature is reviewed and critically analyzed, it leads to the next process which is information gathering from the testbed. The following sections describe the experimental data gathering process, which explains how the data were collected and analyzed.

4.3.2 Experimental Data Gathering Process

The main resource for data gathering for this research was the collection of data from the tests. To achieve this, a network was set up in the computer laboratory, and numerous tests were run in order to get accurate results. In this process, there were five different types of data used to measure the impact of a UDP flood attack on the network. They were CPU utilization, round-trip time, packet loss, jitter, and user throughputs, all of which were collected by using appropriate tools. Table 4.2 shows the DDoS impact metrics and the tools used in the data gathering process. In addition, the process of collecting this data will be explained in the next section.

DDoS Impact Metrics	Tools Used For Collecting Data
CPU utilization	Microsoft Resource Monitor
Round-trip time	TCPing
Packet loss	Iperf
Jitter	Iperf
User throughput	Iperf

Table 4.2: DDoS Impact Metrics and Tools Used For Collecting Data

4.3.2.1 DDoS Impact Metrics

This section describes the performance metrics that were used in the analysis process (Chapter 6 and Chapter 7). The following is a list of the five common parameters that have been used by many researchers to evaluate the impact of DDoS attacks on networks (Mirkovic, Sonia, et al., 2009).

CPU Utilization

The primary tool used to collect the CPU utilization was Microsoft Resource Monitor. In this study, the CPU usage was collected in three phases: before the attack, during the attack, and after using solutions. In terms of the CPU utilization before the attack, the CPU usage was recorded on the victim computer for 5 minutes (30 runs) without any other running process. Regarding CPU utilization after the attack, CPU monitoring software was run before launching the UDP flood attack. By doing this, the impact of the attack can be seen at the beginning. Figure 4.2 illustrates the CPU utilization output from Microsoft Resource Monitor.

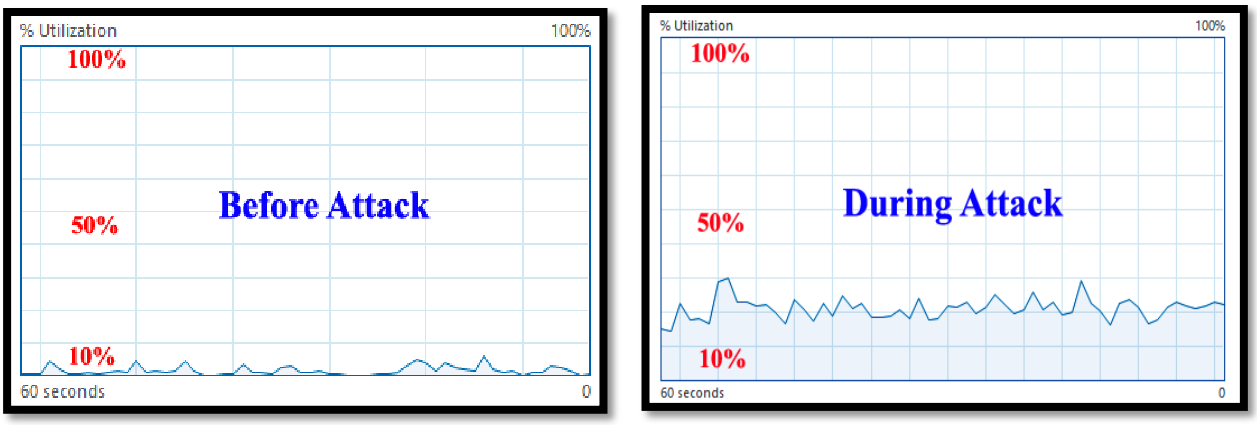
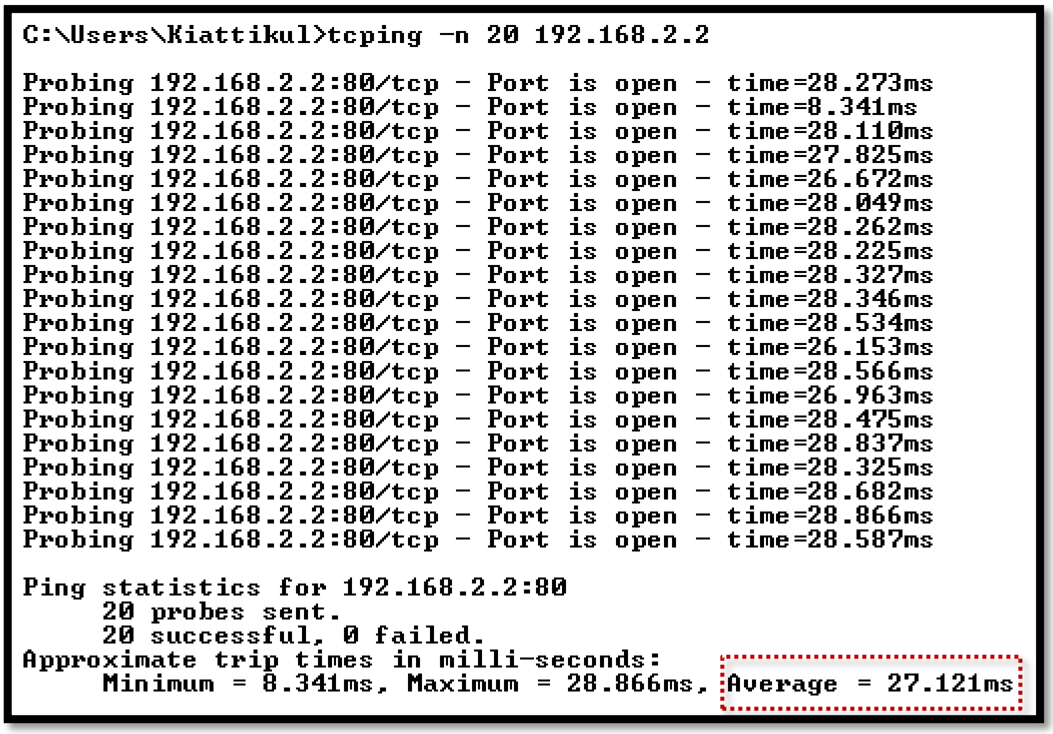


Figure 4.1: Example of CPU Utilization Output from Microsoft Resource Monitor

Round-trip Time

The primary tool used to collect the RTT was TCPing ("Ping over a TCP Connection", 2013). It was done in order to analyze the delay between a legitimate computer and the webserver. RTT was collected in three phases: before the attack, during the attack, and after using the solutions. In each scenario, it was run 30 times for 5 minutes. In terms of the RTT before the attack, the RTT was collected during the normal network operation and without any other running process. Regarding the RTT after the attack, the RTT was collected during the UDP flood attack. Figure 4.3 illustrates an example of round-trip time output (1 run) using TCPing. The details of this tool are also explained in Section 5.3.3.

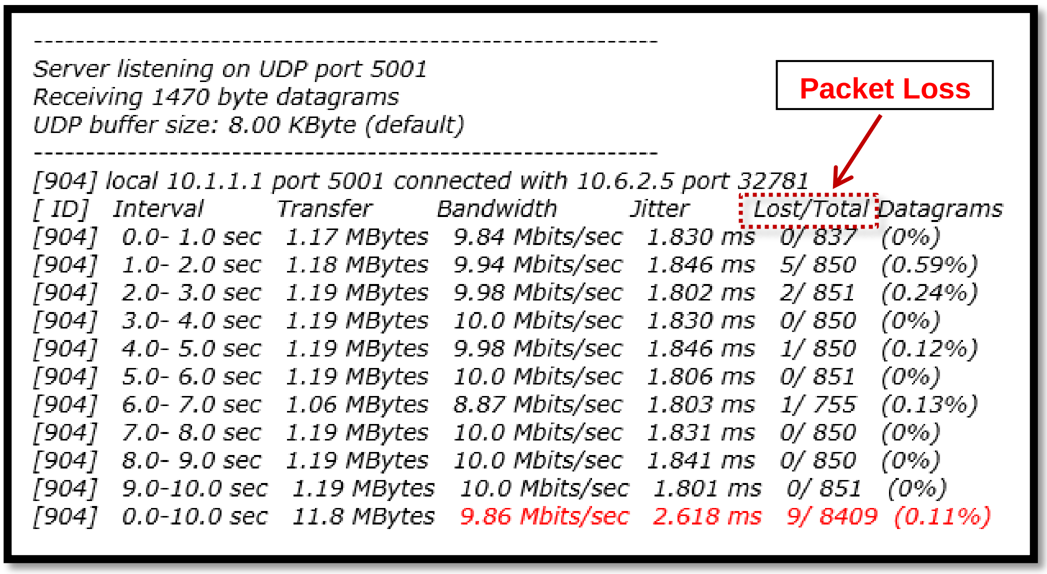


```
C:\Users\Kiattikul>tcping -n 20 192.168.2.2

Probing 192.168.2.2:80/tcp - Port is open - time=28.273ms
Probing 192.168.2.2:80/tcp - Port is open - time=8.341ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.110ms
Probing 192.168.2.2:80/tcp - Port is open - time=27.825ms
Probing 192.168.2.2:80/tcp - Port is open - time=26.672ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.049ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.262ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.225ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.327ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.346ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.534ms
Probing 192.168.2.2:80/tcp - Port is open - time=26.153ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.566ms
Probing 192.168.2.2:80/tcp - Port is open - time=26.963ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.475ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.837ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.325ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.682ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.866ms
Probing 192.168.2.2:80/tcp - Port is open - time=28.587ms

Ping statistics for 192.168.2.2:80
     20 probes sent.
     20 successful, 0 failed.
Approximate trip times in milli-seconds:
     Minimum = 8.341ms, Maximum = 28.866ms, Average = 27.121ms
```

Figure 4.2: Example of Round- trip Time Output from TCPing

Packet Loss

The primary tool used to collect packet loss values was Iperf (Openmaniak, 2009). It was done in order to analyze the relationship between the attack packet size and legitimate packet loss value. The sizes of attack packets were 64, 128, 256, 512, 1024, and 1280 Bytes per packet. These sizes include the maximum and minimum frame sizes permitted by the Ethernet standard (Bradner & McQuaid, 1999). Packet loss value was collected in three phases: before the attack, during the attack, and after using the solutions. For each scenario, it was run 30 times to calculate an average and standard deviation. Figure 4.4 illustrates an example of packet loss values using Iperf. The detail of this tool is also explained in Section 5.3.1.

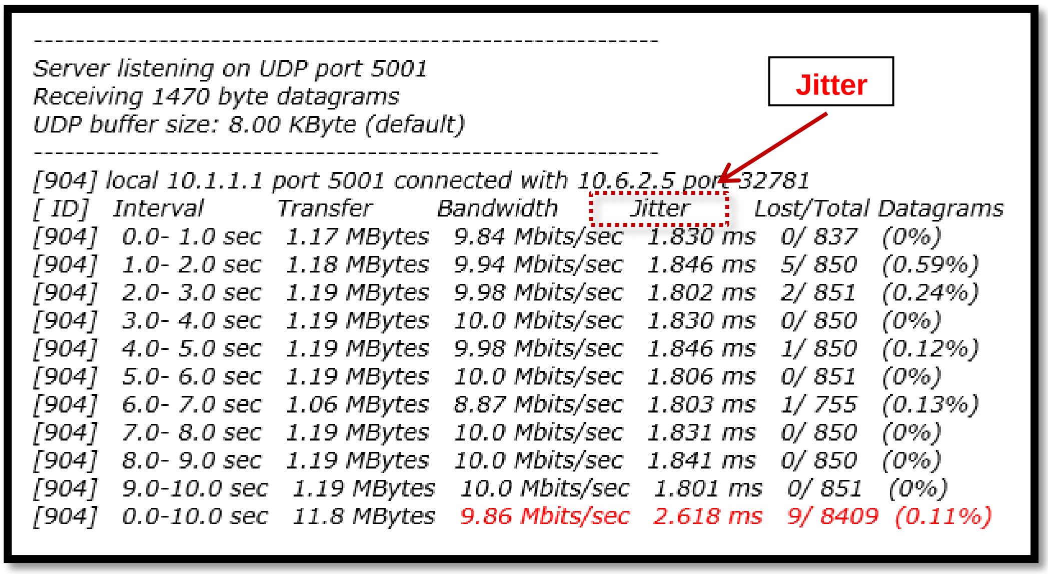


```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[904] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 32781
[ ID]   Interval       Transfer      Bandwidth       Jitter    Lost/Total Datagrams
[904]   0.0- 1.0 sec   1.17 MBytes   9.84 Mbits/sec  1.830 ms   0/ 837   (0%)
[904]   1.0- 2.0 sec   1.18 MBytes   9.94 Mbits/sec  1.846 ms   5/ 850   (0.59%)
[904]   2.0- 3.0 sec   1.19 MBytes   9.98 Mbits/sec  1.802 ms   2/ 851   (0.24%)
[904]   3.0- 4.0 sec   1.19 MBytes   10.0 Mbits/sec  1.830 ms   0/ 850   (0%)
[904]   4.0- 5.0 sec   1.19 MBytes   9.98 Mbits/sec  1.846 ms   1/ 850   (0.12%)
[904]   5.0- 6.0 sec   1.19 MBytes   10.0 Mbits/sec  1.806 ms   0/ 851   (0%)
[904]   6.0- 7.0 sec   1.06 MBytes   8.87 Mbits/sec  1.803 ms   1/ 755   (0.13%)
[904]   7.0- 8.0 sec   1.19 MBytes   10.0 Mbits/sec  1.831 ms   0/ 850   (0%)
[904]   8.0- 9.0 sec   1.19 MBytes   10.0 Mbits/sec  1.841 ms   0/ 850   (0%)
[904]   9.0-10.0 sec   1.19 MBytes   10.0 Mbits/sec  1.801 ms   0/ 851   (0%)
[904]   0.0-10.0 sec   11.8 MBytes   9.86 Mbits/sec  2.618 ms   9/ 8409  (0.11%)
```

Figure 4.3: Example of Packet Loss Output from Iperf

Jitter

The primary tool used to collect jitter values was Iperf. The test was done in order to analyze the legitimate delay variation when using the different attack packet sizes. The sizes of attack packets were 64, 128, 256, 512, 1024, and 1280 Bytes per packet, each of which was run for 30 times to calculate the maximal, minimum and average values. Figure 4.5 illustrates an example of jitter using Iperf.

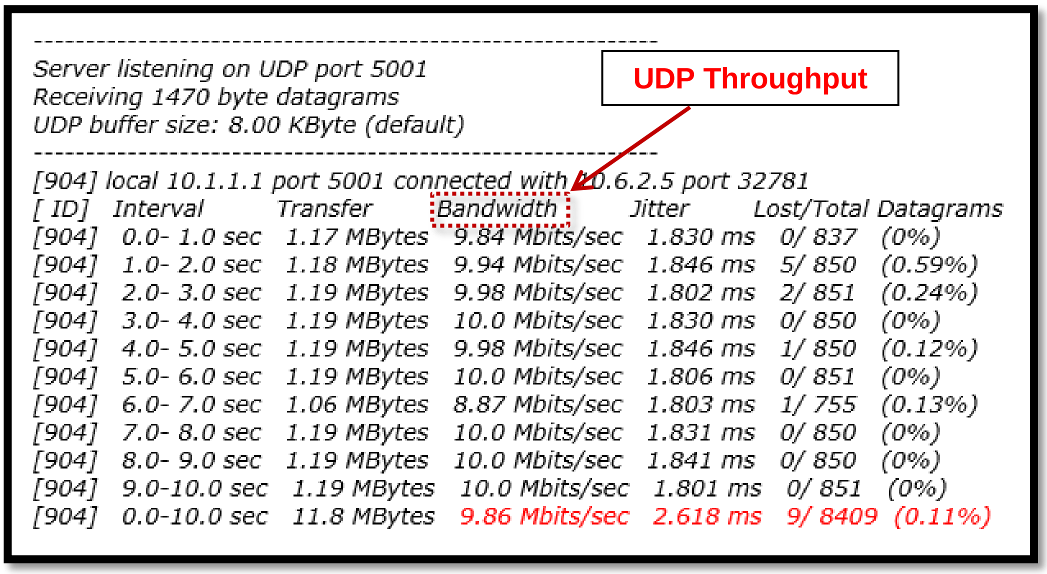


```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[904] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 32781
[ ID]   Interval       Transfer      Bandwidth       Jitter    Lost/Total Datagrams
[904]   0.0- 1.0 sec   1.17 MBytes   9.84 Mbits/sec  1.830 ms   0/ 837   (0%)
[904]   1.0- 2.0 sec   1.18 MBytes   9.94 Mbits/sec  1.846 ms   5/ 850   (0.59%)
[904]   2.0- 3.0 sec   1.19 MBytes   9.98 Mbits/sec  1.802 ms   2/ 851   (0.24%)
[904]   3.0- 4.0 sec   1.19 MBytes   10.0 Mbits/sec  1.830 ms   0/ 850   (0%)
[904]   4.0- 5.0 sec   1.19 MBytes   9.98 Mbits/sec  1.846 ms   1/ 850   (0.12%)
[904]   5.0- 6.0 sec   1.19 MBytes   10.0 Mbits/sec  1.806 ms   0/ 851   (0%)
[904]   6.0- 7.0 sec   1.06 MBytes   8.87 Mbits/sec  1.803 ms   1/ 755   (0.13%)
[904]   7.0- 8.0 sec   1.19 MBytes   10.0 Mbits/sec  1.831 ms   0/ 850   (0%)
[904]   8.0- 9.0 sec   1.19 MBytes   10.0 Mbits/sec  1.841 ms   0/ 850   (0%)
[904]   9.0-10.0 sec   1.19 MBytes   10.0 Mbits/sec  1.801 ms   0/ 851   (0%)
[904]   0.0-10.0 sec   11.8 MBytes   9.86 Mbits/sec  2.618 ms   9/ 8409  (0.11%)
```

Figure 4.4: Example of Jitter Output from Iperf

User Throughput

The primary tool used to collect the throughput values was Iperf. It was used in order to analyze the number of user throughputs between the client and server. There were two types of user throughputs in this study, namely, TCP and UDP. Both of them were collected in three phases: before the attack, during the attack, and after using the solutions. For each scenario, it was run 30 times to calculate an average and standard deviation. Two computers needed to have Iperf installed; one was the victim computer, and the other the monitoring computer (legitimate computer). Figure 4.6 illustrates an example of UDP throughputs using Iperf.



```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[904] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 32781
[ ID]  Interval       Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[904]  0.0- 1.0 sec  1.17 MBytes  9.84 Mbits/sec  1.830 ms  0/ 837  (0%)
[904]  1.0- 2.0 sec  1.18 MBytes  9.94 Mbits/sec  1.846 ms  5/ 850  (0.59%)
[904]  2.0- 3.0 sec  1.19 MBytes  9.98 Mbits/sec  1.802 ms  2/ 851  (0.24%)
[904]  3.0- 4.0 sec  1.19 MBytes  10.0 Mbits/sec  1.830 ms  0/ 850  (0%)
[904]  4.0- 5.0 sec  1.19 MBytes  9.98 Mbits/sec  1.846 ms  1/ 850  (0.12%)
[904]  5.0- 6.0 sec  1.19 MBytes  10.0 Mbits/sec  1.806 ms  0/ 851  (0%)
[904]  6.0- 7.0 sec  1.06 MBytes  8.87 Mbits/sec  1.803 ms  1/ 755  (0.13%)
[904]  7.0- 8.0 sec  1.19 MBytes  10.0 Mbits/sec  1.831 ms  0/ 850  (0%)
[904]  8.0- 9.0 sec  1.19 MBytes  10.0 Mbits/sec  1.841 ms  0/ 850  (0%)
[904]  9.0-10.0 sec  1.19 MBytes  10.0 Mbits/sec  1.801 ms  0/ 851  (0%)
[904]  0.0-10.0 sec  11.8 MBytes  9.86 Mbits/sec  2.618 ms  9/ 8409  (0.11%)
```

Figure 4.5: Example of UDP Throughputs Output from Iperf

4.3.2.2 UDP Flood Attack Evaluation Process

There were 3 main processes were used to evaluate the impact of a UDP flood attack on the network. The first process was to identify the performance of the network before a UDP flood attack. The second process was to identify the performance of the network during the attack and, the final process was to evaluate the performance of the network after using the DDoS defenses. The following sections explain the steps involved in each process.

Process of Generating Legitimate Traffic

In order to identify the performance of the network before a UDP flood attack, legitimate traffic was required. To achieve this, software called Webserver Stress Tool was used (Webstress, 2014). This software can simulate legitimate users who want to access the webserver, and therefore we can identify legitimate traffic. In this study, the number of legitimate users was 10 users, and the test was run 30 times using Window Server 2012 and Linux Ubuntu 13. This generated traffic at approximately 4.4 packets per second. During this time, CPU utilization, delay, jitter, packet loss, and user throughput values were collected, and used to compare with the same parameters during the attack. In addition, the detail of Webserver Stress Tool is explained in Section 5.3.4.

Process of Generating Attack Traffic

The primary tool used to generate the attack traffic was Hping3 (Hping, 2014). In this study, an attack rate of 13000 packets per second and a packet size of 512 Bytes per packet were used to attack the targeted network. This generated attack traffic at approximately 50.7 Mbps. These were the maximum values that the victim computer could withstand, and we would not have been able to measure traffic (e.g., RTT) if the packet rate and packet size had been higher than those values. During the UDP flood attack, CPU utilization, delay, packet loss, and user throughput values were collected, and used to compare with legitimate traffic. In addition, the detail of Hping3 is also explained in Section 5.3.5.

Process of Evaluating Defenses

Six defense mechanisms were used against the UDP flood attack, namely, Access Control Lists, Threshold Limit, Hybrid Defense, IP Verify, Network Load Balancing, and the Software Firewall. After implementing each solution, a UDP flood attack was launched again. The efficiency of defense was evaluated by comparing the number of CPU utilization, delay, jitter, packet loss, and user throughput values before and after using the solutions. In addition, the detail of defense mechanisms and the network set up will be explained in Section 5.2.

4.4 Chapter Summary

This chapter covered research hypotheses, methodology of study, and data collection process. The pre-defined hypotheses showed the scope or the boundary of this research, while a quantitative research method was selected as the main methodology and used to manage the research from beginning to end. A testbed was set up to gather quantitative data. This chapter also detailed the most common DDoS evaluation methods, which are simulation, testbed, and theory. In addition, the testbed method was selected as an evaluation method since it provided a more realistic evaluation environment compared to the simulation and theory methods. The end of this chapter presented the procedures of data collection, and the steps involved in the process of both the literature review, and the experiment for this research, with examples of graphs also presented.

The next chapter covers the testbed set-up for a UDP flood attack and defenses.

CHAPTER 5

EXPERIMENTAL DESIGN

This chapter describes the experimental network design used in this research to obtain the results. Section 5.1 covers the experimental set-up of a UDP flood attack, including the hardware and software specifications. Section 5.2 covers the experimental set-up of defenses, which are: Access Control Lists, Threshold Limit, Hybrid Defense, IP Verify, Network Load Balancing, and Software Firewall. Section 5.3 presents the five main network measurement tools used in this research.

5.1 Experimental Set-up of a UDP Flood Attack

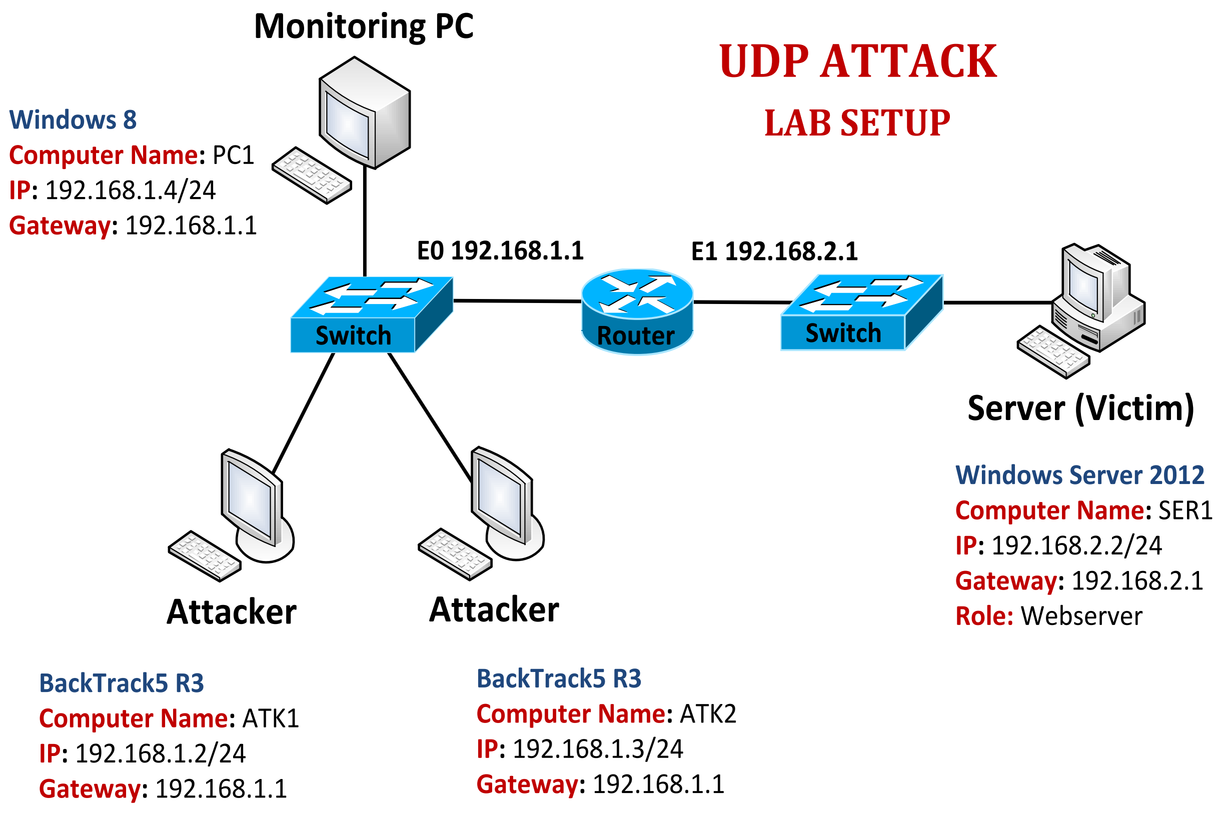


Figure 5.1: UDP Flood Attack Testbed Set-up

Figure 5.1 illustrates the experimental network designed for a UDP flood attack. The proposed network testbed was designated to simulate a real live implementation of a DDoS attack. To achieve this, the network was set up through a direct connection using a standard category 5e cabling between workstations. The router was used to separate the two networks, and to monitor incoming and outgoing traffic between the networks. There were four types of workstations in the testbed: Two workstations would act as attackers, one would act as a victim, and another was used as a monitoring machine. The purpose of this network design is to maintain consistency with previous work and similar research carried out in the past (Subramani, 2011).

The workstations where the attackers perform had BackTrack 5 R3 installed. It was a Linux-based penetration testing arsenal that provided a comprehensive and large collection of security-related tools including DDoS attack tools (Backtrack, 2013). The victim machine had Windows Server 2012 installed with the latest version of Microsoft Webserver (IIS8). The monitoring PC, in which Windows 8 was installed, was where the different varieties of monitoring tools were installed to gather data and perform the network testing analysis. Software installed on the monitoring machine was TCPing, Iperf, Wireshark, and, Webserver Stress Tool.

DDoS is comprised of multiple computers that take part to flood a targeted machine. To achieve this, two attack machines were used. One machine would launch UDP flood attack using a valid IP address (the original attacker's IP address), and the other machine, which was a Zombie machine, would attack the victim with massive spoofed IP addresses (random IP addresses).

In this study, an attack rate of 13000 packets per second and a packet size of 512 Bytes per packet were used to attack the targeted network. This generated the attack traffic at approximately 50.7 Mbps. These were the maximum values the server could withstand, and we would not have been able to measure traffic (e.g., RTT) if the attack packet rate and packet size had been higher than these values. During the UDP flood attack, CPU utilization, delay, jitter, packet loss, and user throughput values were collected, and used to compare with legitimate traffic. The following command was executed from a Linux command prompt on the attacker machines to craft the UDP packets, and send them to the victim machine:

- Hping3 - -rand-source -i u13000 -UDP -p ++5000 -d 512 192.168.2.2

According to the command, "rand-source" was a command for crafting random source IP addresses. This command can be changed to the original attacker's IP address by typing his/her IP address instead. "i u 13000" was a command for sending attack packets at 13,000 packets per second. "UDP" was a command for crafting a UDP packet. "P ++5000" was a command for sending the UDP packets to the destination ports number starting from 5,000 and increases the number every packet. “D 512” was a command for using an attack packet size of 512 Bytes per packet. “192.168.2.2” was an IP address of a targeted computer. The details of the UDP flood attack evaluation process was explained in Section 4.3.2.2.

5.1.1 Hardware and Software Specification

5.1.1.1 Hardware

In order to be consistent and produce accurate data from this study, the hardware used in all of the experiments was kept identical. The hardware benchmark was comprised of an Intel® Core™ i5 2.80 GHz processor with 8.00 GB RAM for the efficient operation of operating systems; Cisco router 2811 and Cisco switch SG 200 were chosen as the network connection devices. The router was also used for defenses against the attack. Table 5.1 outlines the type and specifications of the hardware involved.

Hardware		**Specifications**
PC	CPU	Intel® Core™ i5 2.80 GHz
	RAM	8.00 GB
	Hard Disk	Western Digital Caviar SE 160 GB
	LAN Card	Intel® PRO/1000 GT Adapter
	Motherboard	Lenovo
	Motherboard Chipset	Intel Q965 Rev. 1
Network Connect Device	Cable	CAT5e
	Switch	Cisco SG 200
	Router	Cisco 2811

Table 5.1: Hardware Specifications

5.1.1.2 Software

In terms of software specification, four operating systems were involved in this study; two of them were Microsoft operating systems and the other two were Linux open source operating systems. Table 5.2 describes the operating systems, roles, and software installed on the system. In addition, the details of software are also explained in Section 5.3.

Operating System	Role	Software installed
Windows Server 2012	Victim	Wireshark, Iperf
Windows 8	Monitoring machine	TCPing, Wireshark, Iperf, Webserver Stress Tool
Linux Ubuntu 13	Victim	Wireshark, Iperf
Linux Backtrack R3	Attacker machine	Hping3

Table 5.2: Software Specifications

5.2 Experimental Set-up of Defenses

This section covers the details of the experimental set-up and configuration of defenses. There were six solutions used against the UDP flood attack in this study. They were Access Control Lists (Section 5.2.1), Threshold Limit (Section 5.2.2), Hybrid Defense (Section 5.2.3), IP Verify (Section 5.2.4), Network Load Balancing (Section 5.2.5), and Software Firewall (Section 5.2.6).

5.2.1 Access Control Lists

Access Control Lists (ACLs) is a network filter utilized by routers to permit and restrict data flows into and out of network interfaces. When an ACL is configured on an interface, the router analyses the data passing through the interface, and compares them to the criteria described in the ACL table. The router can either permit the data to flow or drop it (Cisco, 2013d).

This study aims at applying ACLs on the victim router in order to prevent a specific set of IP addresses. An example of these IP addresses is the private IP addresses. These IP addresses are usually not routable on the Internet, therefore, if traffic comes in with these IP addresses from the public Internet, it must be fraudulent traffic. Figure 5.2 shows the experimental set-up for ACLs defense.

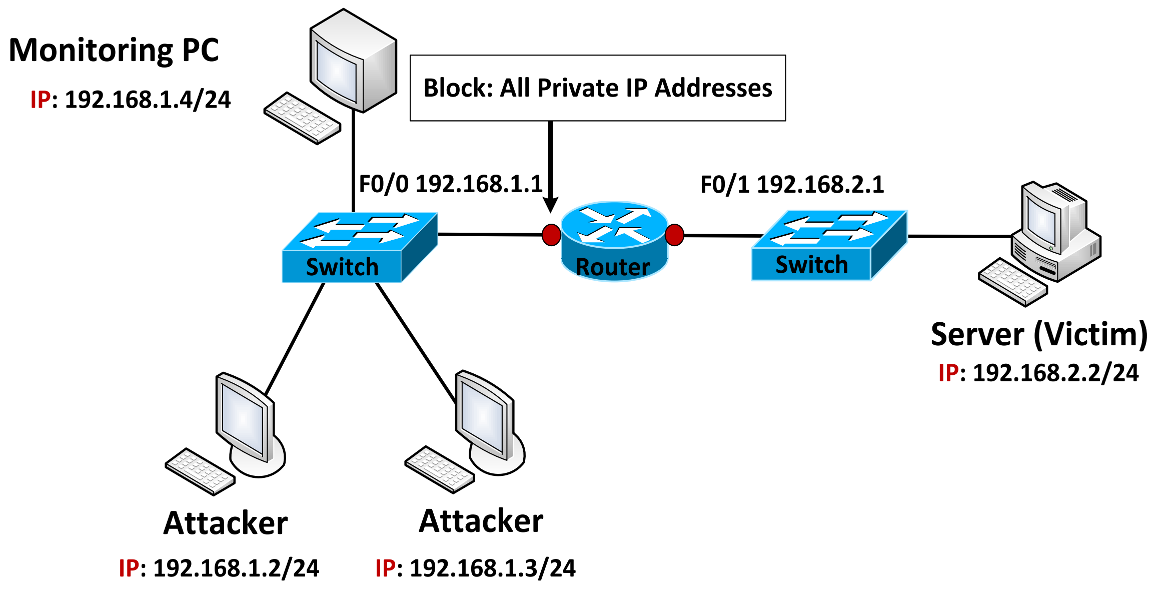


Figure 5.2: Access Control Lists Set-up

According to Figure 5.2, in order to prevent a UDP flood attack on the network, the first step is to block all IP addresses that pose a risk (David, 2007). The example of these IP address is private IP addresses, and other types of shared/special IP addresses. These IP addresses are usually not routable on the Internet, and the attacker can use one of these IP addresses to hide the true source of the attack (IP spoofing). Table 5.2 is a list of IP addresses that need to be blocked:

Risky IP Address	Reason
10.10.0.0	Private IP Address Class A
172.16.0.0	Private IP Address Class B
192.168.0.0	Private IP Address Class C
224.0.0.0	Multicast Address
127.0.0.0	Loop Back Address
169.254.0.0	Automatic Private Internet Protocol Addressing

Table 5.2: List of Risky IP Addresses

All of IP addresses showing in Table 5.2 are either private IP addresses that cannot be used on the Internet or are used for other purposes. For example, “IP 169.254.0.0” is reserved for client computers when they cannot connect to a DHCP server. Therefore, if traffic comes in with this IP address from the public Internet, it must be malicious traffic. In addition, it can be noted that this defense cannot block attacking traffic if the attacker uses IP addresses that are not on the black lists, e.g., spoofed public IP addresses.

In this thesis, the following ACL rules were used to stop the UDP flood attack. These rules were entered into the router's command line interface using the router interface (f0/0), which was the router interface of the victim network.

- Config t
- IP access-list ext ingress-antispoof
- Deny IP 224.0.0.0 31.255.255.255 any
- Deny IP 169.254.0.0 0.0.255.255 any
- Deny IP 172.16.0.0 0.15.255.255 any
- Deny IP 127.0.0.0 0.255.255.255 any
- Deny IP 10.0.0.0 0.255.255.255 any
- Permit IP any any
- Int f0/0
- IP access-group ingress-antispoof IN

The first command "Config t" was used to execute configuration commands from the terminal. The next command "Deny IP 224.0.0.0 31.255.255.255 any" was a set of IP addresses that were blocked. The next command "Int f0/0" was used for specifying the interface of the network that was to be protected from the attack (victim's network). Finally, when the command "IP access-group ingress-antispoof IN" was entered, the router would block the IP addresses defined.

5.2.2 Threshold Limit

Unlike Access Control Lists, the Threshold Limit technique does not aim to block any traffic from the outside of the network (Cisco, 2013b), instead it sets up a rate limit of traffic that the server can withstand. The advantage of this approach is that it allows the network administrator to decide how much traffic should be let inside the network. This traffic rate also depends on the size of the organization, the server's processing capacity, and the traffic it would be able to withstand. Figure 5.3 shows the experimental set-up for the Threshold Limit technique.

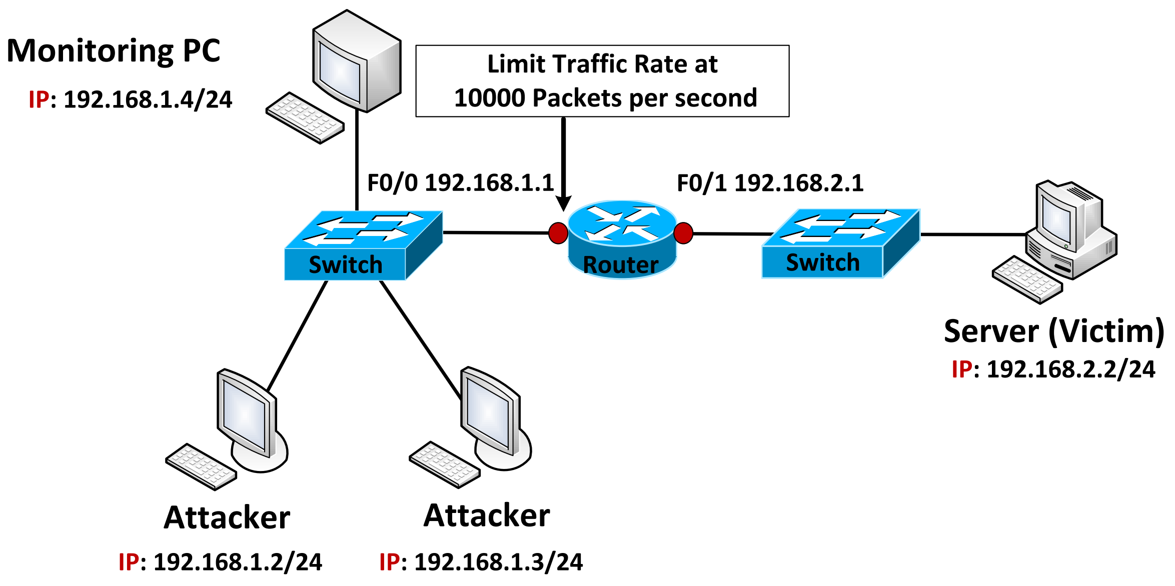


Figure 5.3: Threshold Limit Set-up

In this study, the following command was used for limiting traffic from the attacking network up to the threshold. These commands were entered into the router's command line interface using the router interface (f0/0), which was the router interface of the victim network.

- Config t
- Int f0/0
- Rate-limit input 10000 10000 10000 conform-action transmit exceed-action drop
- Ctrl + C
- Write

The first command "Config t" was used to execute configuration commands from the terminal. On the second line, "Int f0/0" was used to define the interface where the incoming traffic from the outside of the network should be policed. The third line was used to define the amount of traffic that could pass through the network. In addition, the first value of 10000 was defined as Bytes per second, which was the amount of traffic that the server would be able to serve for the clients connected to it (Subramani, 2011). The last value of 10000 was defined as the burst maximum, which was used to maintain a steady flow of packets and handle the traffic fluctuations. After applying this command, the incoming traffic that did not exceed these conditions was transmitted, whereas the rest of it was dropped. Figure 5.4 illustrates the traffic rate after using the Threshold Limit technique.

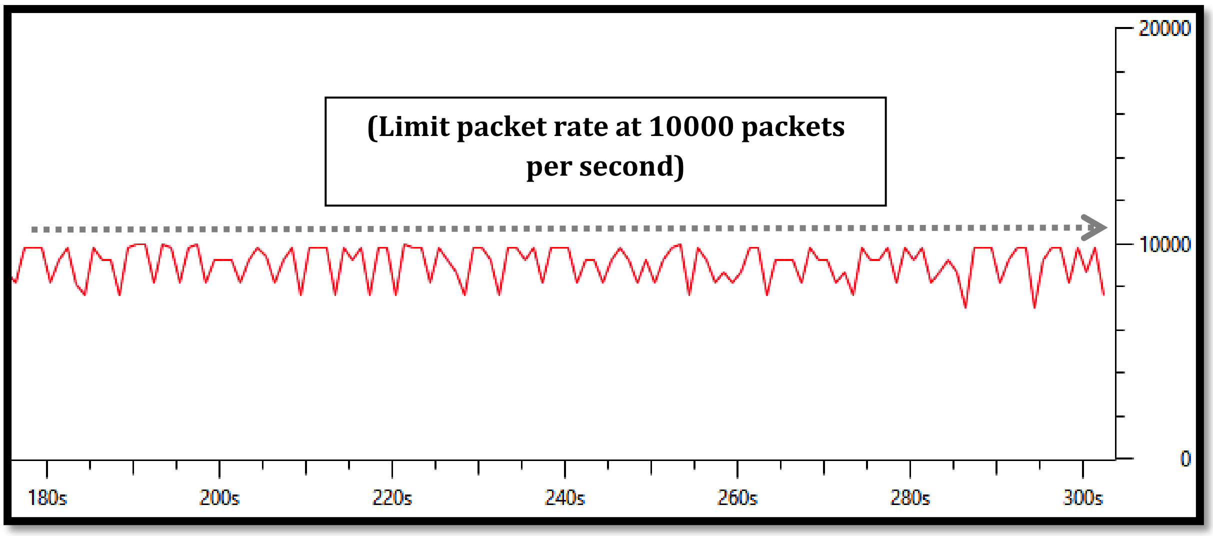


Figure 5.4: Output from Wireshark After Using Threshold Limit Technique

According to Figure 5.5 below, this defense allows all packets to enter the network but limits the traffic rate up to the threshold (up to 10000 packets per second). One of the advantages of this defense is that it has the ability to maintain a steady flow of packets, and the traffic fluctuations. This technique is also suitable for the originations that do not want to block private IP addresses with some reasons. Moreover, since it does not block private IP addresses, it can be implemented inside the network to protect servers from DDoS attacks launched by malicious insiders (Subramani, 2011).

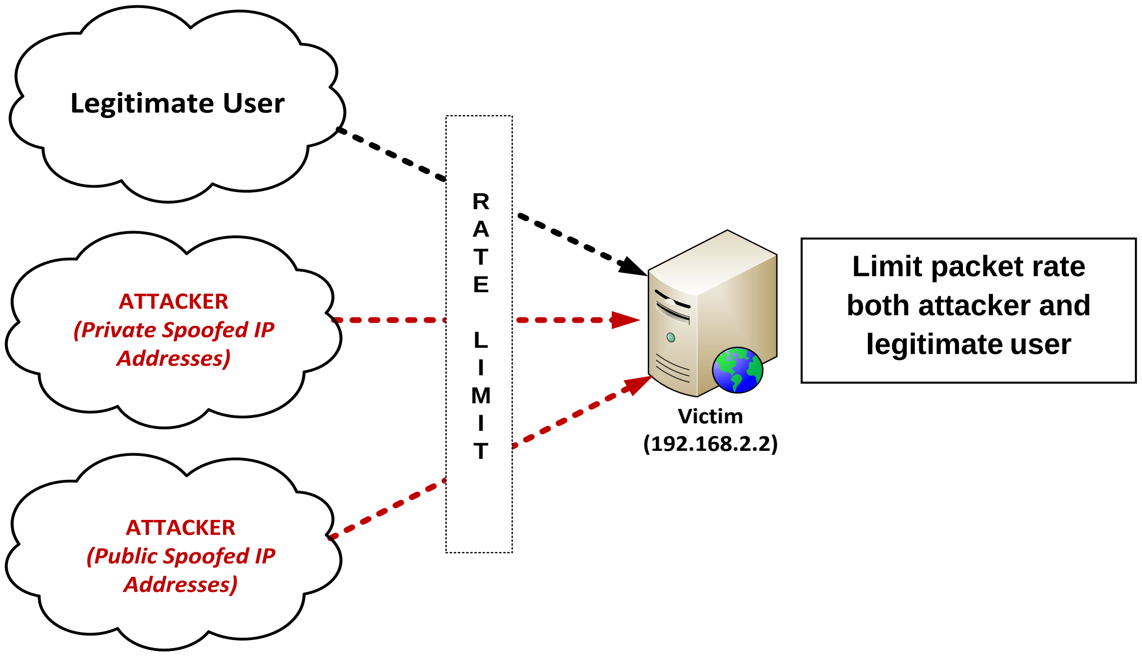

Legitimate User
ATTACKER
(Private Spoofed IP Addresses)
ATTACKER
(Public Spoofed IP Addresses)
RATE LIMIT
Victim
(192.168.2.2)
Limit packet rate both attacker and legitimate user

Figure 5.5: Threshold Limit Defense Strategy

5.2.3 Hybrid Defense

Hybrid Defense was proposed by Subramani (2011) to improve existing DDoS solutions: ACLs and Threshold Limit. Hybrid Defense is the enhanced solution that combines the main advantages of both Access Control Lists and Threshold Limit techniques to govern the traffic flow better. This defense gives the ability to a victim's router to drop the malicious packets and control the incoming traffic rate up to the threshold. Figure 5.6 shows the experimental set-up for Hybrid Defense.

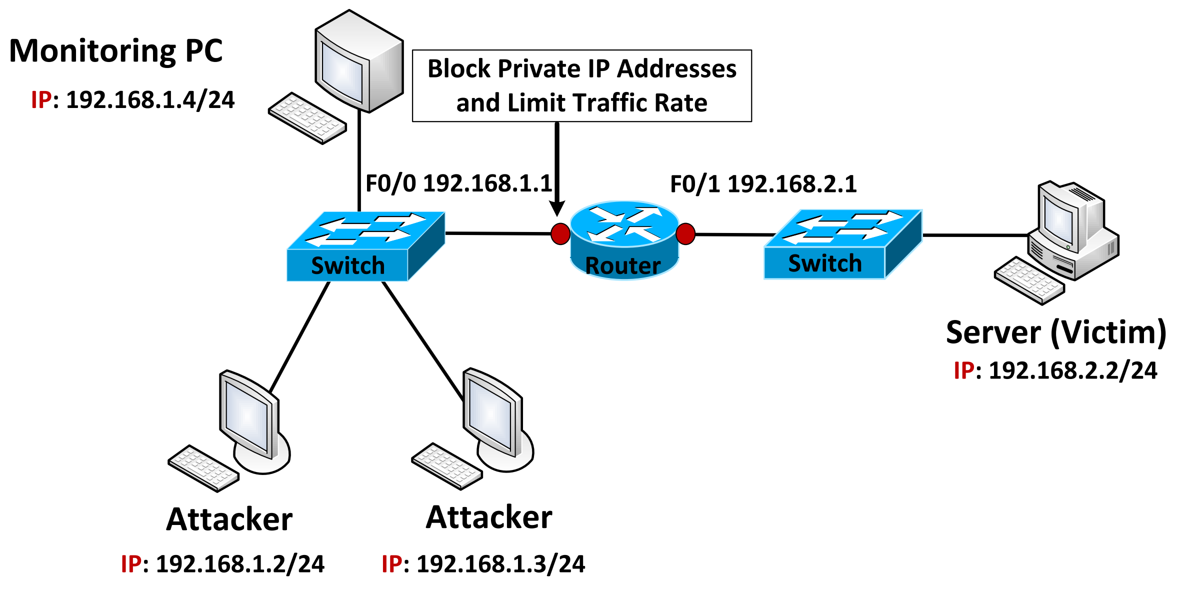


Figure 5.6: Hybrid Defense Set-up

In this thesis, the following command was used in order to block and limit traffic from the attacking network. These commands were entered into the router's command line interface using the router interface (f0/0), which was the router interface of the victim network.

- Config t
- IP access-list ext hybrid-defense
- Deny IP 224.0.0.0 31.255.255.255 any
- Deny IP 169.254.0.0 0.0.255.255 any
- Deny IP 172.16.0.0 0.15.255.255 any
- Deny IP 127.0.0.0 0.255.255.255 any
- Deny IP 10.0.0.0 0.255.255.255 any
- Permit IP any any
- Exit

- Int f0/0
- Rate-limit input 10000 10000 10000 conform-action transmit exceed-action drop
- IP access-group hybrid-defense IN

The first command “Config t” was used to execute configuration commands from the terminal. Since private IP addresses and special IP addresses cannot be used on the Internet, all of the IP addresses described on the third to seventh lines were blocked. On the eleventh command, it allowed the rest of the IP addresses to pass through the network, but it still limited the traffic rate up to 10000 packets per second. These commands were applied on the inference f0/0 which was the router interface of the victim network. The Hybrid Defense strategy is explained in Figure 5.7:

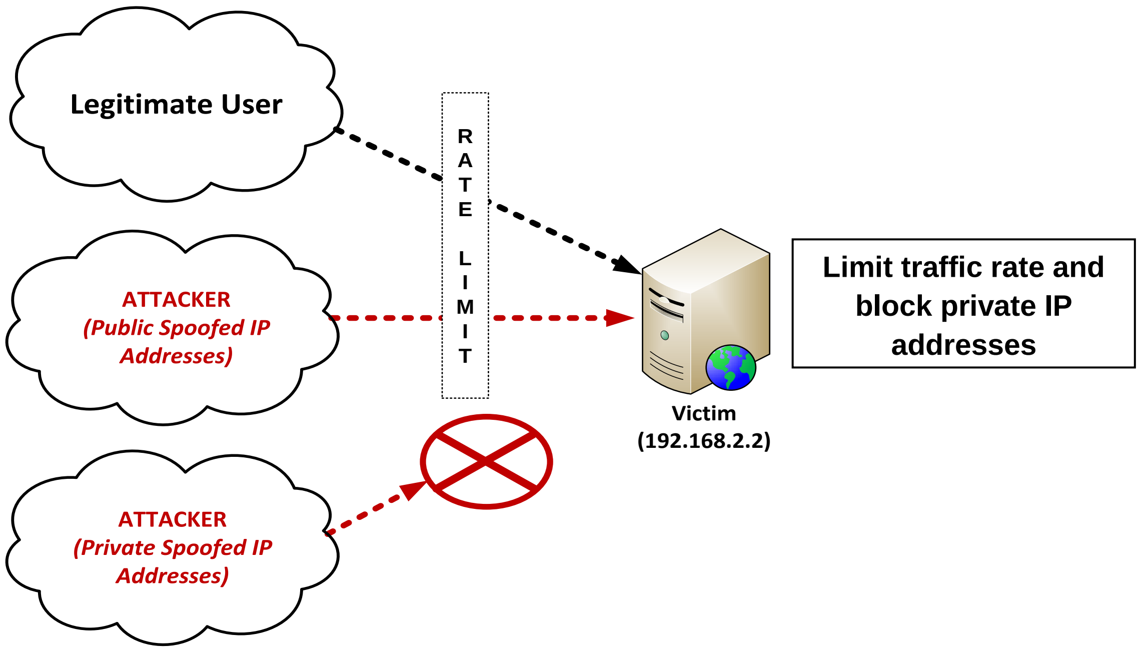

Legitimate User
ATTACKER
(Public Spoofed IP Addresses)
ATTACKER
(Private Spoofed IP Addresses)
RATE LIMIT
Victim
(192.168.2.2)
Limit traffic rate and block private IP addresses

Figure 5.7: Hybrid Defense Strategy

According to Figure 5.7, after implementing Hybrid Defense on the victim’s router, the result from Wireshark installed on the server (victim computer) showed that none of private spoofed IP address could gain access to the webserver. This is because the router had dropped all IP addresses that fell into the black-lists. Another advantage of this defense is that it could also drop a traffic rate that was higher than the threshold (Subramani, 2011).

5.2.4 IP Verify

IP Verify is a technique used in routers for the purpose of protecting a network from a DDoS attack or preventing infinity loops of packets in multicast routing. This security feature gives the ability to the router to verify the reachability of the source IP addresses before they can enter the network. If the source IP address is not valid, the packet is dropped (David, 2007).

The IP Verify technique has two modes: strict mode, and loose mode. In terms of the strict mode, the packet must be received on the same interface that the router will use to send a return packet. One of the disadvantages of the strict mode is that it may drop legitimate traffic if the traffic is coming from different interfaces from those the router has decided on (Cisco, 2013c).

For IP Verify in the loose mode, on the other hand, the source address must appear in the routing table. Each incoming source IP address needs to be tested against this forwarding table, and the packet is discarded if the source address in not reachable via any interface on the router (Cisco, 2013c). Since IP Verify in the strict mode can drop legitimate traffic, the unicast RPF loose mode was selected as a proposed defense. Figure 5.8 shows the experimental set-up for IP Verify:

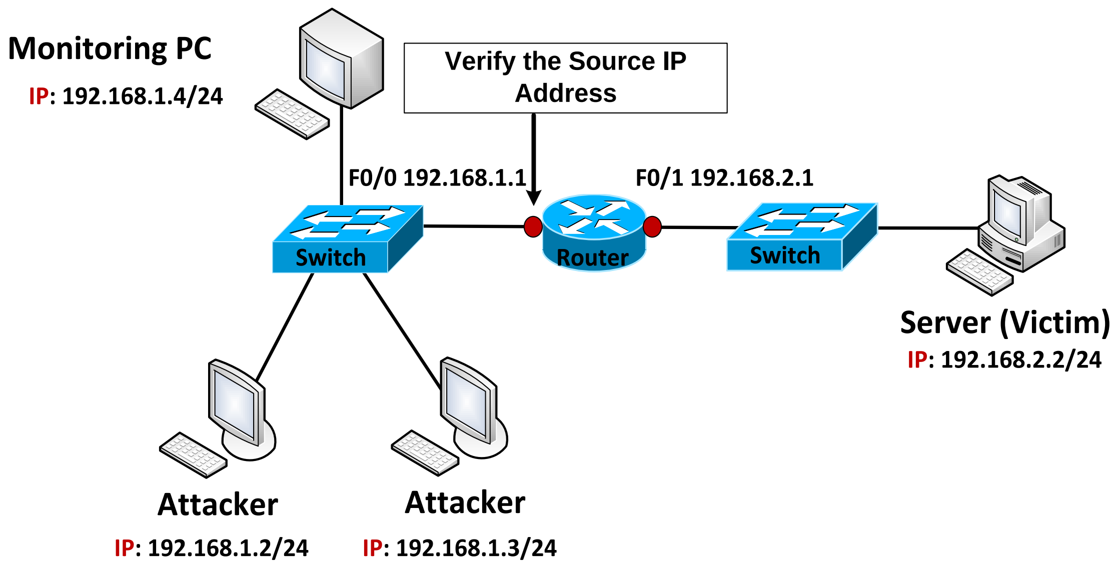


Figure 5.8: IP Verify Set-up

In this thesis, the following commands were used for implementing the IP Verify loose mode on the router. These rules were entered into the router's command line interface using the router interface (f0/0), which was the router interface of the victim network:

- Config t
- IP cef
- Int f0/0
- IP verify unicast source reachable-via any

The first command "Config t" was used to execute configuration commands from the terminal. On the next commands "IP cef" was a command for implementing IP Verify defense on the router. The command "Int f0/0" was used for specifying the interface of the network that needed to be protected from the attack. Finally, when the command "IP verify unicast source reachable-via any" was entered, the router would verify the reachability of the source IP address before it could enter the network. The IP Verify strategy is explained in Figure 5.9:

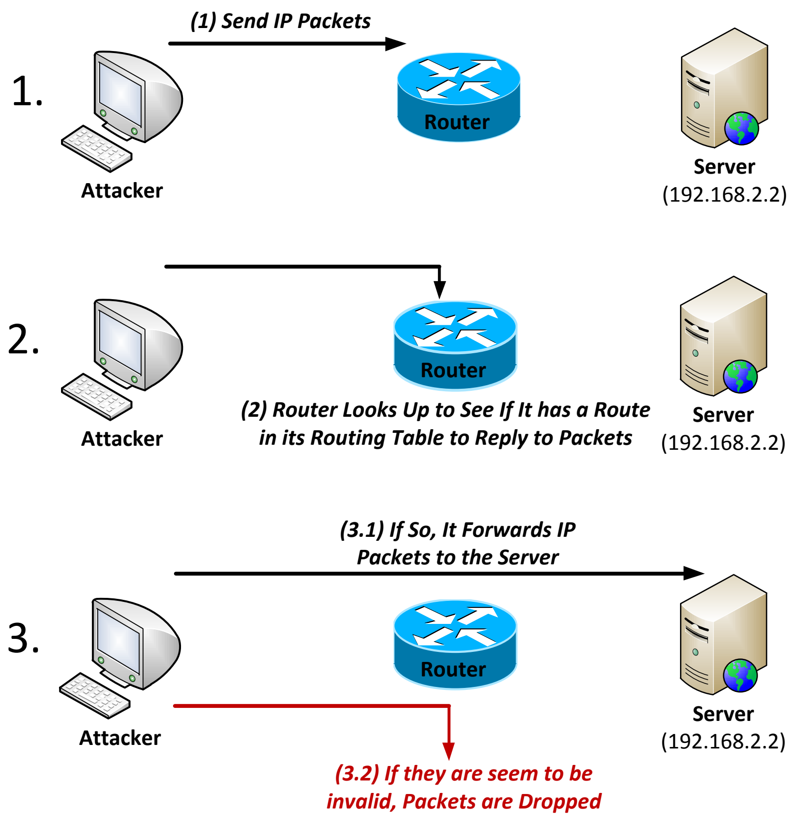

(1) Send IP Packets
1.
Attacker
Router
Server
(192.168.2.2)
2.
Attacker
Router
(2) Router Looks Up to See If It has a Route in its Routing Table to Reply to Packets
Server
(192.168.2.2)
(3.1) If So, It Forwards IP Packets to the Server
3.
Attacker
Router
Server
(192.168.2.2)
(3.2) If they are seem to be invalid, Packets are Dropped

Figure 5.9: IP Verify Strategy

According to Figure 5.9, IP Verify takes the source IP address of a packet received from the Internet and looks up to see if the router has a route in its routing table to reply to that packet. If there is no route in the routing table for a response to return to the source IP, then someone likely has spoofed the packet, and the router drops the packet. However, the disadvantage of this defense is that it cannot prevent DDoS attacks based on a valid IP address (David, 2007). For example, if the attacker launches the attack by using his/her IP address as a source IP address, this defense will be useless, because this defense will determine that the IP address is a valid IP address.

5.2.5 Network Load Balancing

Network Load Balancing (NLB) is a Microsoft implementation of clustering and load balancing that is available in Microsoft Server 2012 (Microsoft, 2014). NLB is software based, which does not require proprietary hardware, and any industry standard compatible computer can be used. In terms of Linux operating system, BalanceNG was selected as a software IP load balancing solution (BalanceNG, 2014). Network Load Balancing has been used by many researchers such as Le, Boutaba and Shaer (2008) who used NLB to improve network performance during DDoS attacks.

In this study, NLB was used to reduce the impact of a UDP flood attack on the network. When the attack took place, it balanced incoming traffic and a workload to an additional server using different paths and cables. It can be noted that the purpose of this solution is not to stop the attack but to share traffic to other paths of the network. As a result, the targeted server still has enough computing resources (e.g., CPU and network bandwidth) to provide service to its clients during the attack (Stephen & Ruby, 2004). Figure 5.10 shows the experimental set-up for the Network Load Balancing:

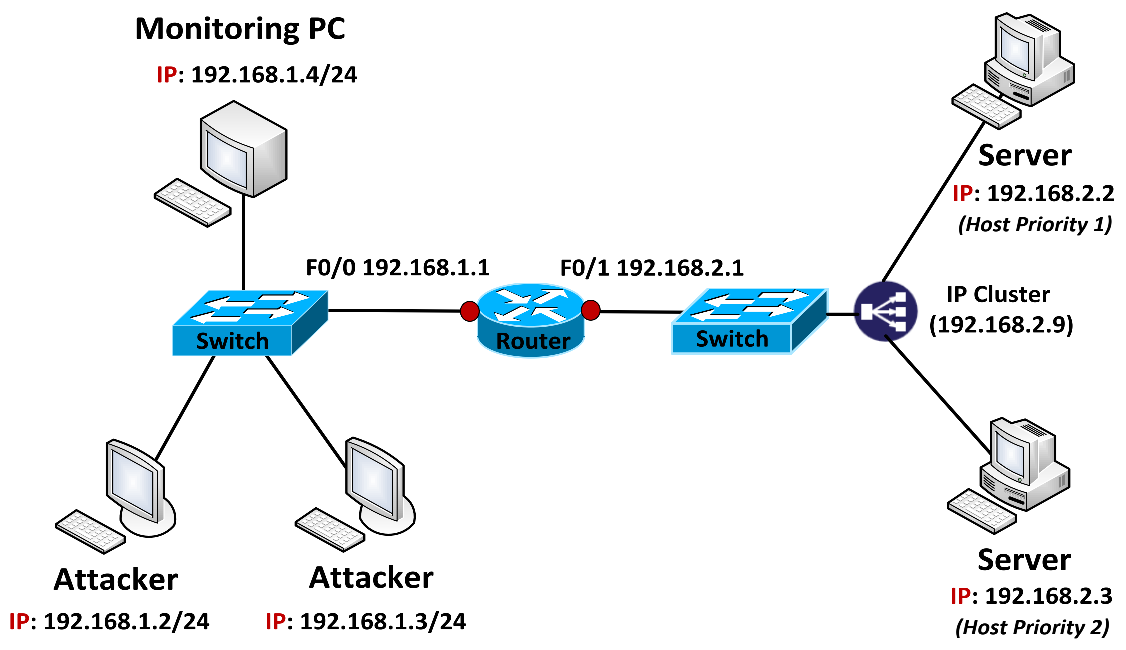


Figure 5.10: Network Load Balancing Network Set-up

As shown in Figure 5.10, in order to implement NLB, an additional server was added to the switch. Two servers were needed to configure an IP cluster, which was used as a “shared” IP address between the two servers. By using this IP address, when a client connected to the server, it automatically connected to the server that had the higher priority first. Similarly, when a large number of attack packets entered the network, NLB shared traffic to both servers equally (in this study the 50:50 rule was used). As a result, the targeted server still had enough computing resources to provide a service to clients during the attack.

There are two NLB modes: unicast mode and multicast mode. In this study, the unicast mode was selected. According to the Cisco website, the multicast mode has an issue with Cisco routers when a client and a server are located in different subnets. This is because Cisco routers do not accept an ARP reply for a unicast IP address that contains a multicast MAC address (Cisco, 2013a). The following parameters were used in this study:

- **Cluster operation mode:** Unicast
- **Port rage:** 0 to 65535
- **Protocols:** TCP and UDP
- **Filtering mode:** Multiple host
- **Load weight:** Equal

5.2.6 Anti-DDoS Firewall

In this study, Anti-DDoS Guardian™ was selected as a software firewall. It is a high-performance firewall designed for modern operating systems including Windows 8 and Windows Server 2012. Anti-DDoS Guardian™ can prevent UDP flood attack by creating IP backlists, limiting UDP bandwidth, UDP connection rate, and UDP packet rate (Beethink, 2014). Figure 5.11 illustrates the experimental set-up for the software firewall:

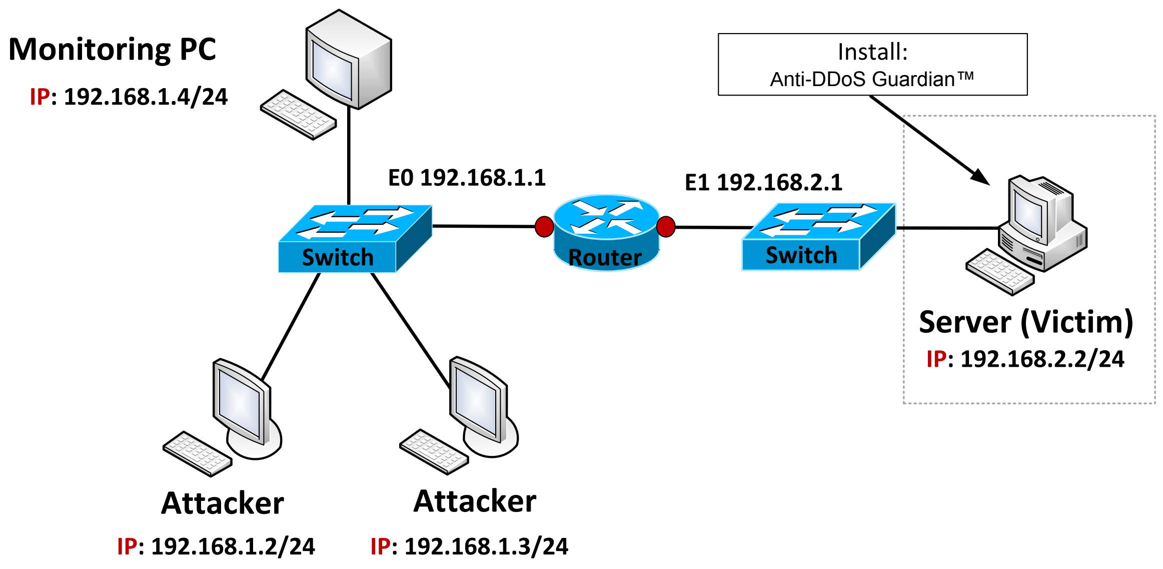
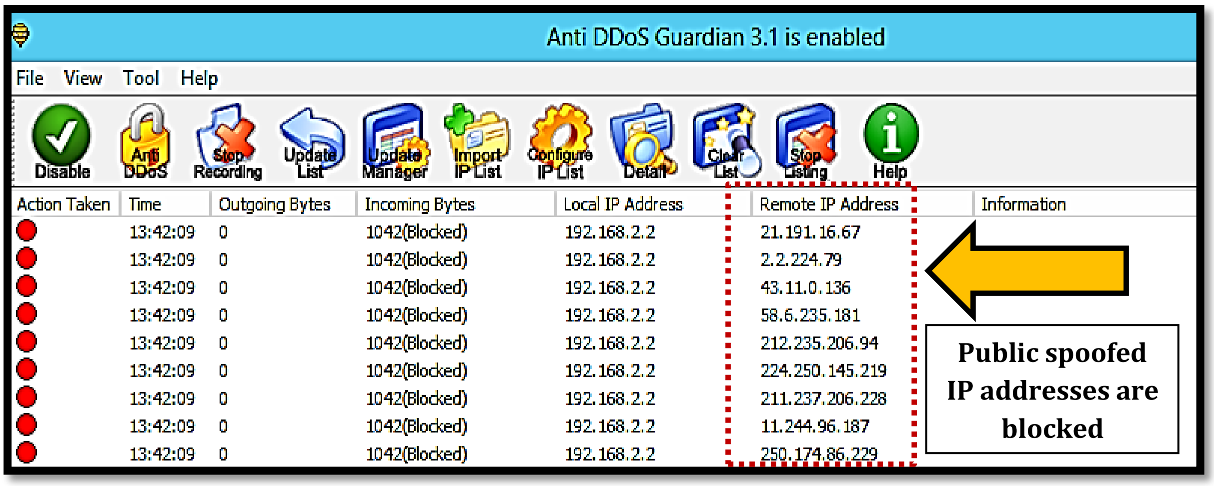


Figure 5.11: Software Firewall Network Set-up

According to Figure 5.11, the software firewall was installed on the victim computer and run in the background as a Windows service. To get the highest performance possible against the attack, the following parameters were used. Note that, if the parameters were set lower than these values, legitimate packets might be dropped during the attack.

- The maximum bandwidth allocated to each IP address is 50,000KBps
- The maximum number of incoming UDP packets per second is 10,000
- The maximum number of incoming ICMP packets per second is 10,000
- The maximum number of concurrent client IP addresses is 1,500



Action Taken	Time	Outgoing Bytes	Incoming Bytes	Local IP Address	Remote IP Address	Information
●	13:42:09	0	1042(Blocked)	192.168.2.2	21.191.16.67	
●	13:42:09	0	1042(Blocked)	192.168.2.2	2.2.224.79	
●	13:42:09	0	1042(Blocked)	192.168.2.2	43.11.0.136	
●	13:42:09	0	1042(Blocked)	192.168.2.2	58.6.235.181	
●	13:42:09	0	1042(Blocked)	192.168.2.2	212.235.206.94	
●	13:42:09	0	1042(Blocked)	192.168.2.2	224.250.145.219	
●	13:42:09	0	1042(Blocked)	192.168.2.2	211.237.206.228	
●	13:42:09	0	1042(Blocked)	192.168.2.2	11.244.96.187	
●	13:42:09	0	1042(Blocked)	192.168.2.2	250.174.86.229	

Figure 5.12: Example of Output from Anti-DDoS Guardian

Figure 5.12 shows the software firewall blocked the public spoofed IP addresses. The result from Wireshark installed on the server (victim computer) also showed that none of the private spoofed IP addresses could gain access to the webserver because the software firewall had dropped the IP addresses that fell into the black-lists. However, the disadvantage of this approach is that it consumed computer resources (especially CPU utilization) at a greater rate than the other defenses because this defense needed to use the CPU of the server in the process of dropping bad traffic (Lad, Alghalbi, & Ahmed, 2013).

5.3 Network Performance Measurements Tools

There are varieties of performance tools available for measuring the performance of various networks. Selecting the right tool for the experiment is critical because different tools have different functionalities. The following list describes the selection criteria for the tools in this thesis:

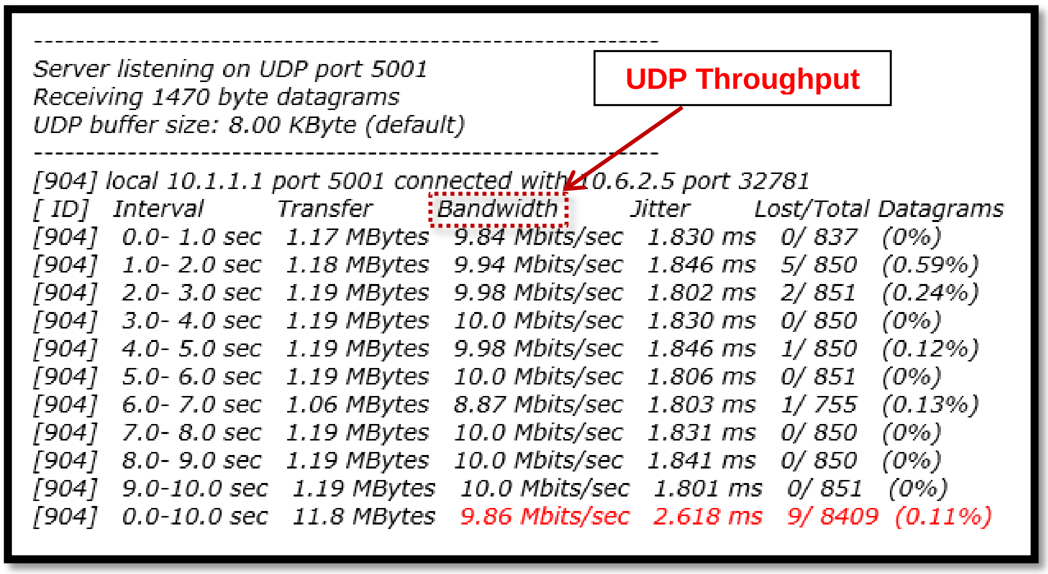
- The tool must support the Internet protocol version 4.
- The tool must support both Windows and Linux based platforms.
- The tool must have high reliability and have been used in other researches before.
- The tool must be able to measure and display results including the required performance metrics.

There were five possible candidates for this project, which are Iperf (Section 5.3.1), Wireshark (Section 5.3.2), TCPing (Section 5.3.3), Webserver Stress Tool (Section 5.3.4), and Hping3 (Section 5.3.5). In this section the analysis, review and discussion of these tools is presented.

5.3.1 Iperf

Iperf (Openmaniak, 2009) is open source software used for measuring the network bandwidth between a client and server. It also has the ability to evaluate the TCP and UDP throughputs, and measure delay, jitter, and packet loss of the targeted computer. Iperf is a command line interface and supports many operating systems including Windows, and Linux.

In this research, Iperf was the primary tool used for measuring user throughputs, jitter, and packet loss. These metrics were collected in three states: before the attack, during the attack, and after using the solutions. To run Iperf, it needed to be installed on two computers, which were a victim computer and a monitoring computer. The former would act as an Iperf client, while the latter would act as the Iperf server. Figure 5.13 shows UDP throughput output from Iperf:



```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[904] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 32781
[ ID]  Interval       Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[904]  0.0- 1.0 sec  1.17 MBytes  9.84 Mbits/sec  1.830 ms  0/ 837  (0%)
[904]  1.0- 2.0 sec  1.18 MBytes  9.94 Mbits/sec  1.846 ms  5/ 850  (0.59%)
[904]  2.0- 3.0 sec  1.19 MBytes  9.98 Mbits/sec  1.802 ms  2/ 851  (0.24%)
[904]  3.0- 4.0 sec  1.19 MBytes  10.0 Mbits/sec  1.830 ms  0/ 850  (0%)
[904]  4.0- 5.0 sec  1.19 MBytes  9.98 Mbits/sec  1.846 ms  1/ 850  (0.12%)
[904]  5.0- 6.0 sec  1.19 MBytes  10.0 Mbits/sec  1.806 ms  0/ 851  (0%)
[904]  6.0- 7.0 sec  1.06 MBytes  8.87 Mbits/sec  1.803 ms  1/ 755  (0.13%)
[904]  7.0- 8.0 sec  1.19 MBytes  10.0 Mbits/sec  1.831 ms  0/ 850  (0%)
[904]  8.0- 9.0 sec  1.19 MBytes  10.0 Mbits/sec  1.841 ms  0/ 850  (0%)
[904]  9.0-10.0 sec  1.19 MBytes  10.0 Mbits/sec  1.801 ms  0/ 851  (0%)
[904]  0.0-10.0 sec  11.8 MBytes  9.86 Mbits/sec  2.618 ms  9/ 8409  (0.11%)
```

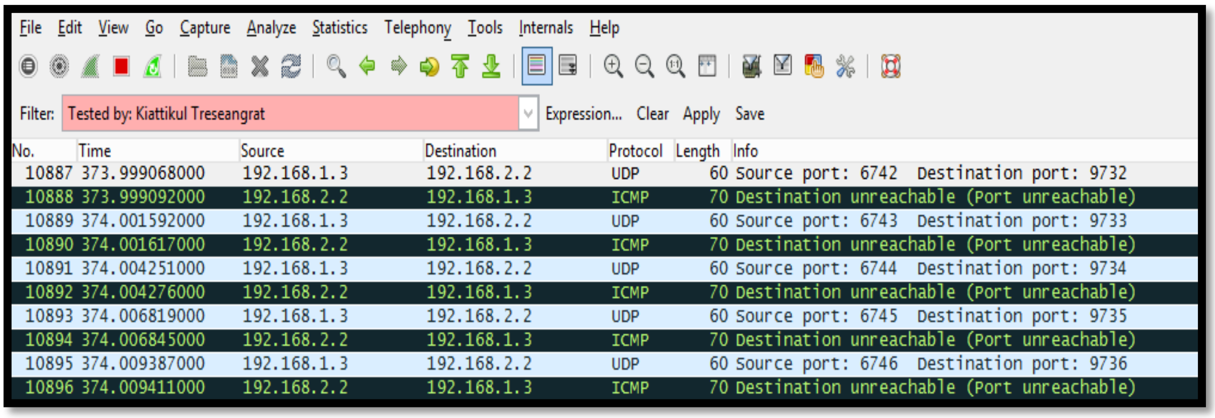
Figure 5.13: Example of UDP throughput Output from Iperf

5.3.2 Wireshark

Wireshark is an open source packet analyzer which can be used for monitoring and analyzing protocols transmitted between the networks. Wireshark is graphical user interface software based, which is compatible with a variety of operating systems, including Windows, and Linux (Wireshark, 2014). Wireshark has many features including:

- It supports over 750 networking protocols.
- It can save captured files in a variety of formats.
- Data display can be refined using a display filter.
- It runs on over 20 platforms.
- It includes a command line version called Tshark.

In this study, the tool was used to monitor UDP packets sent from the attacker computer to the victim computer. To run the software, it needed to be installed on a victim computer. After installation, it required the user to choose the desired network interface card (NIC) that he/she wished to monitor the traffic from. Figure 5.14 illustrates an example of Wireshark output:


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

Filter: Tested by: Kiattikul Treseangrat

```
No.   Time          Source       Destination  Protocol Length Info
10887 373.999068000 192.168.1.3  192.168.2.2  UDP      60 Source port: 6742  Destination port: 9732
10888 373.999092000 192.168.2.2  192.168.1.3  ICMP     70 Destination unreachable (Port unreachable)
10889 374.001592000 192.168.1.3  192.168.2.2  UDP      60 Source port: 6743  Destination port: 9733
10890 374.001617000 192.168.2.2  192.168.1.3  ICMP     70 Destination unreachable (Port unreachable)
10891 374.004251000 192.168.1.3  192.168.2.2  UDP      60 Source port: 6744  Destination port: 9734
10892 374.004276000 192.168.2.2  192.168.1.3  ICMP     70 Destination unreachable (Port unreachable)
10893 374.006819000 192.168.1.3  192.168.2.2  UDP      60 Source port: 6745  Destination port: 9735
10894 374.006845000 192.168.2.2  192.168.1.3  ICMP     70 Destination unreachable (Port unreachable)
10895 374.009387000 192.168.1.3  192.168.2.2  UDP      60 Source port: 6746  Destination port: 9736
10896 374.009411000 192.168.2.2  192.168.1.3  ICMP     70 Destination unreachable (Port unreachable)
```

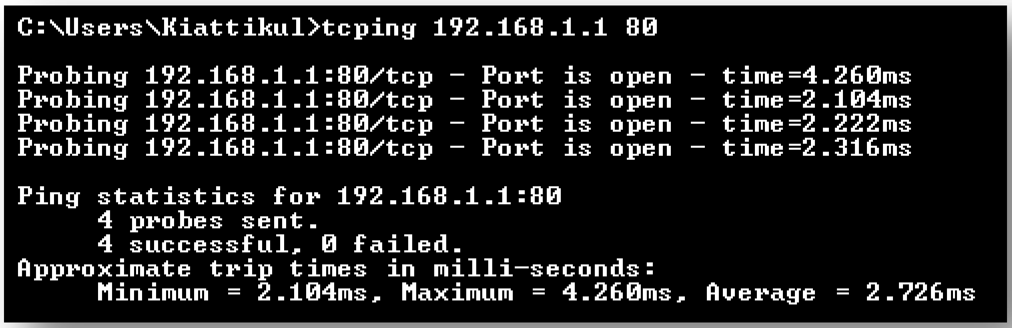
Figure 5.14: Example of Wireshark Output

5.3.3 TCPing

TCPing is a small console application that operates similarly to “ping”. However, it works over a TCP port ("Ping over a TCP Connection", 2013). TCPing allows network administrators to check the round-trip time between the source and destination using TCP (not ICMP). In addition, the tool can be used for testing open ports on remote machines, or as an alternative to the standard “ping” in a case where ICMP packets are blocked or ignored.

In this study, TCPing was a primary tool used to investigate the RTT between the server (victim) and legitimate computer. RTT was collected in three states: before the attack, during the attack, and after using solutions. To achieve this, the tool was installed on the monitoring computer and ran the following command using a command line interface. Figure 5.15 shows an example of TCPing output:

- tcping <the destination IP address> <port number>



```
C:\Users\Kiattikul>tcping 192.168.1.1 80

Probing 192.168.1.1:80/tcp - Port is open - time=4.260ms
Probing 192.168.1.1:80/tcp - Port is open - time=2.104ms
Probing 192.168.1.1:80/tcp - Port is open - time=2.222ms
Probing 192.168.1.1:80/tcp - Port is open - time=2.316ms

Ping statistics for 192.168.1.1:80
     4 probes sent.
     4 successful, 0 failed.
Approximate trip times in milli-seconds:
     Minimum = 2.104ms, Maximum = 4.260ms, Average = 2.726ms
```

Figure 5.15: Example of TCPing Output

5.3.4 Webserver Stress Tool

The Webserver Stress Tool is software for load and performance testing of a webserver, which is designed to simulate up to 1000 simultaneous users accessing a website (Webstress, 2014). By simulating the HTTP requests generated by simulated users, this software can be used to test webserver performance under normal and excessive loads (Bai & Yang, 2006). The Webserver Stress Tool complies with a number of different testing types, including:

- Load Tests
- Performance Tests
- Ramp Tests
- Stress Tests

In this study, the Webserver Stress Tool was used to generate legitimate traffic. In order to identify the performance of the network before the UDP flood attack, legitimate traffic was required. To achieve this, the software was installed on the monitoring computer using 10 users, and the test was run for 5 minutes, which generated traffic at 1,333 packets per 5 minutes. Figure 5.16 shows an example of TCPing output:

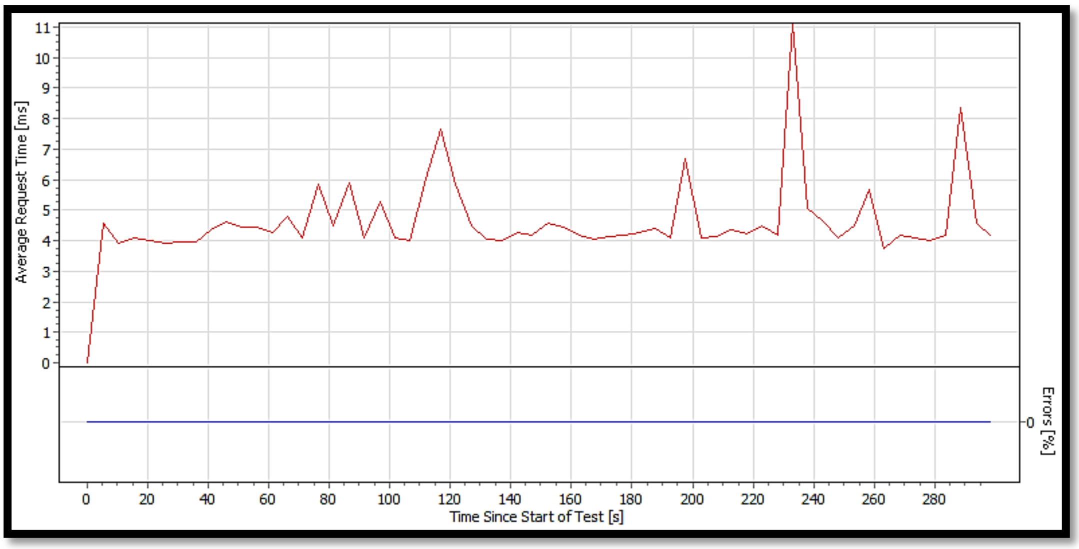


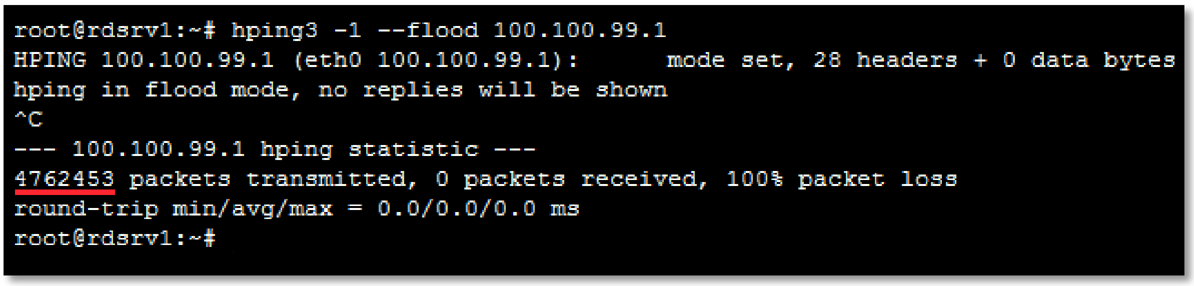
Figure 5.16: Example of Webserver Stress Tool Output

5.3.5 Hping3

Hping3 is a command-line packet crafter used to generate IP packets containing TCP, UDP or ICMP payloads. It has the ability to generate different types of DDoS attacks including UDP flood attacks, TCP flood attacks and Smurf attacks. Hping3 is used by security professionals to test the vulnerability of the computer ports, and firewall rules on networks (Hping, 2014).

In this study, Hping3 was the primary tool used for generating the UDP flood attack on the victim computer. To achieve this, the tool was installed on two attacker computers. One machine would launch UDP flood attack using a valid IP address (the original attacker's IP address), and the other machine, which was a Zombie machine, would attack the victim with massive spoofed IP addresses (random IP addresses).

The UDP flood attack was generated for 5 minutes using an attack rate of 13,000 packets per second and the packet size was 512 Bytes per packet, which generated attack traffic at approximately 50.7 Mbps. Figure 5.17 shows a screenshot of the UDP flood attack using Hping3:



```
root@rdsrv1:~# hping3 -1 --flood 100.100.99.1
HPING 100.100.99.1 (eth0 100.100.99.1):      mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 100.100.99.1 hping statistic ---
4762453 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@rdsrv1:~#
```

Figure 5.17: Example of Hping3 Output

5.4 Chapter Summary

This chapter covered the experimental network set-up designs and network test-bed diagrams. There were four computers used in this study, two workstations were attacker computers, one was a victim computer, and the last one was a monitoring machine. This chapter also covered the experimental set-up and configuration of defenses. There were six solutions used against the UDP flood attack in this study. They were Access Control Lists, Threshold Limit, Hybrid Defense, IP Verify, Network Load Balancing, and Software Firewall. The last section presented the review and discussion of the five main network measurement tools used in this study. An example of screenshots was also presented.

The next chapter is an evaluation of the UDP flood attack on a webserver using Windows Server 2012.

CHAPTER 6

EVALUATION OF UDP FLOOD ATTACK ON WEBSERVER USING WINDOWS SERVER 2012

This chapter covers the analysis of the results gathered from experiments using Windows Server 2012. Section 6.1 covers the impact of a UDP flood attack on throughputs based on TCP and UDP applications. Section 6.2 covers the impact of a UDP flood attack on the webserver, and evaluates round-trip time, packet loss, CPU utilization, and jitter. Section 6.3 covers the analysis of six defense mechanisms, namely, Access Control Lists, Threshold Limit, Hybrid Defense, IP Verify, Network Load Balancing, and DDoS Software Firewall. Throughputs, round-trip time, packet loss, CPU utilization of the network with/without defenses are compared.

6.1 Impact of UDP Flood Attack on Throughputs

The section covers the impact of a UDP flood attack on throughputs before and during the attack. TCP and UDP were selected as the transmission protocols. The primary tools used in this study were Iperf and Hping3. The former was used to fill up the Ethernet link at 86.4 Mbps while Hping3, which was the attack tool, was used to generate attack traffic at 156.2 Mbps. The server was put on high traffic load to evaluate the impact on the bandwidth. To ensure high data accuracy, the test was repeated 30 times and result average and runs continued until standard deviation of results was below 0.07% of the average. The evaluation process and measurement tools used in this chapter were discussed in Section 4.3.2.2 and Section 5.3.

6.1.1 TCP Throughput

This section presents the impact of a UDP flood attack on throughputs. TCP was selected as the transmission protocols, which was used for webserver traffic. The Y axis shows the number of throughputs in megabits per second, while the X axis shows the length of the experiment in seconds.

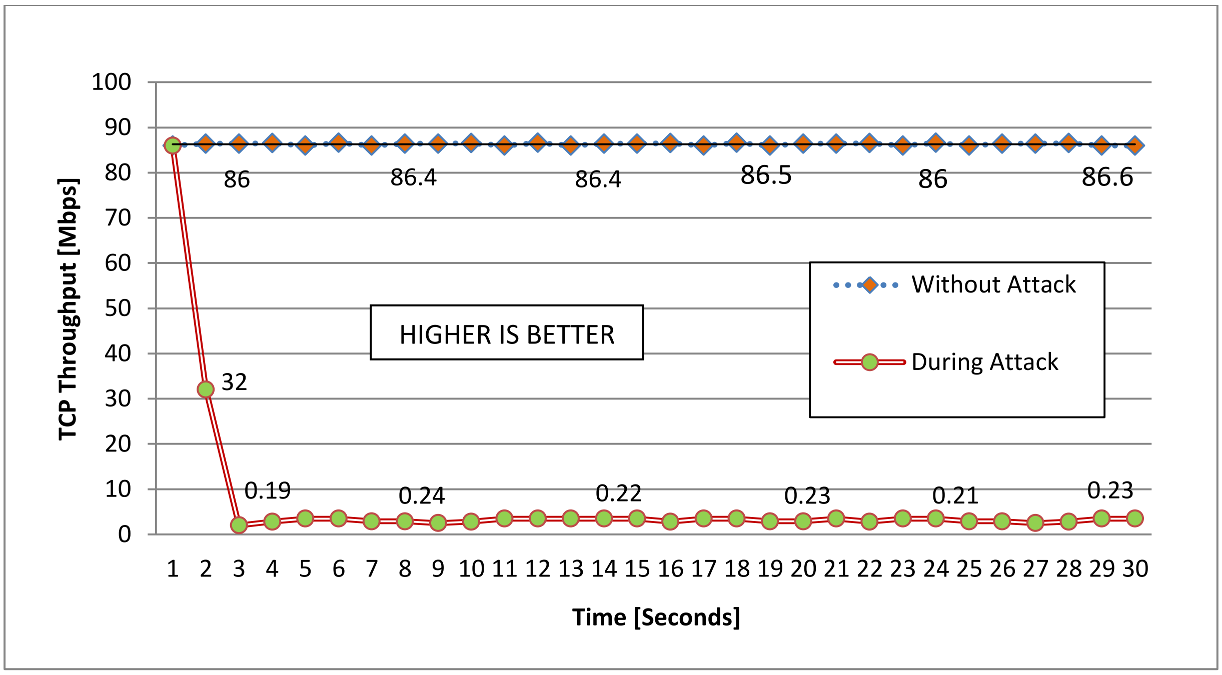


Figure 6.1: TCP Throughputs Before and During Attack on Windows Server 2012

Scenario	Time (Seconds)	Throughput (Mbps)	Standard Deviation
Before Attack	All	86.40	0.002
During Attack	2	32	0.045
	3	0.19	0.064
	After 4 Seconds	0.19-0.24	-

Table 6.1: Average TCP Throughputs and Standard Deviation for 30 Runs Before and During Attack on Windows Server 2012

Figure 6.1 illustrates the TCP throughput values before and during the attack. The result shows that the TCP throughput value before the attack was constant at 86 Mbps. This number was generated by Iperf. During the attack, however, this number significantly dropped from 86 Mbps to 0.19 Mbps within 3 seconds. The reason is that the attacker had sent a large number of UDP packets to the victim computer and eventually caused network congestion between the two nodes. It can be noted that the TCP throughput value was almost stable after 3 seconds, at around 0.19 Mbps to 0.24 Mbps.

6.1.2 UDP Throughput

This section presents the impact of a UDP flood attack on throughputs. UDP was selected as the transmission protocol, which was used for VoIP applications. The Y axis shows the number of throughputs in megabits per second, while the X axis shows the length of the experiment in seconds.

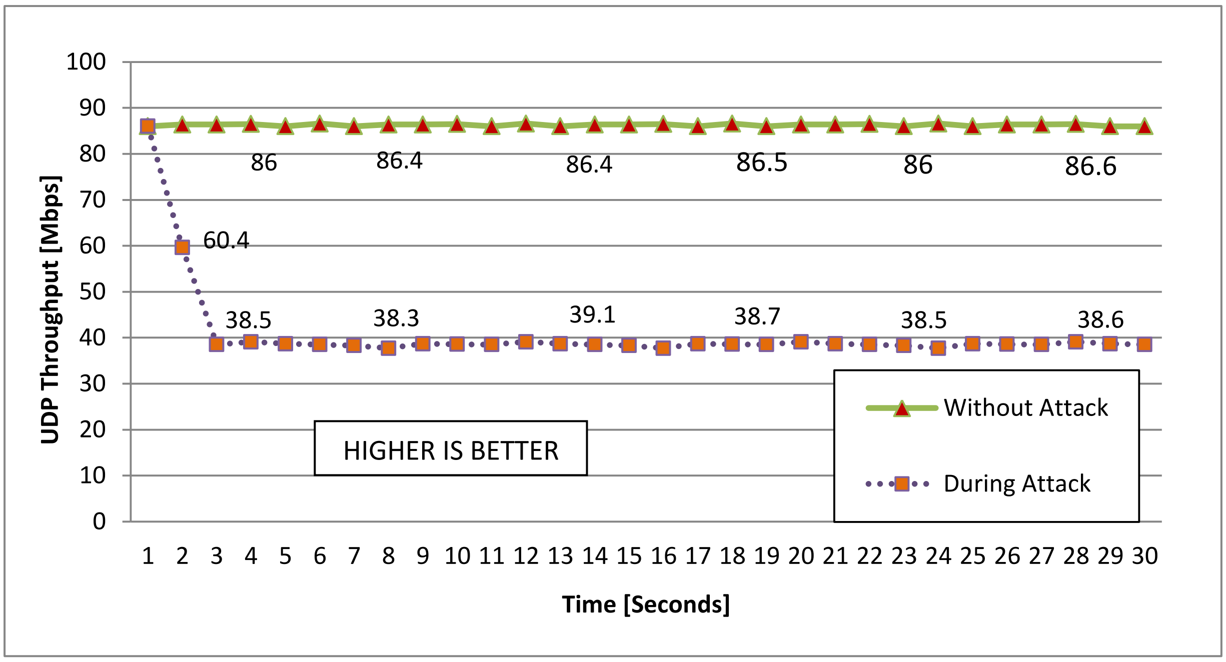


Figure 6.2: UDP Throughputs Before and During Attack on Windows Server 2012

Scenario	Time (Seconds)	Throughput	Standard Deviation
Before Attack	All	86.50	0.002
During Attack	2	60.48	0.041
	3	28.52	0.012
	After 4 Seconds	38-39	-

Table 6.2: Average UDP Throughputs and Standard Deviation for 30 Runs Before and During Attack on Windows Server 2012

Figure 6.2 illustrates the UDP throughput values before and during the attack. The result shows that the UDP throughput value before the attack was similar to TCP (Figure 6.1), at around 86 Mbps. During the attack, however, this number dropped to 38.5 Mbps within 3 seconds. After that, the UDP throughput value was constant at around 38 to 39 Mbps. It can be noted that the UDP throughput value during the attack was higher than TCP. The reason is that UDP is a connectionless protocol, which does not require a dedicated end-to-end connection. Also, it does not use a flow control; therefore, it can keep sending data even if traffic gets congested (Kolahi, Rico, & Hong, 2013). In terms of TCP, it uses a flow control protocol to limit the rate a sender transfer's data to guarantee reliable delivery (Loshin, 2003). This could be the reason why the number of the UDP throughputs was higher than TCP during the attack.

6.2 Impact of UDP Flood Attack on Webserver Using Windows Server 2012

This section covers the impact of a UDP flood attack on the webserver using Windows Server 2012. The primary tools used in this study were Webserver Stress Tool and Hping3. The former was used to generate the connection request from users to the webserver assuming on average 10 users per second, while Hping3 was used to generate the attack traffic at 13000 packets per second (unless the attack packet rate and packet size are otherwise defined in the next sections). The monitoring PC was used to gather DDoS parameters when a legitimate or attack traffic was launched. To ensure high data accuracy, the test was repeated 30 times and data average and runs continued until the standard deviation of results was below 0.07% of the average. In addition, in some studies, the attack rate and attack packet sizes were increased to find out the effect of traffic load on different attack packet rates and attack packet sizes.

6.2.1 Round-trip Time

Round-trip time (RTT) is the time required for a packet to travel from a source to a destination and back again. The primary tool used in this study was TCPing, which was the tool used to gather the delay values of the webserver. The test was repeated 30 times using an attack rate of 13000, 12000, and 9000 packets per second, while the attack packet size was constant at 512 Bytes per packet. These tests generated attack traffic at 50.7 Mbps, 46.8 Mbps, and 35.1 Mbps, respectively. This study was done in order to evaluate the impact of packet rate on delay.

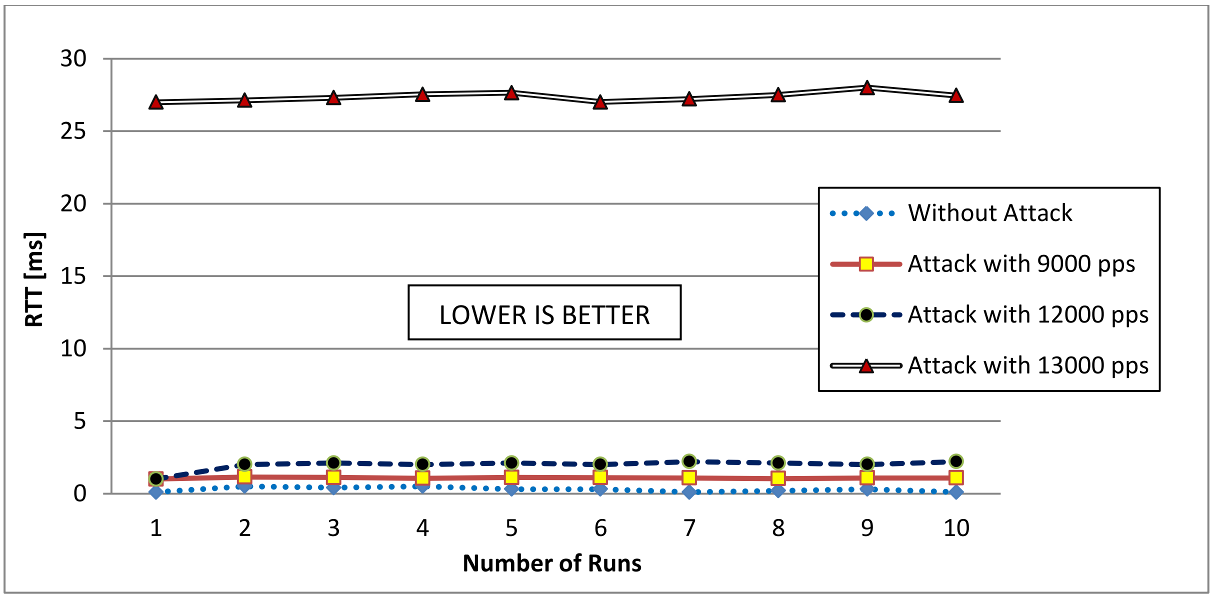


Figure 6.3: Round-trip Time Before and During Attack on Webserver Using Windows Server 2012

Attack Packet Rate (pps)	Average RTT(ms)	Standard Deviation
Before Attack	0.990ms	0.016
9000	1.096ms	0.031
12000	1.275ms	0.045
13000	27.384ms	0.063
14000+	Connection out	-

Table 6.3: Average RTT and Standard Deviation for 30 Runs Before and During Attack on Webserver Using Windows Server 2012

Figure 6.3 illustrates the average RTT for 30 runs between the client and the webserver during the UDP flood attack. The result shows that the RTT before the attack was 0.99ms. In terms of RTT during the attack, it can be observed that the RTT value was generally increased as the packet rate increased. By using an attack packet rate at 9000pps (35.1 Mbps), the average RTT slightly increased from 0.99ms to 1.09ms. By using 12000pps (46.8 Mbps), the average RTT went up to 1.27ms. Interestingly, the average RTT significantly increased from 0.99ms to 27.38ms if the attacker used a packet rate of 13000pps (50.7 Mbps). In addition, the connection was cut off if the packet rate increased any further and it was impossible to measure RTT due to high traffic loads.

6.2.2 Packet Loss

In this study, packet loss refers to the amount of legitimate packets lost during the UDP flood attack. The primary tool used in this study was Iperf, which was the tool used to gather the packet loss values. The attack packet sizes used in this study were 64, 128, 256, 512, 1024, 1280, and 1518 Bytes, while the attack packet rate was constant at 13000 packets per second. They generated attack traffic at 6.34 Mbps, 12.69 Mbps, 25.3 Mbps, 50.7 Mbps, 101.5 Mbps, 126.9 Mbps, and 150.5 Mbps, respectively. This study was done in order to evaluate the packet loss versus attack traffic load.

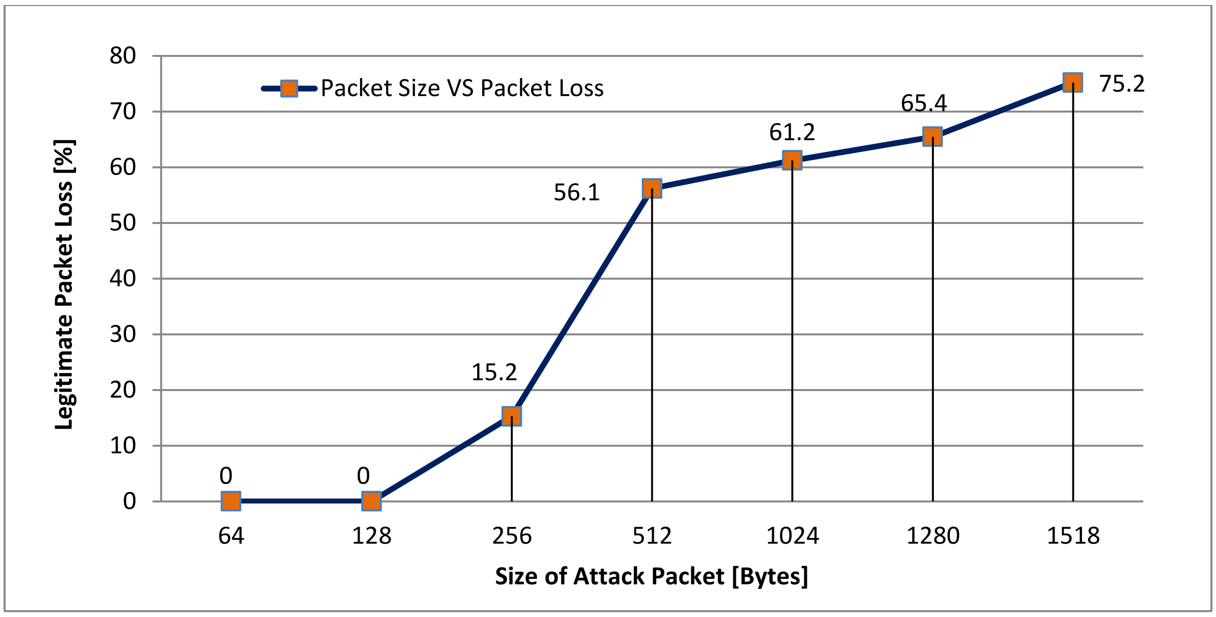


Figure 6.4: Size of Attack Packet VS Legitimate Packet Loss on Webserver Using Windows Server 2012

Attack Packet Size (Bytes)	Legitimate Packet Loss (%)	Standard Deviation
256	15.215	0.021
512	56.170	0.042
1024	61.215	0.023
1280	65.466	0.047
1518	75.200	0.031

Table 6.4: Size of Attack Packet VS Legitimate Packet Loss and Standard Deviation for 30 Runs on Webserver Using Windows Server 2012

Figure 6.4 illustrates the legitimate packet loss versus the attack packet size. On the whole, there was no packet loss on the frame sizes 64, and 128 (size of attack packet). By using 256 Bytes (25.3 Mbps), the average of legitimate packet loss went up from 0% to 15.2%. The packet loss value significantly increased to 56.1% if the attack packet size was 512 Bytes (50.7 Mbps). On the largest frame size, 1518 Bytes (150.5 Mbps), the average packet loss value went up to 75.2%. The results in this study showed that the packet loss was related to the size of the attack packet. This is because the larger packet size could consume more of the network bandwidth. In other words, the legitimate packets start to get dropped when the availability of bandwidth is insufficient.

6.2.3 CPU Utilization

This section shows the impact of the CPU usage before and during the attack on the webserver using Windows Server 2012. The primary tool used in this study was Microsoft Resource Monitor, which was the tool used to monitor the CPU usage on the server.

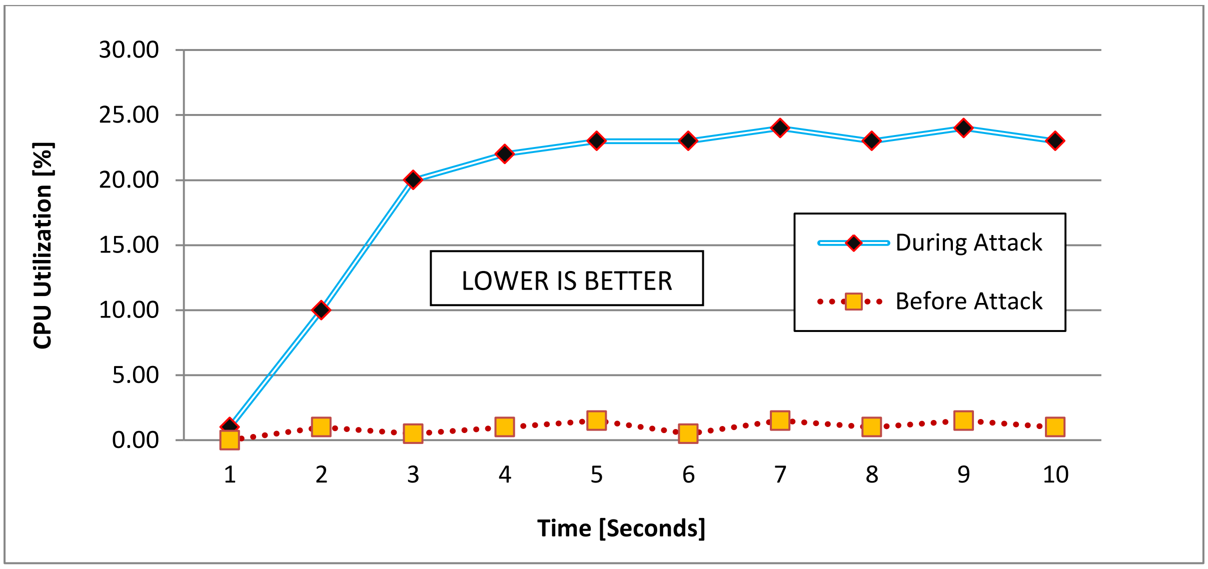


Figure 6.5: CPU Utilization Before and During Attack on Webserver Using Windows Server 2012

Scenario	Time (Second)	CPU Utilization (%)	Standard Deviation
Before Attack	All	1-2	-
During Attack	2	9.36	0.150
	3	19.52	0.077
	4	21.28	0.080
	After 5 Seconds	19-24	-

Table 6.5: Average CPU Utilization and Standard Deviation Measured 30 Times Before and During Attack on Webserver Using Windows Server 2012

Figure 6.5 illustrates the CPU utilization before and during the attack on the webserver using Windows Server 2012. The result shows that the CPU usage before the attack was constant at 1 to 2%. During the attack, however, the CPU utilization went up at approximately 10% within 2 seconds and increased to 22% in 4 seconds. Afterwards, it fluctuated between 19% and 24%. In terms of the RAM utilization, the result shows that there was only a minor difference between RAM usage before and during the attack (the graph is not shown). The RAM utilization before the attack was constant at 8%, and went up to between 10% and 15% during the attack. In terms of hard disk utilization, there was no difference between the hard disk usage before and during the attack (the graph is not shown); the hard disk utilization was constant at 100 KB per second. This indicates that a UDP flood attack has more impact on the CPU than on the RAM and hard disk.

6.2.4 Jitter

This section presents a comparison of legitimate jitter values during the UDP flood attack. The primary tool used in this study was Iperf, which was the tool used to gather the jitter values of the victim computer. The attack packet sizes used in this study were 64, 128, 256, 512, 1024, 1280, and 1518 Bytes, while the attack packet rate was constant at 13000 packets per second. These tests generated attack traffic at 6.34 Mbps, 12.69 Mbps, 25.3 Mbps, 50.7 Mbps, 101.5 Mbps, 126.9 Mbps, and 150.5 Mbps, respectively.

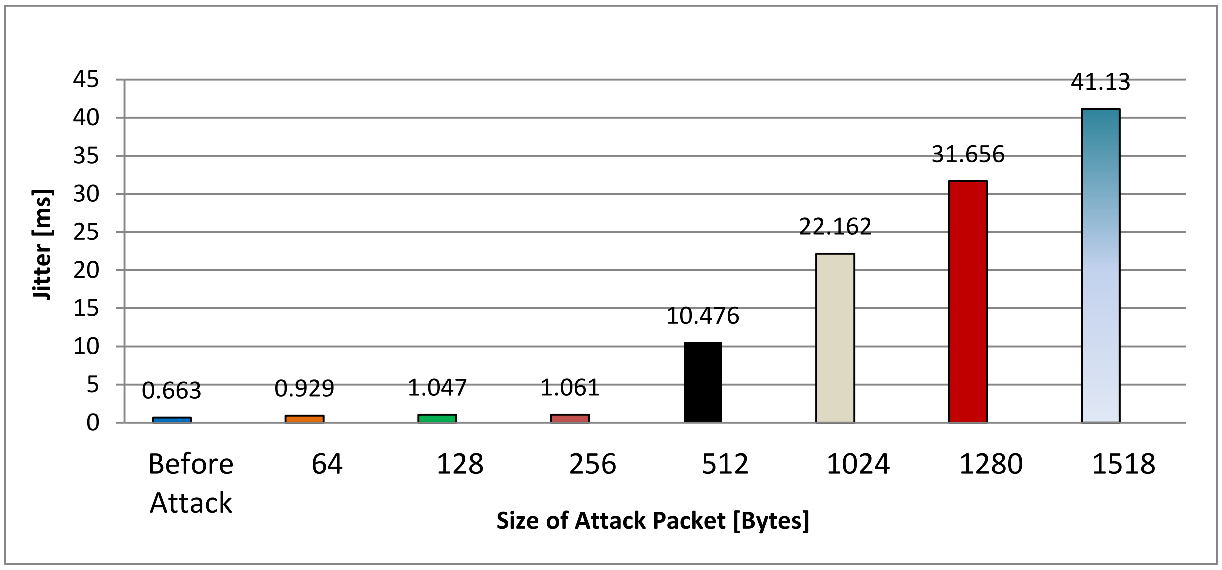


Figure 6.6: Jitter VS Packet Size on Webserver Using Windows Server 2012

Attack Packet Size (Bytes)	Jitter (ms)	Standard Deviation
Before Attack	0.663	0.031
64	0.929	0.022
128	1.047	0.059
256	1.061	0.021
512	10.476	0.047
1024	22.162	0.023
1280	31.656	0.067
1518	41.130	0.074

Table 6.6: Size of Attack Packet VS Jitter and Standard Deviation for 30 Runs on Webserver Using Windows Server 2012

Figure 6.6 illustrates the average legitimate jitter value versus packet sizes (attack packet). On the whole, the result shows that the jitter value was increased as the attack packet size increased. Without the attack, the jitter value was 0.663ms. During attack, the jitter values slightly went up when using the attack packet sizes between 64 to 256 Bytes (6.3 Mbps, 12.6 Mbps, and 25.3 Mbps).

The most noticeable feature of this graph is that the jitter value significantly increased when using attack packet sizes between 512 Bytes and 1518 Bytes (50.7 Mbps, 101.5 Mbps, 126.9 Mbps, and 150.5 Mbps). By using 512 Bytes (50.7 Mbps), the average jitter value went up from 0.663ms to 10.476ms. The jitter value increased to 22.162ms if the attack packet size was 1024 Bytes (101.5 Mbps). On the largest frame size, 1518 Bytes (150.5 Mbps), the average jitter value went up significantly to 41.13ms. High jitter values cause degradation in performance and lead to instability in services especially in real-time applications (Marti, Fuertes, Fohler, & Ramamritham, 2001).

6.3 Comparisons of Defense Mechanisms on Windows Server 2012

This section covers an analysis of defense mechanisms against a UDP flood attack on Windows Server 2012. There were six defenses used in this study, namely, Access Control Lists, Threshold Limit, Hybrid Defense, IP Verify, Network Load Balancing, and DDoS Software Firewall. The efficiency of each defense was evaluated by comparing throughputs, delay, packet loss, and CPU utilization values before and after using the defenses. To ensure high data accuracy, the test was repeated 30 times and result average and runs continued until standard deviation of results was below 0.07% of the average. The evaluation process and experimental set-up of defenses were discussed in Section 4.3.2.2 and Section 5.2.

6.3.1 TCP Throughput

This section covers the impact of the UDP flood attack on throughputs after using defenses. TCP was selected as the transmission protocol, which was used for webserver traffic. The primary tools used in this study were Iperf and Hping3. The former was used to fill up the Ethernet link at 86.4 Mbps, while Hping3, which was the attack tool, was used to generate attack traffic at 156.2 Mbps.

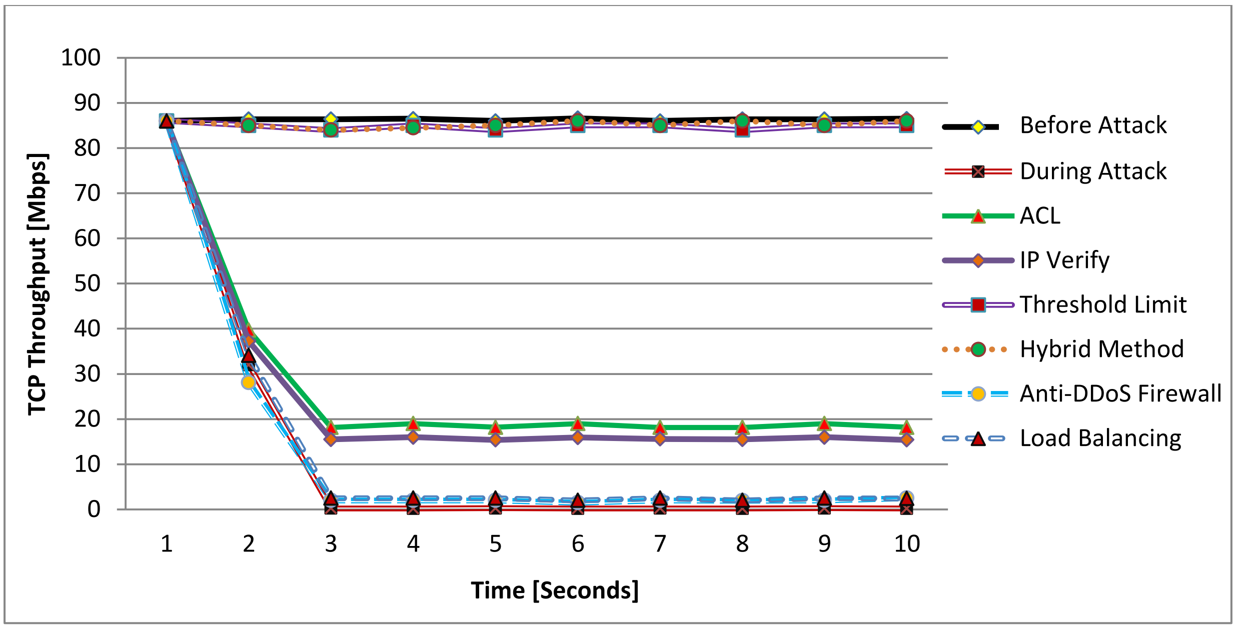


Figure 6.7: TCP Throughputs After Using Defenses on Windows Server 2012

Scenario	Defense	Throughput (Mbps)	Standard Deviation
Before Attack	-	86.60	0.002
During Attack	No Defense	0.19	0.066
	Hybrid Method	86.40	0.012
	Threshold Limit	86.30	0.018
	ACLs	18.48	0.047
	IP Verify	15.67	0.081
	Load Balancing	0.28	0.071
	Software Firewall	0.251	0.068

Table 6.7: Average TCP Throughputs and Standard Deviation For 30 Runs After Using Defenses on Windows Server 2012

Figure 6.7 presents TCP throughput values after using six defenses. The result shows that the TCP throughput value before the attack was stable at around 86 to 86.6 Mbps. During the attack, however, this number significantly dropped to 0.19 Mbps. This is because the attacker sent huge numbers of attack packets which eventually caused bandwidth consumption.

The most effective defenses in this study were the Hybrid Method and the Threshold Limit in which the number of throughput values before the attack and after using the solutions were almost the same, which was about 86.60 Mbps. ACLs came in third, which increased the TCP user throughput value from 0.19 Mbps to 18.48 Mbps. This number was similar to IP Verify, which increased TCP throughputs to 15.67 Mbps. The most ineffective defense mechanisms in this study were Network Load Balancing and the Software Firewall, which only increased TCP user throughput values from 0.19 Mbps to 0.288 Mbps, and 0.251 Mbps, respectively. The major disadvantage of Network Load Balancing is that it did not aim to stop the attack, instead, it shared incoming traffic with an additional server. Therefore, the network bandwidth would be entirely consumed if the attacker kept sending the attack packets. In terms of the Software Firewall, it blocked attack traffic at the victim computer (unlike other defenses that dropped the attack packets at the router). As a result, the targeted computer was still affected by the attack packets.

6.3.2 UDP Throughput

This section covers the impact of the UDP flood attack on throughputs after using defenses. UDP was selected as the transmission protocol, which was used for VoIP traffic. The primary tools used in this study were Iperf and Hping3. The former was used to fill up the Ethernet link at 86.4 Mbps, while Hping3, which is the attack tool, generated the attack traffic at 156.2 Mbps.

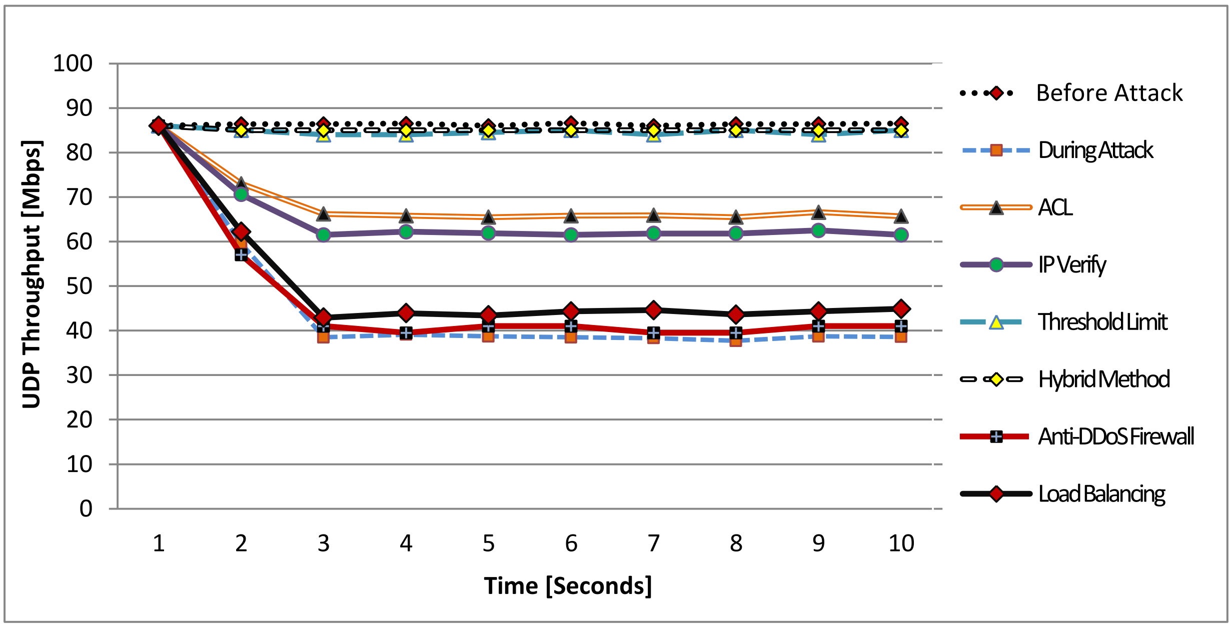


Figure 6.8: UDP Throughputs After Using Defenses on Windows Server 2012

Scenario	Defense	Throughput (Mbps)	Standard Deviation
Before Attack	**-**	86.60	0.002
During Attack	No Defense	38.70	0.081
	Hybrid Method	86.50	0.012
	Threshold Limit	86.40	0.017
	ACLs	66.01	0.020
	IP Verify	61.96	0.026
	Load Balancing	43.35	0.032
	Software Firewall	39.10	0.071

Table 6.8: Average UDP Throughputs and Standard Deviation For 30 Runs After Using Defenses on Windows Server 2012

Figure 6.8 presents a comparison of UDP throughput values after using six defenses. The result shows that the UDP throughput value before the attack was stable at about 87 Mbps. During the attack, however, this number dropped to 38.7 Mbps. This is because the attacker sent huge numbers of attack packets, which eventually caused bandwidth consumption.

The most effective solutions in this study were the Hybrid Method and the Threshold Limit. Both defenses increased UDP throughput values from 38.7 Mbps to 86.5 Mbps, and 86.4 Mbps, respectively. ACLs, which increased the number of UDP throughput values from 38.7 Mbps to 66.01 Mbps, came in third. This number was similar to IP Verify, which increased the UDP throughput values to 61.96 Mbps. Network Load Balancing and the Software Firewall were the most ineffective defense mechanisms in this study, insignificantly increasing UDP throughput values from 38.7 Mbps to 43.35 Mbps and 39.1 Mbps, respectively.

6.3.3 Round-trip Time

This section presents the comparison of RTT before and after using six defenses on the webserver using Windows Server 2012. The primary tool used in this study was TCPing, which was the tool used to gather the delay values of the webserver.

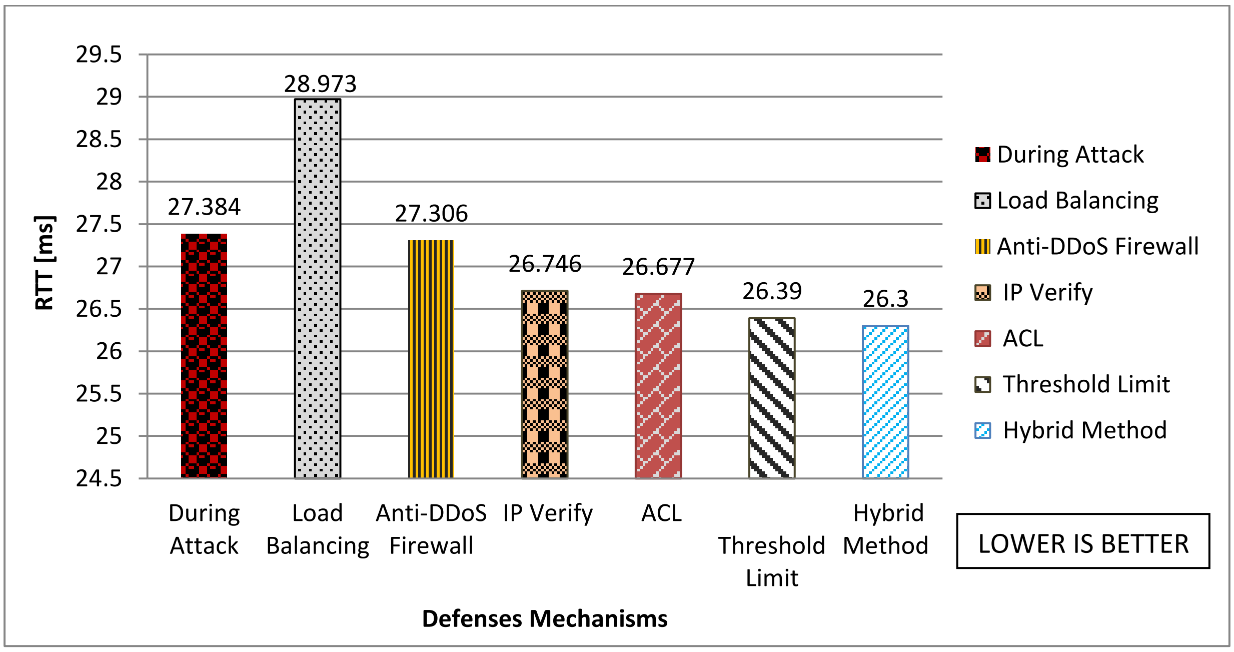


Figure 6.9: Round-trip Time After Using Defenses on Webserver Using Windows Server 2012

Scenario	Defense	Average RTT (ms)	Standard Deviation
Before Attack	**-**	0.990	0.016
During Attack	No Defense	27.384	0.063
	Anti-DDoS Firewall	27.306	0.076
	IP Verify	26.746	0.055
	Load Balancing	28.973	0.037
	ACLs	26.677	0.032
	Threshold Limit	26.390	0.014
	Hybrid Method	26.300	0.060

Table 6.9: Average RTT and Standard Deviation For 30 Runs After Using Defenses on Webserver Using Windows Server 2012

Figure 6.9 illustrates the average round-trip time for 30 runs after using DDoS defenses on the webserver using Windows Server 2012. The result shows that the RTT before the attack was 0.99ms. During the attack, however, this number significantly increased to 27.38ms. In terms of defenses, the Hybrid Method and the Threshold Limit were the most effective defenses in this study. Both solutions could decrease the average RTT from 27.38ms to 26.30ms and 26.39ms, respectively.

ACLs came in third, which reduced the RTT from 27.38ms to 26.67ms. IP Verify came in fourth, which decreased the RTT to 26.746ms. The most inefficient defense in this study was the Software Firewall, which insignificantly reduced the RTT from 27.38ms to 27.30ms. Unlike other defenses, the Software Firewall dropped the attack packets at the victim computer; therefore, the victim computer was still affected by the attack packets. The other defenses, e.g., ACLs, IP Verify, Threshold Limit, and Hybrid Method, dropped the attack packets at the router.

Network Load Balancing resulted in the highest RTT, which was approximately 28.97ms. This number was even higher than the RTT value during the attack. The reason is that this solution required the system resources to examine incoming packets and made load-balancing decisions, and therefore imposed an overhead on network performance (Microsoft, 2014).

6.3.4 Packet Loss

This section presents the comparison of the packet loss values before and after using six defenses on the webserver using Windows Server 2012. The primary tool used in this study was Iperf, which was the tool used to gather the packet loss values.

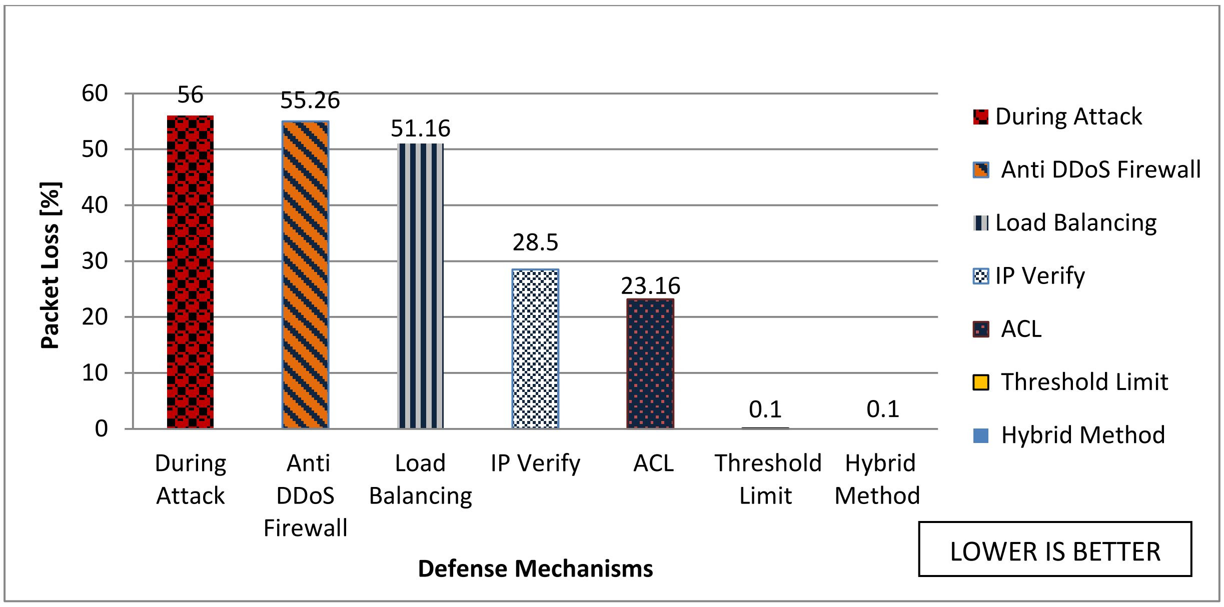


Figure 6.10: Packet Loss Values After Using Defenses on Webserver Using Windows Server 2012

Scenario	Defense	Packet Loss (%)	Standard Deviation
Before Attack	-	0	-
During Attack	No Defense	56.17	0.042
	Anti-DDoS Firewall	55.26	0.024
	IP Verify	28.50	0.017
	Load Balancing	51.16	0.050
	ACLs	23.16	0.025
	Threshold Limit	0.1	0.035
	Hybrid Method	0.1	0.018

Table 6.10: Average Packet Loss and Standard Deviation For 30 Runs After Using Defenses on Webserver Using Windows Server 2012

Figure 6.10 presents a comparison of legitimate packet loss values after using DDoS defenses on the webserver using Windows Server 2012. The result shows there was no packet loss before the attack. During the attack, however, the number of legitimate packet loss went up to 56.1%. In terms of defenses, the most effective defenses in this study were Hybrid Method and Threshold Limit, which significantly reduced the packet loss value from 56.1% to 0.1%. IP Verify and ACLs moderately decreased the number of packet loss from 56.1% to 28.5%, and 23.1% respectively. The ineffective defenses in this study were Network Load Balancing and the Software Firewall. Both solutions insignificantly reduced the packet loss values from 56.1% to 51.1%, and 55.2% respectively.

6.3.5 CPU Utilization

This section shows the CPU utilization before and after using defenses on the webserver using Windows Server 2012. The primary tool used in this study was Microsoft Resource Monitor, which was the tool used to monitor the CPU usage on the server.

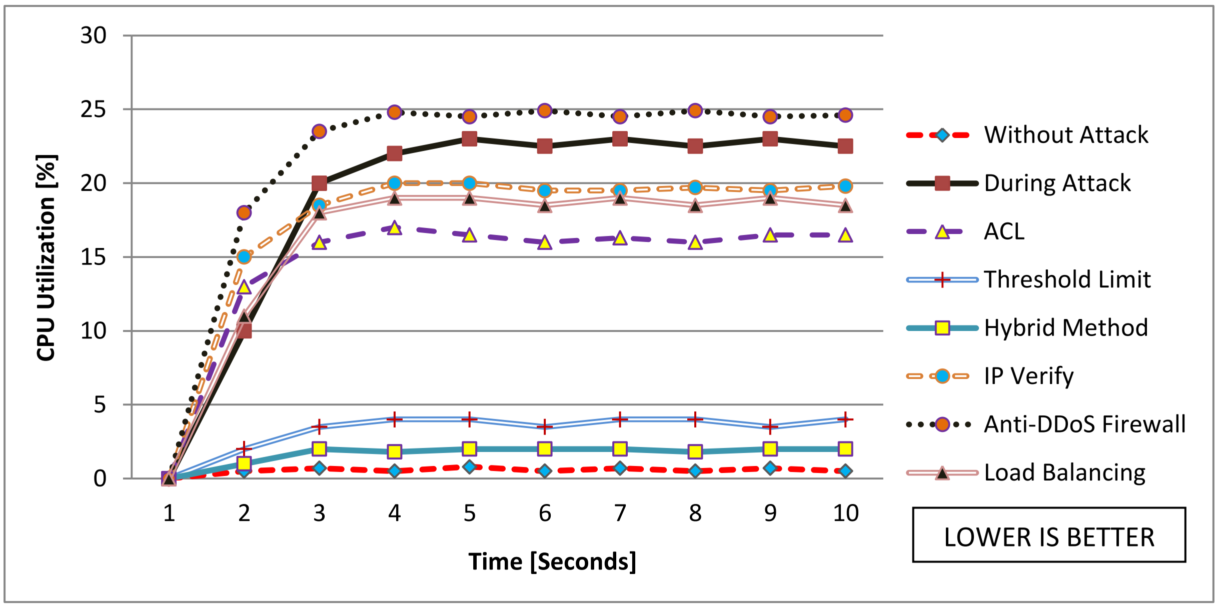


Figure 6.11: CPU Utilization After Using Defenses on Webserver Using Windows Server 2012

Scenario	Defense	CPU Utilization (%)
Before Attack	-	1-2
During Attack	No Defense	19-24
	Anti-DDoS Firewall	25-26
	IP Verify	18-20
	Load Balancing	17-19
	ACLs	16-18
	Threshold Limit	2-3
	Hybrid Method	2

Table 6.11: Average CPU Utilization For 30 Runs After Using Defenses on Webserver Using Windows Server 2012

Figure 6.11 illustrates the average CPU utilization before and after using DDoS defenses. The result shows that the CPU usage before the attack was stable at around 1%. During the attack, the CPU usage fluctuated between 19% and 24%.

In terms of defenses, the most effective defense in this study was the Hybrid Method, which reduced the CPU usage from 24% to 2%. This number was similar to Threshold Limit, which decreased the CPU utilization to 3%. ACLs came in third, and reduced the CPU usage from 24% to 16%. In terms of Network Load Balancing, the defense decreased the CPU usage to approximately 18%. This figure was similar to IP Verify, which reduced the server's CPU to

20%. Interestingly, the Software Firewall consumed the CPU utilization more than the other defenses, at around 25%. This number was even higher than the CPU usage during the attack. It is noted that the defenses (except DDoS firewall) did not reduce the CPU utilization directly. Instead, they had dropped attack packets before the packets could enter the server. As a result, the CPU did not have to work all the time in order to reply to the attack packets. As explained in Section 3.2 (Characteristics of UDP flood attack) the victim computer will send ICMP packets back to the source IP addresses when it receives UDP packets on the closed ports.

6.4 Chapter Summary

This chapter presented the results from the tests conducted in the computer lab. Several metrics, e.g., throughputs, delay, packet loss, jitter, and CPU utilization, were collected and used to analyze the impact of a UDP flood attack on the webserver using Windows Server 2012. This chapter also evaluated six defenses, namely, Access Control Lists, Threshold Limit, Hybrid Defense, IP Verify, Network Load Balancing, and DDoS Software Firewall.

The results showed that the performance of Windows Server 2012 was reduced during the UDP flood attack. The TCP throughputs before the attack were constant at 86 Mbps. During the attack, this number dropped significantly to 0.19 Mbps (Figure 6.1). The reason is that an attacker had sent a large number of UDP packets to the server, and eventually caused network congestion between the two nodes. The UDP throughputs before the attack were around 86 Mbps. During the attack, the UDP throughputs dropped moderately to 38.3 Mbps (Figure 6.2). The result showed that the RTT value generally increased as the attack packet rate increased. The average RTT before the attack was 0.99ms, and significantly increased to 27.38ms during the attack (Figure 6.3). The CPU usage before the attack was about 2% and went up significantly to 24% during the attack (Figure 6.5).

After using the defenses on Windows Server 2012, the results showed that the performance of Windows OS increased. The Hybrid Method and the Threshold Limit were the most effective defenses against the UDP flood attack in all studies, whereas the Software Firewall and Network Load Balancing were the least effective defenses (IP Verify, and ACLs were average defenses). The Hybrid Method and Threshold Limit could increase the TCP throughput from 0.19 Mbps (no defense) to 86.30 Mbps, while ACLs and IP Verify moderately increased the TCP throughput to 18.48 Mbps and 15.67, respectively. On the other hand, Network Load Balancing and the Software Firewall only increased the TCP throughput from 0.19 Mbps (no defense) to 0.28 Mbps, and 0.25 Mbps respectively (Figure 6.7). The UDP throughput result showed that Hybrid Method and Threshold Limit significantly increased the UDP throughput from 38.7 Mbps (no defense) to 86.5 Mbps whereas Network Load Balancing and the Software Firewall only increased the TCP throughput from 38.7 Mbps (no defense) to 43.35 Mbps, and 39.10 Mbps respectively (Figure 6.8). The RTT result showed that Hybrid Method and Threshold Limit were the most effective defenses in this study. Both defenses could decrease the average RTT from 27.38ms (no defense) to 26.30ms and 26.39ms respectively. On the other hand, Network Load Balancing increased the RTT from 27.38ms (no defense) to 28.97ms (Figure 6.9). The reason is that this solution required the system resources to examine incoming packets and make load-balancing decisions, and therefore imposed an overhead on network performance. The Hybrid Method and Threshold Limit could reduce the CPU utilization during the attack from 24% (no defense) to 2%, while ACLs and IP Verify reduced the CPU utilization to 16% and 18%, respectively. On the other hand, the Software Firewall consumed more CPU utilization, i.e., 25%, than other the defenses.

The next chapter covers the evaluation of the UDP flood attack using Linux Ubuntu 13.

CHAPTER 7

EVALUATION OF UDP FLOOD ATTACK ON LINUX UBUNTU 13

This chapter covers the analysis of the results gathered from experiments using Linux Ubuntu 13. Section 7.1 covers the impact of a UDP flood attack on throughputs based on TCP and UDP applications. Section 7.2 shows the comparison of the computer performance during a UDP flood attack between Linux Ubuntu 13 and Windows Server 2012 using round-trip time, packet loss, CPU utilization, and jitter. Section 7.3 covers the analysis of six defense mechanisms, namely, Access Control Lists, Threshold Limit, Hybrid Defense, IP Verify, Network Load Balancing, and DDoS Software Firewall. Section 7.4 provides a summary of the comparison of the results of the flood defenses between Linux Ubuntu 13 and Windows Server 2012.

7.1 Impact of UDP Flood Attack on Throughputs

This section covers the impact of the UDP flood attack on throughputs before and during the attack. TCP and UDP were selected as the transmission protocols. The primary tools used in this study were Iperf and Hping3. The former was used to fill up the Ethernet link at 86.4 Mbps while Hping3, which was the attack tool, was used to generate attack traffic at 156.2 Mbps. To ensure high data accuracy, the test was repeated 30 times and result average and runs continued until the standard deviation of results was below 0.07% of the average. The evaluation process and measurement tools used in this chapter were discussed in Section 4.3.2.2 and Section 5.3. In addition, in order to be consistent and produce accurate data, all of the hardware used was kept identical from Chapter 6.

7.1.1 TCP Throughput

This section presents the impact of a UDP flood attack on throughputs between Linux Ubuntu 13 and Windows Server 2012. TCP is selected as the transmission protocol, which is used for webserver traffic. The Y axis shows the number of throughputs in megabits per second, while the X axis shows the length of the experiment in seconds.

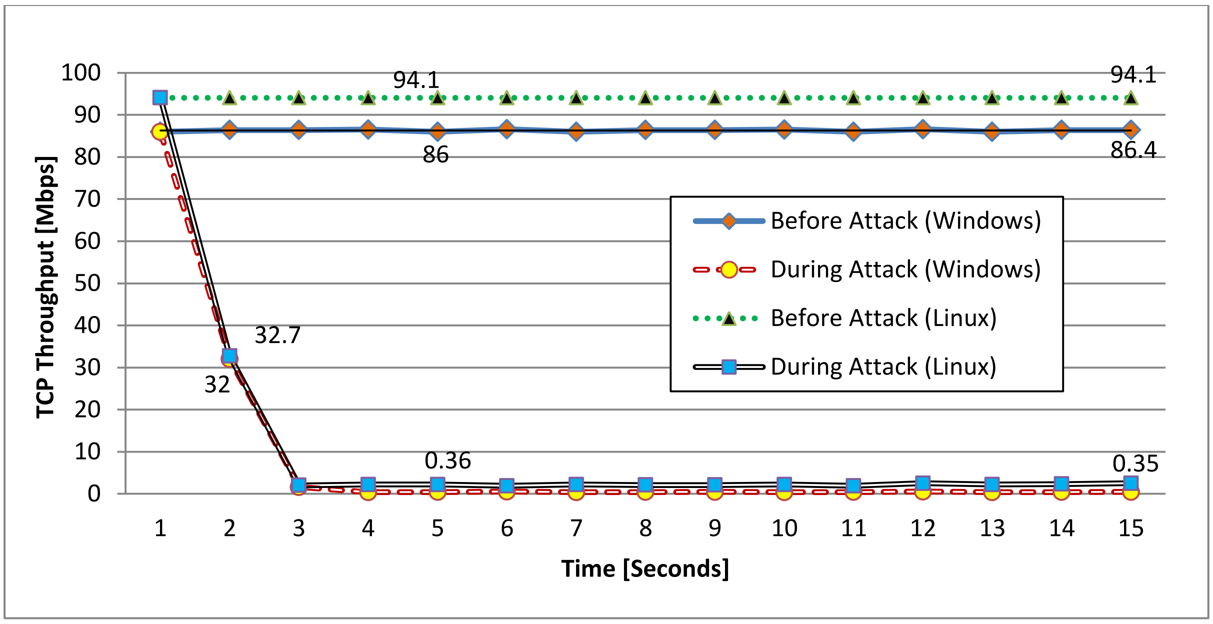


Figure 7.1: Comparison of TCP Throughputs Between Linux Ubuntu 13 and Windows Server 2012

Linux Ubuntu 13	Time (Seconds)	Throughput (Mbps)	Standard Deviation
Before Attack	All	94.10	0.010
During Attack	2	32.96	0.067
	3	0.39	0.072
	After 4 Seconds	0.36-0.45	-

Table 7.1: Average TCP Throughputs and Standard Deviation for 30 Runs Before and During Attack on Linux Ubuntu 13

Figure 7.1 illustrates the TCP throughput values between Linux and Windows platform. On the whole, Linux Ubuntu 13 outperformed Windows Server 2012 in terms of the number of throughputs before and during the attack. The result shows that the TCP throughput value before the attack on the Linux platform was constant at 94 Mbps, which was higher than Windows at about 8 Mbps. During the attack, however, the TCP throughput value on Linux platform significantly dropped from 94 Mbps to 32.96 Mbps within 2 seconds. Afterwards, it was stable at around 0.36 to 0.45 Mbps. In terms of Windows, the number of TCP throughputs reduced from 86 Mbps to 32 Mbps within 2 seconds. After that, it was constant at around 0.19 to 0.24Mbps.

7.1.2 UDP Throughput

This section presents the impact of a UDP flood attack on throughputs between Linux Ubuntu 13 and Windows Server 2012. UDP is selected as the transmission protocol, which is used for VoIP applications. The Y axis shows the number of throughputs in megabits per second, while the X axis shows the length of the experiment in seconds.

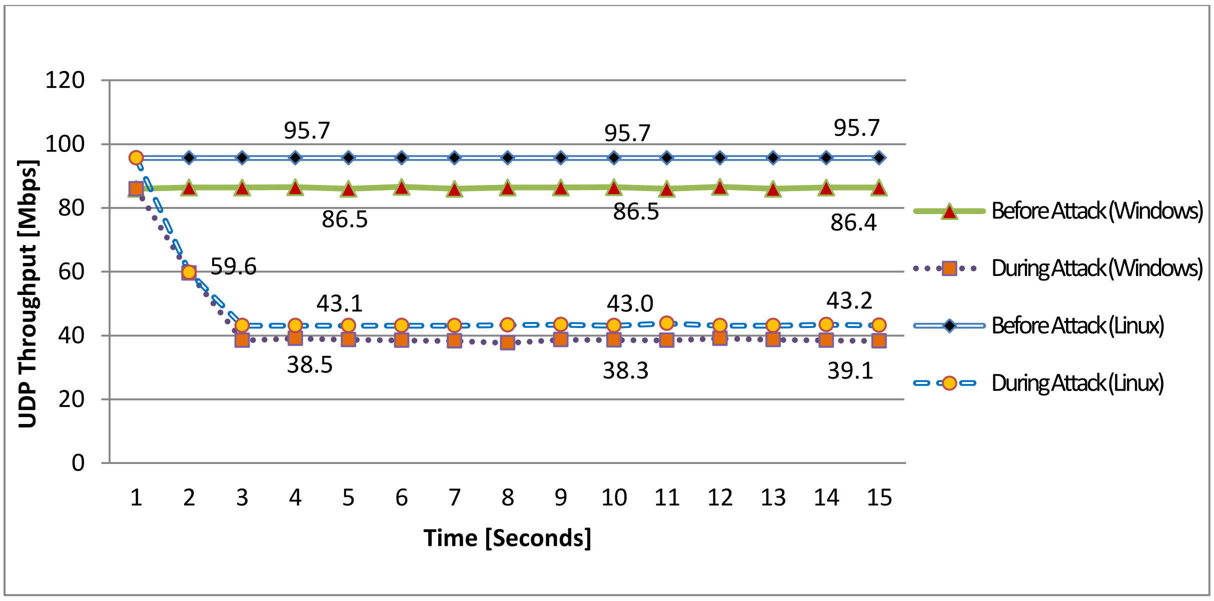


Figure 7.2: Comparison of UDP Throughputs Between Linux Ubuntu 13 and Windows Server 2012

Linux Ubuntu 13	Time (Seconds)	Throughput (Mbps)	Standard Deviation
Before Attack	All	95.70	0.010
During Attack	2	59.85	0.034
	3	43.14	0.010
	After 4 Seconds	42.9-43.8	-

Table 7.2: Average UDP Throughputs and Standard Deviation for 30 Runs Before and During Attack on Linux Ubuntu 13

Figure 7.2 illustrates the UDP throughput values between Linux and Windows platform. The result shows that the UDP throughput value before the attack on Linux platform was constant at 95.7 Mbps, which was higher than Windows at about 10Mbps. During the attack, however, the UDP throughput value on Linux platform dropped moderately from 95.7 Mbps to 59.85 Mbps within 2 seconds. Afterwards, it was stable at around 42.9 Mbps to 43.8 Mbps. In terms of Windows OS, the number of UDP throughputs reduced from 86.5 Mbps to 60.4Mbps within 2 seconds. After that, it was constant at around 38 to 39Mbps.

A plausible speculation that could explain why Linux Ubuntu 13 outperformed Windows 2012 is the way kernel network buffers are allocated and used by Linux platforms (Kolahi & Li, 2011). That is, Linux operating systems have a pre-allocation of fixed-sized memory buffers so that when a network application sends data, these buffers will be used to avoid the overhead associated with buffer allocations.

7.2 Impact of UDP Flood Attack on Linux Ubuntu 13 and Windows Server 2012

This section covers the impact of a UDP flood attack using Linux Ubuntu 13 and Windows Server 2012. The primary tools used in this study were the Webserver Stress Tool and Hping3. The former generated the connection request from users to the webserver assuming on average 10 users per second, while Hping3 was used to generate the attack traffic at 13000 packets per second (unless the attack packet rate and packet size are otherwise defined in the next sections). The monitoring PC used gathered DDoS parameters while legitimate or attack traffic was launched. To ensure high data accuracy, the test was repeated 30 times and data average and runs continued until standard deviation of results was below 0.07% of the average.

7.2.1 Round-trip Time

This section presents a comparison of the RTT results between Linux Ubuntu 13 and Windows Server 2012. The primary tool used in this study was TCPing, which was the tool used to gather the delay values of the victim computers.

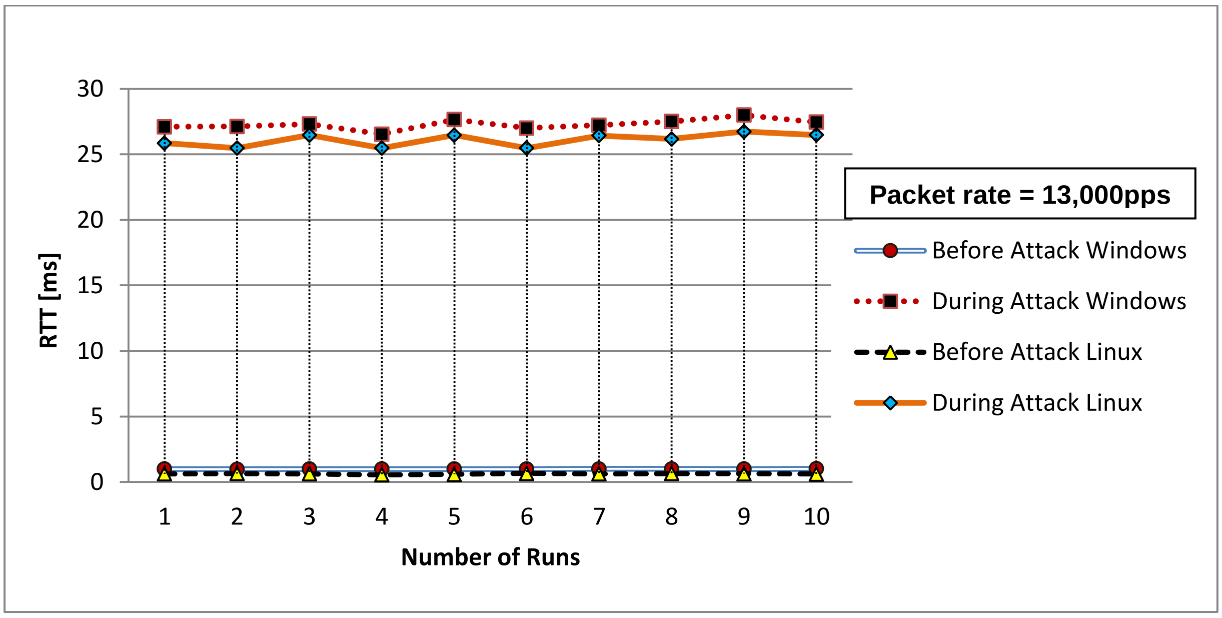


Figure 7.3: Comparison of RTT Results Between Linux Ubuntu 13 and Windows Server 2012

Operating System	Scenario	Average RTT	Standard Deviation
Windows	Before Attack	0.99ms	0.016
	During Attack	27.38ms	0.059
Linux	Before Attack	0.62ms	0.050
	During Attack	26.42ms	0.067

Table 7.3: Average RTT and Standard Deviation for 30 Runs Before and During Attack on Linux Ubuntu 13

Figure 7.3 illustrates the average of RTT for 30 runs before and during the UDP flood attack. On the whole, the result shows that Microsoft Server 2012 had higher delay values than Linux Ubuntu 13. Without the attack, the average RTT of Windows platform was 0.99ms, while the average RTT of Linux platform was 0.62ms. During the attack, the RTT of Windows Server 2012 went up significantly from 0.99 to 27.38ms, while the RTT of Linux Ubuntu 13 increased from 0.62 to 26.42ms. Since the hardware and monitoring tools used were identical from Chapter 6 and only the operating system was changed, it can be said, therefore, that Linux Ubuntu 13 withstood the UDP flood attack better than Windows Server 2012 in this study.

7.2.2 Packet Loss

This section shows a comparison of legitimate packet loss results between Linux Ubuntu 13 and Windows Server 2012. The primary tool used in this study was Iperf, which was the tool used to gather the packet loss values. The test was run 30 times using attack packet sizes ranging from 64 to 1518 Bytes, while the attack packet rate was constant at 13,000 packets per second. These tests generated attack traffic at 6.34 Mbps, 12.69 Mbps, 25.3 Mbps, 50.7 Mbps, 101.5 Mbps, 126.9 Mbps, and 150.5 Mbps respectively.

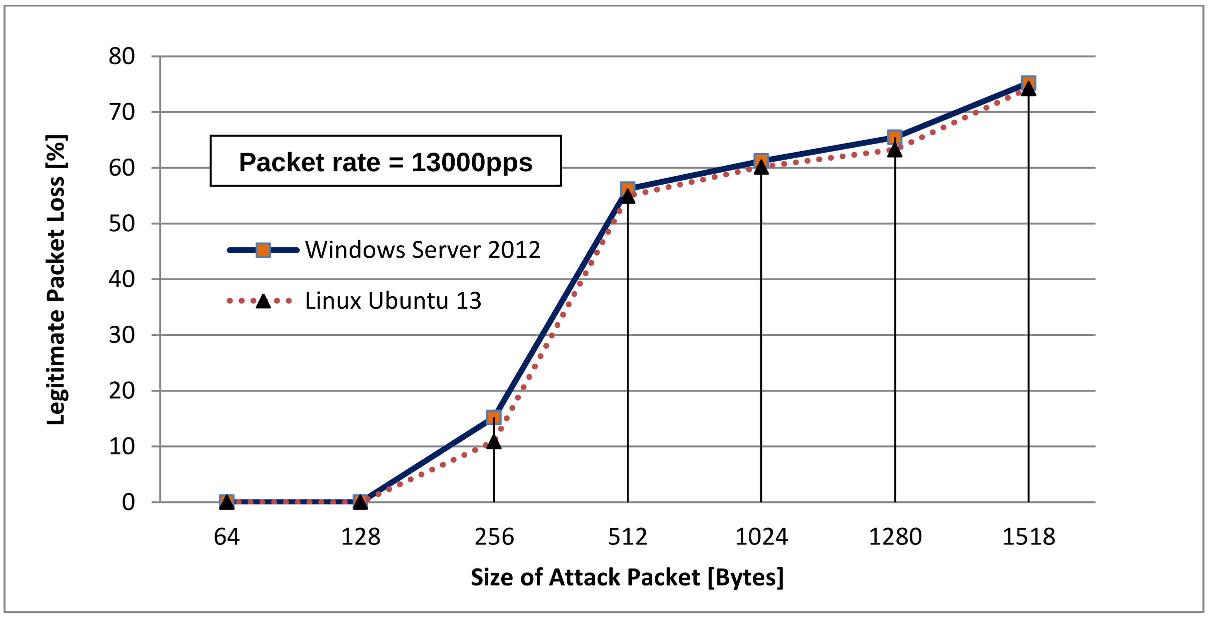


Figure 7.4: Comparison of Legitimate Packet Loss Results Between Linux Ubuntu 13 and Windows Server 2012

Operating System	Attack Packet Size	Packet Loss (%)	Standard Deviation
Windows	256	15.215	0.021
	512	56.170	0.042
	1024	61.215	0.023
	1280	65.466	0.047
	1518	75.200	0.031
Linux	256	10.897	0.019
	512	54.900	0.015
	1024	60.142	0.098
	1280	63.247	0.048
	1518	74.211	0.057

Table 7.4: Average Legitimate Packet Loss and Standard Deviation For 30 runs Before and During Attack on Linux Ubuntu 13

Figure 7.4 illustrates the average of packet loss values before and during the UDP flood attack. On the whole, the result shows that the legitimate packet loss value increased as the attack packet size increased. There was no packet loss on either operating system when using the attack packet size of 64, and 128 Bytes. However, on frame size 256 Bytes (25.3 Mbps), the packet loss on Windows Server 2012 went up moderately to 15.2% and 10.8% for Linux Ubuntu 13. Regarding frame size 512 Bytes (50.7 Mbps), the packet loss value of Windows and Linux platform significantly increased to 56.1% and 54.9%, respectively. On the largest frame size, 1518 Bytes (150.5 Mbps), the packet loss went up to 75.2% for Windows, and 74.2% for Linux.

7.2.3 CPU Utilization

This section presents a comparison of CPU utilization before and during a UDP flood attack between Linux Ubuntu 13 and Windows Server 2012. The primary tool used in this study was Microsoft Resource Monitor, which was the tool used to monitor the CPU usage on Windows Server 2012. For Linux operating system, we used the “top” command to find out the Linux CPU usage.

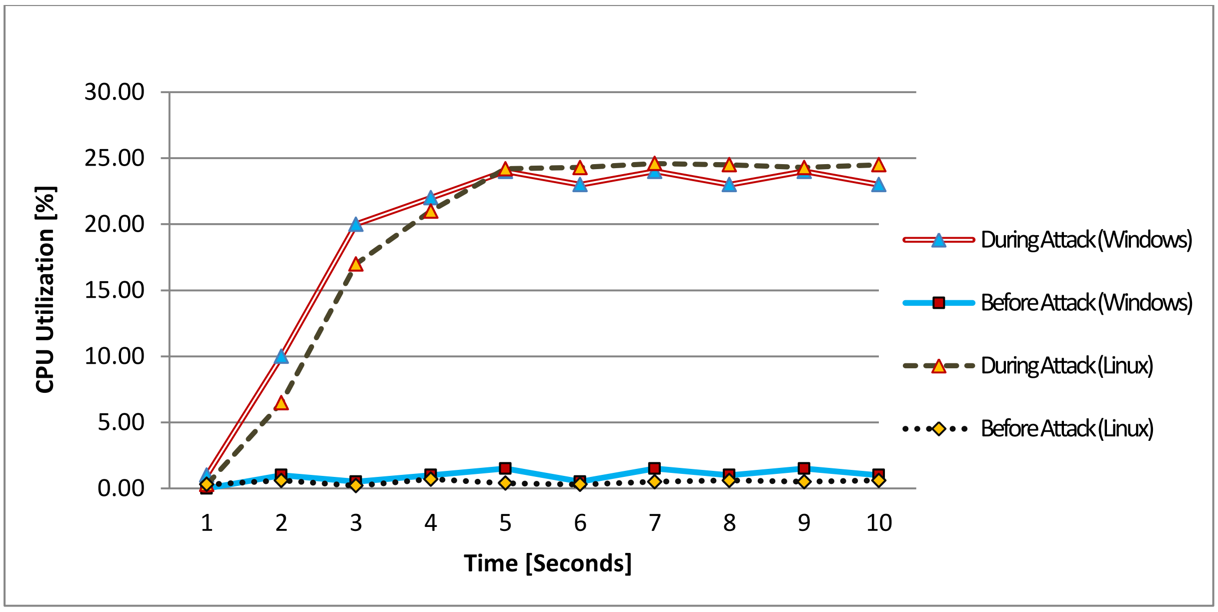


Figure 7.5: Comparison of CPU Utilization Results Between Linux Ubuntu 13 and Windows Server 2012

Scenario	Time (Seconds)	CPU Utilization (%)	Standard Deviation
During Attack (Windows)	2	9.36	0.150
	3	19.52	0.077
	4	21.28	0.080
	After 5 Seconds	19-24	-
During Attack (Linux)	2	9.36	0.073
	3	18.01	0.087
	4	20.9	0.059
	After 5 Seconds	23.5-24.9	-

Table 7.5: Average CPU Utilization and Standard Deviation for 30 Runs Before and During Attack on Linux Ubuntu 13

Figure 7.5 illustrates the CPU utilization before and during a UDP flood attack. In terms of Windows Server 2012, the CPU usage before the attack was constant at 1% to 2%. During the attack, however, the CPU utilization went up approximately 10% within 2 seconds and increased to 22% in 4 seconds. Afterwards, it fluctuated between 19% and 24%.

In terms of Linux Ubuntu 13, the result shows that the CPU usage before the attack was slightly lower than for Windows, which was around 0.3 to 0.7%. During the attack, Linux Ubuntu 13 demonstrated the better performance in terms of stability; the CPU utilization increased at approximately 6% within 2 seconds. Afterwards, it remained steady at 23.5% to 24.9%.

7.2.4 Jitter

This section presents a comparison of legitimate jitter results on UDP applications between Linux Ubuntu 13 and Windows Server 2012. The primary tool used in this study was Iperf, which was the tool used to gather the jitter values of the victim computers. The attack packet sizes used in this study were 64, 128, 256, 512, 1024, 1280, and 1518 Bytes, while the attack packet rate was constant at 13000 packets per second. The tests generated attack traffic at 6.34 Mbps, 12.69 Mbps, 25.3 Mbps, 50.7 Mbps, 101.5 Mbps, 126.9 Mbps, and 150.5 Mbps respectively.

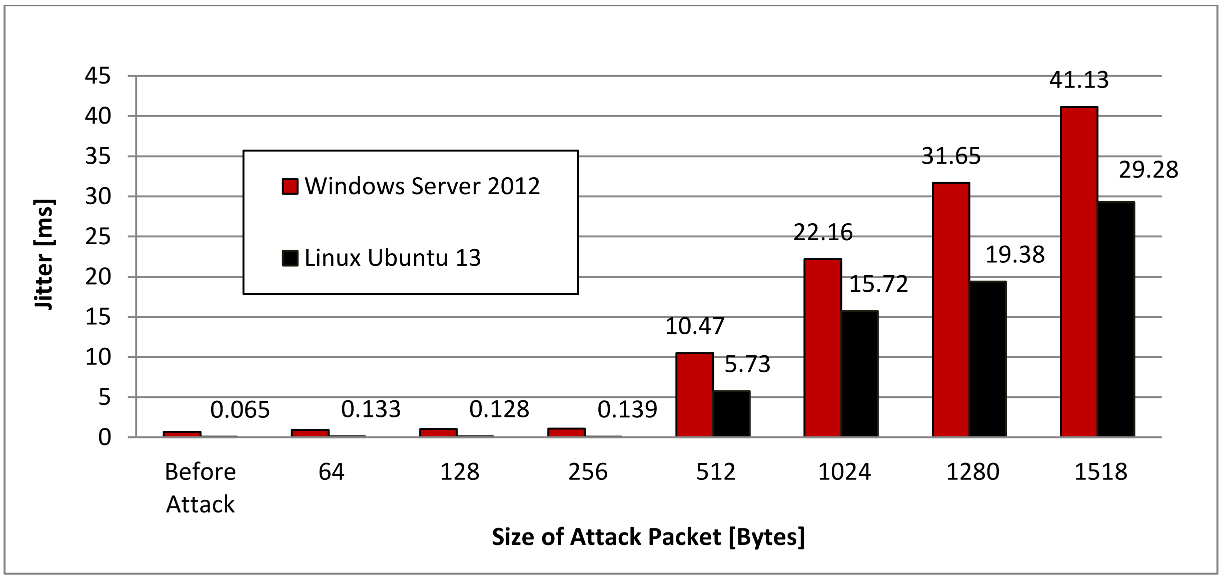


Figure 7.6: Comparison of Jitter Results Between Linux Ubuntu 13 and Windows Server 2012

Attack Packet Size (Bytes)	Jitter (ms)	Standard Deviation
Before Attack	0.065	0.015
64	0.133	0.018
128	0.128	0.027
256	0.139	0.021
512	5.73	0.045
1024	15.72	0.068
1280	19.38	0.054
1518	29.28	0.081

Table 7.6: Average Jitter and Standard Deviation for 30 Runs Before and During Attack on Linux Ubuntu 13

Figure 7.6 illustrates the average jitter value results on Linux and Windows platform. On the whole, Windows Server 2012 generated more jitter than Linux Ubuntu 13. In terms of Linux platform, the result shows that the legitimate jitter value before the attack was 0.065ms. On frame sizes 64, 128 and 256 Bytes (6.3 Mbps, 12.6 Mbps, and 25.3 Mbps), Linux generated very similar jitter values, which were 0.133ms, 0.128ms, and 0.139ms, respectively. The most noticeable feature of this graph is that the jitter value significantly increased when using attack packet sizes between 512 and 1518 Bytes (50.7 Mbps, 101.5 Mbps, 126.9 Mbps, and 150.5 Mbps). By using 512 Bytes (50.7 Mbps), the average jitter value went up from 0.06 to 5.73ms. The jitter value increased to 19.38ms if the attack packet size was 1024 Bytes (101.5 Mbps). On the largest frame size, 1518 Bytes (150.5 Mbps), the average jitter value increased significantly from 0.06 to 29.28ms.

7.3 Comparisons of Defense Mechanisms on Linux Ubuntu 13

This section provides an analysis of defense mechanisms against a UDP flood attack on Linux Ubuntu 13. There were six defenses used in this study, namely, Access Control Lists, Threshold Limit, Hybrid Defense, IP Verify, Network Load Balancing, and DDoS Software Firewall. The efficiency of each defense was evaluated by comparing throughputs, delay, packet loss, and CPU utilization values before and after using the defenses. To ensure high data accuracy, the test was repeated 30 times and result average and runs continued until standard deviation of results was below 0.07% of the average. The evaluation process and experimental set-up of defenses were discussed in Section 4.3.2.2 and Section 5.2.

7.3.1 TCP Throughput

This section covers the impact of a UDP flood attack on throughputs after using defenses. TCP was selected as the transmission protocol, which was used for webserver traffic. The primary tools used in this study were Iperf and Hping3. The former was used to fill up the Ethernet link at 86.4 Mbps, while Hping3, which was the attack tool, was used to generate attack traffic at 156.2 Mbps.

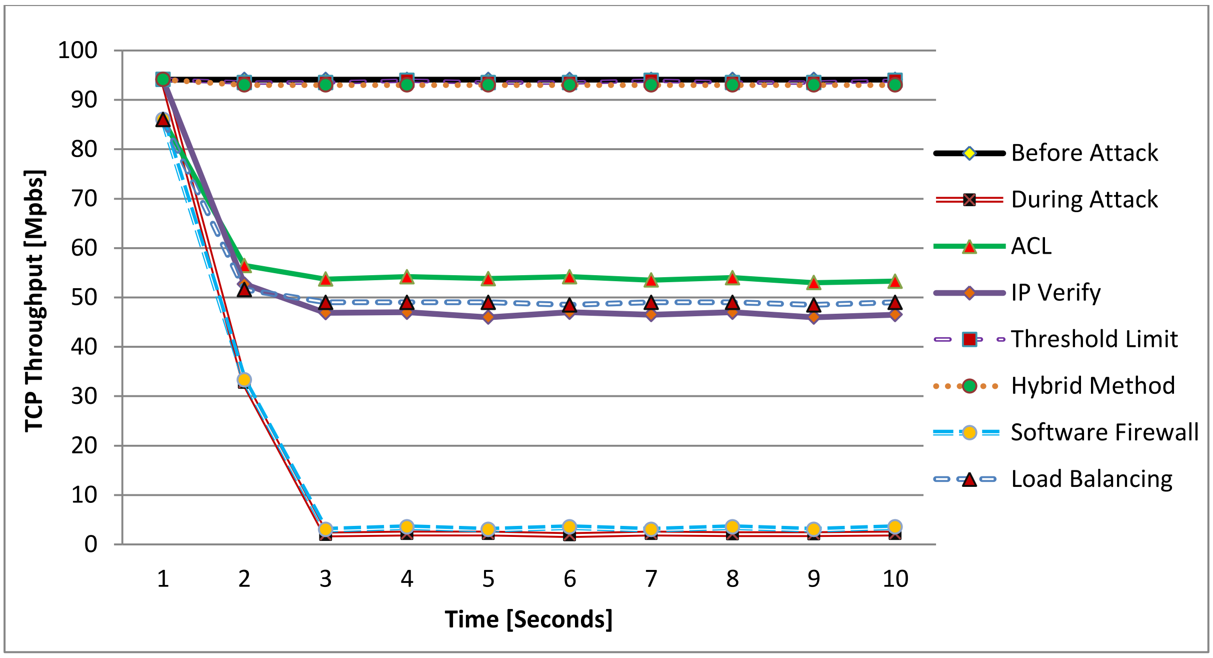


Figure 7.7: TCP Throughput Values After Using Defenses on Linux Ubuntu 13

Scenario	Defense	Throughput (Mbps)	Standard Deviation
Before Attack	**-**	94.1	0.003
During Attack	No Defense	0.36	0.146
	Hybrid Method	94.0	0.002
	Threshold Limit	94.0	0.001
	ACLs	53.37	0.016
	Load Balancing	47.39	0.011
	IP Verify	46.93	0.018
	Software Firewall	0.64	0.052

Table 7.7: Average TCP Throughput Values and Standard Deviation for 30 Runs After Using Defenses on Linux Ubuntu 13

Figure 7.7 presents the TCP throughput values after using six defenses. The result shows that the TCP throughput value before the attack was stable at 94.1 Mbps. This was higher than the TCP throughput on Windows Server 2012 at about 7.5 Mbps (as shown in Figure 6.7). During the attack, the TCP throughput on Linux platform significantly dropped from 94.1 Mbps to 0.36 Mbps, while the TCP throughput on Windows dropped from 86.60 Mbps to 0.19 Mbps.

The most effective defenses in this study were the Hybrid Method and Threshold Limit in which the number of throughput values before the attack and after using solutions were almost the same, at 94 Mbps. ACLs came in third, and increased the throughput value from 0.36 Mbps to 53.37 Mbps. This number was marginally higher than Network Load Balancing and IP Verify at 5.98 Mbps and 6.44 Mbps, respectively. The most ineffective defense mechanism in this study was the Software Firewall, which only increased the TCP throughput value from 0.36 Mbps to 0.64 Mbps.

7.3.2 UDP Throughput

This section covers the impact of a UDP flood attack on throughputs after using defenses. UDP was selected as the transmission protocol, which was used for VoIP traffic. The primary tools used in this study were Iperf and Hping3. The former was used to fill up the Ethernet link at 86.4 Mbps, while Hping3, which is the attack tool, generated the attack traffic at 156.2 Mbps.

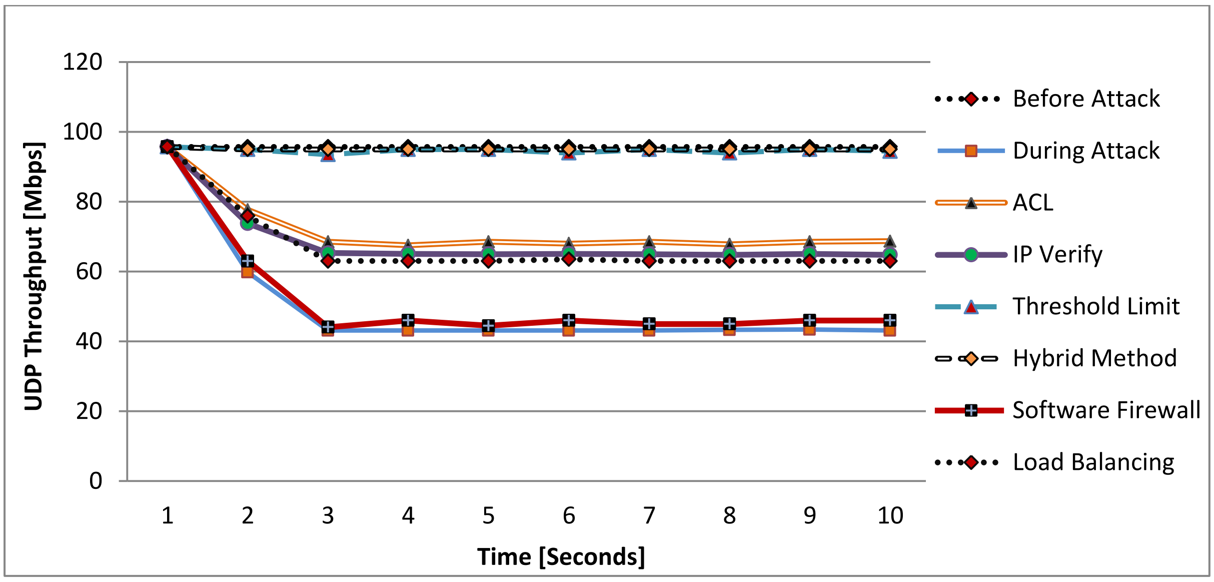


Figure 7.8: UDP Throughput Values After Using Defenses on Linux Ubuntu 13

Scenario	Defense	Throughput (Mbps)	Standard Deviation
Before Attack	-	95.7	0.004
During Attack	No Defense	43.04	0.010
	Hybrid Method	95.6	0.005
	Threshold Limit	95.6	0.003
	ACLs	68.55	0.007
	IP Verify	65.35	0.006
	Load Balancing	64.15	0.007
	Software Firewall	44.07	0.011

Table 7.8: Average UDP Throughput Values and Standard Deviation For 30 Runs After Using Defenses on Linux Ubuntu 13

Figure 7.8 presents a comparison of UDP throughput values after using six defenses on Linux Ubuntu 13. The result shows that the UDP throughput value before the attack was stable at 95.7 Mbps. This number was higher than the UDP throughput on Windows Server 2012 at about 9.1 Mbps (as shown in Figure 6.8). During the attack, this number dropped moderately from 95.70 Mbps to 43.04 Mbps, while the UDP throughput on Windows platform dropped from 86.60 Mbps to 38.7 Mbps.

In terms of defenses, the most effective solutions in this study were the Hybrid Method and the Threshold Limit. Both defenses significantly increased UDP throughput values from 43.04 Mbps to 95.6 Mbps. ACLs came in third, and increased the number of user throughputs from 43.04 Mbps to 68.55 Mbps. This number was marginally higher than IP Verify and Network Load Balancing at 3.2 Mbps, and 4.4 Mbps, respectively. The most ineffective defense in this study was the Software Firewall, which insignificantly increased the UDP throughput value from 43.04 Mbps to 44.07 Mbps.

7.3.3 Round-trip Time

This section presents the comparison of RTT before and after using six defenses on Linux Ubuntu 13. The primary tool used in this study was TCPing, which was the tool used to gather the delay values of the webserver.

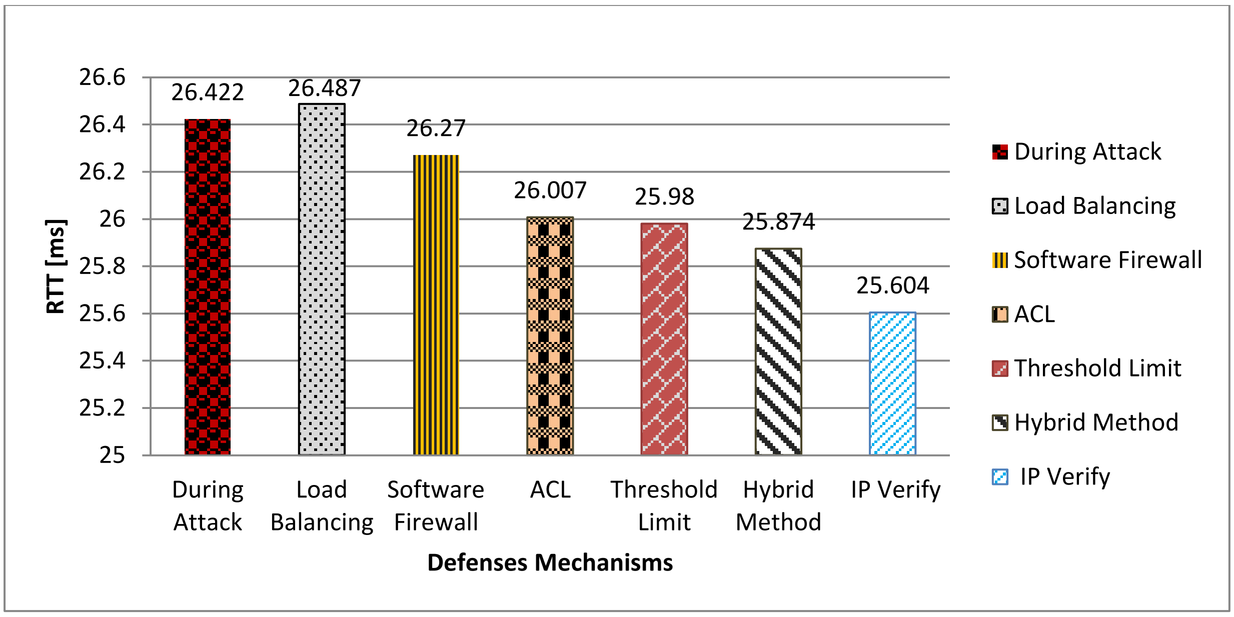


Figure 7.9: Round-trip Time After Using Defenses on Linux Ubuntu 13

Scenario	Defense	Average RTT (ms)	Standard Deviation
Before Attack	**-**	0.621	0.050
During Attack	No Defense	26.422	0.017
	Load Balancing	26.487	0.023
	Software Firewall	26.270	0.028
	ACLs	26.007	0.011
	Threshold Limit	25.980	0.011
	Hybrid Method	25.874	0.017
	IP Verify	25.604	0.018

Table 7.9: Average RTT and Standard Deviation for 30 Runs After Using Defenses on Linux Ubuntu 13

Figure 7.9 illustrates the average round-trip time for 30 runs after using DDoS defenses on Linux Ubuntu 13. On the whole, the result shows that Linux Ubuntu 13 produced delay values lower than Windows Server 2012 (as shown in Figure 6.9). The RTT value before the attack on the Linux platform was 0.62ms, and increased to 26.42ms during the attack.

In terms of defenses, IP Verify was the most effective defense in this study, as it reduced the average RTT from 26.42ms to 25.60ms. It can be noted that the most effective defense for Windows Server 2012 was the Hybrid Method, which reduced the RTT from 27.38ms to 26.30ms (Figure 6.9).

In terms of Linux Ubuntu 13, the Threshold Limit, Hybrid Method, and the Software Firewall could decrease the average RTT during the attack to 25.98ms, 25.87ms, and 26.27ms, respectively. The most inefficient defense in this study was Network Load Balancing, which increased the RTT from 26.422ms to 26.487ms. This is caused by the switching and routing process as the router needs to examine incoming packets and make load-balancing decisions, and therefore imposes an overhead on network performance (Microsoft, 2014).

7.3.4 Packet Loss

This section presents the comparison of the packet loss values before and after using six defenses on Linux Ubuntu 13. The primary tool used in this study was Iperf, which was the tool used to gather the packet loss values.

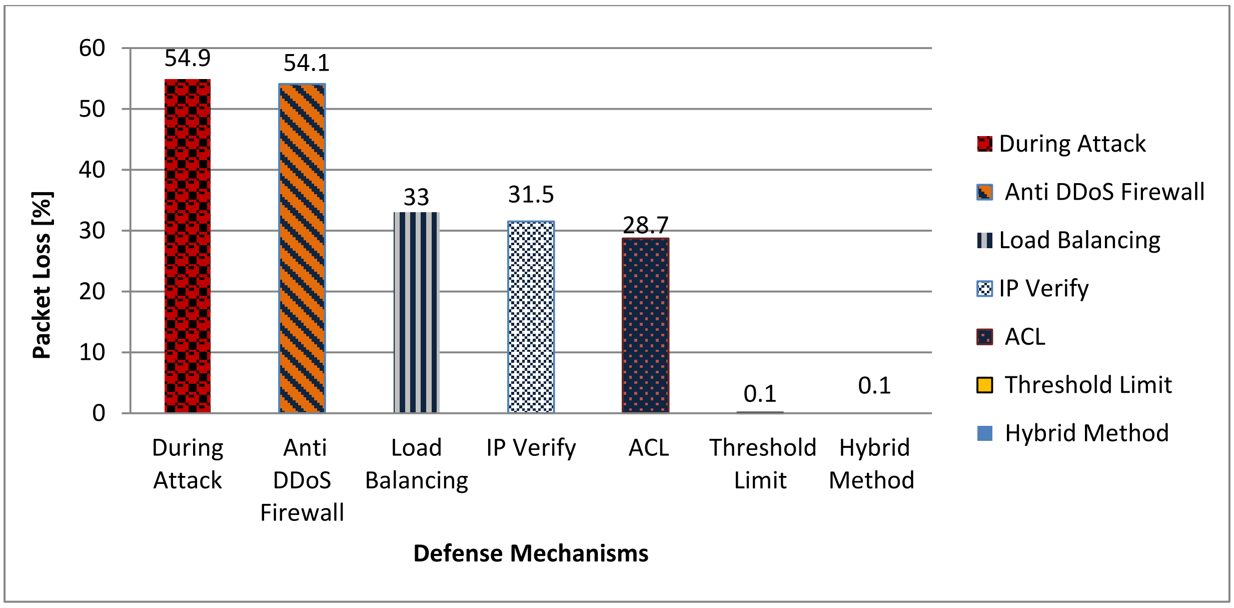


Figure 7.10: Packet Loss Values After Using Defenses on Linux Ubuntu 13

Scenario	Defense	Packet Loss (%)	Standard Deviation
Before Attack	-	0	-
During Attack	No Defense	54.9	0.015
	Anti-DDoS Firewall	54.1	0.008
	IP Verify	31.5	0.015
	Load Balancing	33.0	0.016
	ACLs	28.7	0.014
	Threshold Limit	0.1	-
	Hybrid Method	0.1	-

Table 7.10: Average Packet Loss and Standard Deviation For 30 Runs After Using Defenses on Linux Ubuntu 13

Figure 7.10 presents a comparison of the legitimate packet loss values after using DDoS defenses. The result shows there was no packet loss before the attack. During the attack, however, the packet loss went up to 54.9%. This number was slightly lower than the packet loss on Windows Server 2012 at about 1% (as shown in Figure 6.10).

In terms of defenses, the most effective defenses in this study were the Hybrid Method and Threshold Limit, which significantly reduced the packet loss value from 54.9% to 0.1%. ACLs, Network Load Balancing, and IP Verify moderately decreased the packet loss to 28.7%, 33%, and 31.5%, respectively. The most ineffective defense in this study was the software firewall, which reduced the packet loss value from 54.9% to 54.1%.

Comparing the packet loss results between the two operating systems, it can be observed that the packet losses after using solutions on Linux Ubuntu 13 and Windows Server 2012 were almost the same. However, the results showed that ACLs and IP Verify outperformed on Windows platform, while Network Load Balancing and the software firewall outperformed on Linux platform (as compared to Figure 6.10).

7.3.5 CPU Utilization

This section shows the CPU utilization before and after using defenses on Linux Ubuntu 13. The primary tool used in this study was Microsoft Resource Monitor, which was the tool used to monitor the CPU usage on the server.

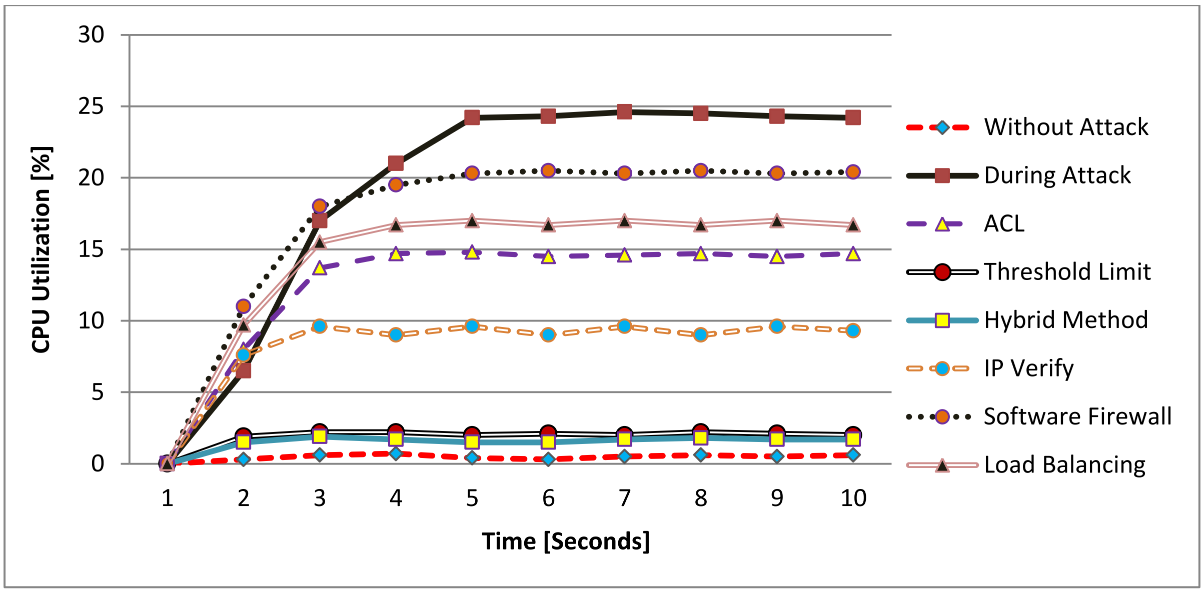


Figure 7.11: CPU Utilization After Using Defenses on Linux Ubuntu 13

Scenario	Defense	CPU Utilization (%)
Before Attack	**-**	0.3-0.7
During Attack	No Defense	23-25
	Software Firewall	19-21
	Load Balancing	15-18
	ACLs	13-15
	IP Verify	8-10
	Threshold Limit	3
	Hybrid Method	1-2

Table 7.11: Average CPU Utilization For 30 Runs After Using Defenses on Linux Ubuntu 13

Figure 7.11 illustrates the average CPU utilization after using DDoS defenses on Linux Ubuntu 13. The results show that the CPU usage before the attack was stable at around 0.3 to 0.7%, which was slightly lower than on Windows Server 2012 at about 1 to 2% (as shown in Figure 6.11). During the attack, the CPU usage of Linux platform fluctuated between 23% and 25%.

In terms of defenses, the most effective defense in this study was the Hybrid Method, which reduced the CPU usage from 25% to 2%. This was similar to the Threshold Limit, which decreased the CPU utilization to 3%. IP Verify came in third, and reduced the CPU usage from 25% to 10%. In terms of ACLs, the CPU usage went down from 25% to 15%. This figure was similar to Network Load Balancing, which reduced the server's CPU usage to 18%. The most inefficient defense in this study was the Software Firewall, which slightly reduced the CPU utilization from 25% to 21%.

7.4 Comparison of Defenses Against UDP Flood Attack between Linux and Windows

This section shows a summary of result comparison of the flood defenses between Linux Ubuntu 13 and Windows Server 2012. The metrics used for comparison are TCP throughput (Section 7.4.1), UDP throughput (Section 7.4.2), round-trip time (Section 7.4.3), packet loss value (Section 7.4.4), and CPU utilization (Section 7.4.5).

7.4.1 TCP Throughput after Using Defenses

In terms of TCP throughputs using Windows Server 2012, the most effective defenses were the Hybrid Method and Threshold Limit, which increased the throughput value from 0.19 Mbps (no defenses) to 86.30 Mbps. ACLs and IP Verify moderately increased the TCP throughput from 0.19 Mbps to 18.48 Mbps and 15.67 Mbps, respectively (Figure 6.7). The most effective defenses for Linux Ubuntu 13 were also the Hybrid Method and Threshold Limit, which significantly increased the throughput value from 0.36 Mbps (no defenses) to 94 Mbps. ACLs and IP Verify moderately increased the TCP throughput from 0.36 Mbps to 53.37 Mbps and 46.93 Mbps, respectively (Figure 7.7).

7.4.2 UDP Throughput after Using Defenses

The most effective defenses for Windows Server 2012 were the Hybrid Method and Threshold Limit, which increased the throughput value from 38.7 Mbps (no defenses) to 86.40 Mbps. ACLs and IP Verify moderately increased the UDP throughput from 38.7 Mbps to 66.01 Mbps and 61.96 Mbps, respectively. On the other hand, Network Load Balancing and Software Firewall could only increase the UDP throughput from 38.7 Mbps to 43.35 Mbps and 39.10 Mbps, respectively (Figure 6.8). In terms of Linux results, the Hybrid Method and Threshold Limit could significantly increase the UDP throughput from 43.04 Mbps (no defenses) to 95.6 Mbps. ACLs and IP Verify moderately increased the UDP throughput from 43.04 Mbps to 68.55 Mbps and 65.35 Mbps, respectively (Figure 7.8).

7.4.3 Round-trip Time after Using Defenses

The Hybrid Method and Threshold Limit were the most effective defenses for Windows Server 2012. These solutions decreased the average RTT from 27.38ms to 26.30ms and 26.39ms, respectively. ACLs and IP Verify reduced the RTT from 27.38ms to 26.67ms and 26.74ms, respectively. On the other hand, Network Load Balancing resulted in the highest RTT, which was approximately 28.97ms (Figure 6.9). In terms of Linux results, IP Verify outperformed other defenses, which reduced the RTT value from 26.42ms to 25.60ms. The Threshold Limit, Hybrid Method, and Software Firewall could decrease the average RTT during the attack from 26.42ms to 25.98ms, 25.87ms, and 26.27ms, respectively. The most inefficient defense for Linux OS was Network Load Balancing, which increased the RTT from 26.42ms to 26.48ms (Figure 7.9).

7.4.4 Packet Loss after Using Defenses

The most effective defenses for Windows Server 2012 were the Hybrid Method and Threshold Limit, which significantly reduced the packet loss value from 56.1% to 0.1%. IP Verify and ACLs moderately decreased the packet loss from 56.1% to 28.5%, and 23.1%, respectively. On the other hand, Network Load Balancing and Software Firewall insignificantly reduced the packet loss values from 56.1% to 51.1%, and 55.2%, respectively (Figure 6.10). In terms of Linux results, the Hybrid Method and Threshold Limit outperformed the other defenses, both of which significantly reduced the packet loss value from 54.9% to 0.1%. ACLs, Network Load Balancing, and IP Verify moderately decreased the packet loss to 28.7%, 33%, and 31.5%, respectively. The most ineffective defense in this study was Software Firewall, which only reduced the packet loss value from 54.9% to 54.1% (Figure 7.10).

7.4.5 CPU Utilization after Using Defenses

In terms of Windows results, the Hybrid Method and Threshold Limit outperformed the other defenses, and significantly reduced the CPU usage from 24% to 2%. ACLs, Load Balancing, and IP Verify moderately decreased the server’s CPU usage from 24% to 16%, 17%, and 18%, respectively. Interestingly, the Software Firewall consumed more CPU utilization than the other defenses, at around 25% (Figure 6.11). The most effective defenses for Linux platform were the Hybrid Method and Threshold Limit, both of which reduced the CPU usage from 25% to 2%. IP Verify, ACLs, and Load Balancing moderately decreased the victim computer’s CPU usage to 8%, 13%, and 15%, respectively. On the other hand, the Software Firewall slightly reduced the CPU utilization from 25% to 21% (Figure 7.11).

7.5 Chapter Summary

This chapter presented the results on computer performance during a UDP flood attack for Linux Ubuntu 13 and the results compared with Windows Server 2012. This chapter also evaluated six defenses, namely, Access Control Lists, Threshold Limit, Hybrid Defense, IP Verify, Network Load Balancing, and DDoS Software Firewall.

The results showed that the performance of Linux Ubuntu 13 was reduced during a UDP flood attack. The TCP throughput before the attack was constant at 94 Mbps, and dropped significantly to 0.36 Mbps during the attack (Figure 7.1). The UDP throughput before the attack was 95.7 Mbps, and it dropped moderately to 43.9 Mbps during the attack (Figure 7.2). The RTT before the attack was 0.62ms, and it increased to 26.42ms during the attack (Figure 7.3). Before the attack, the CPU utilization was constant at 0.3% and went up to 25% during the attack (Figure 7.5).

After using the defenses, the results showed that the performance of Linux Ubuntu 13 was increased. The Hybrid Method and Threshold Limit were the most effective defenses against a UDP flood attack in most of the studies, whereas the Software Firewall and Network Load Balancing were the least effective defenses. The Hybrid Method and Threshold Limit could increase the TCP throughput from 0.36 Mbps to 94 Mbps (Figure 7.7), and increased the UDP throughput from 43.04 Mbps to 95.6 Mbps (Figure 7.8). The RTT result showed that IP Verify could reduce the RTT from 26.42ms to 25.60ms, while the Hybrid Method reduced the RTT from 26.42 to 25.87ms (Figure 7.9). The Hybrid Method and Threshold Limit could significantly reduce the CPU usage from 25% (during the attack) to 2%, whereas IP Verify, ACLs, Network Load Balancing, and the Software Firewall only moderately reduced the CPU utilization between 8% and 21% (Figure 7.11).

When comparing the operating systems' performance, it can be said that Linux Ubuntu 13 could withstand a UDP flood attack better than Windows Server 2012. The result showed that the TCP throughput before the attack on Linux platform was constant at 94 Mbps, which was higher than Windows at about 8 Mbps. During the attack, the throughput values of both operating systems significantly dropped to 0.36 Mbps and 0.19 Mbps, respectively (Figure 7.1). The UDP throughput before the attack on Linux platform was constant at 95.7 Mbps, which was higher than Windows at about 10Mbps. During the attack, the UDP throughput of Linux and Windows OS dropped moderately to 43.9 Mbps and 38.3 Mbps, respectively (Figure 7.2). Without the attack, the average RTT of Windows platform was 0.99ms, which was higher than Linux platform at 0.36ms. During the attack, the average RTT of Windows Server 2012 went up to 27.38ms, while the RTT of Linux Ubuntu 13 increased to 26.42ms

(Figure 7.3). Windows Server 2012 utilized the CPU resource more than Linux Ubuntu 13 at approximately 0.7% (before the attack). During the attack, Linux Ubuntu 13 demonstrated the better performance in terms of stability; the CPU utilization remained steady at around 23.5% while the CPU utilization of Windows Server 2012 fluctuated between 19% and 24%.

When comparing the defenses between the two operating systems, it can be said that the Hybrid Method and Threshold Limit were the most effective defenses against a UDP flood attack in most studies, whereas the Software Firewall and Network Load Balancing were the least effective defenses (IP Verify, and ACLs were average defenses). The Hybrid Method and Threshold Limit could increase the TCP throughput from 0.36 Mbps to 94 Mbps (for Linux Ubuntu 13), and from 0.19 Mbps to 86.4 Mbps (for Windows Server 2012). In terms of UDP throughputs, the Hybrid Method and Threshold Limit increased the UDP throughput from 43.04 Mbps to 95.6 Mbps (Linux Ubuntu 13), and from 38.7 Mbps to 86.5 Mbps (Windows Server 2012). The RTT results after using the defenses showed that the Hybrid Method was the most effective defense for Windows OS, which reduced the RTT from 27.38ms to 26.30ms (Figure 6.9), while IP Verify was the most effective defense for Linux OS, which reduced the RTT from 26.42ms to 25.60ms (Figure 7.9). The Hybrid Method and Threshold Limit could reduce the CPU usage of Windows OS during an attack from 24% to 2% (Figure 6.11), and from 25% to 2% for Linux Ubuntu 13 (Figure 7.11).

The next chapter covers the discussion of the results obtained in Chapter 6 and Chapter 7.

CHAPTER 8

SUMMARY, CONCLUSIONS, AND FUTURE WORKS

A revolution occurred in the world of computers and communication with the advent of the Internet. This technology has become increasingly important to our current society; it has changed our way of communication, business modes, and made information publicly accessible quickly and easily, and put it within easy reach. However, along with the advantages of the Internet, there are also disadvantages. There is no absolute security in the Internet world, and hackers can use the Internet to launch many different types of attacks on a targeted network, one of which is DDoS attacks.

In this research, new results were obtained to evaluate the impact of a UDP flood attack, which was the type of DDoS attack used in this study. Tests were conducted on computers using the latest version of Windows and Linux platforms, namely, Windows Server 2012 and Linux Ubuntu 13. This research also produced new evaluation results on various defense mechanisms such as Access Control Lists, Threshold Limit, IP Verify, Hybrid Method, Network Load Balancing, and Software Firewall.

The results showed that the performance of Windows Server 2012 (e.g., throughputs, round-trip time, packet loss, CPU utilization, and jitter) reduced during the UDP flood attack. The TCP throughput before the attack was constant at 86 Mbps, and it significantly dropped to 0.19 Mbps during the attack. The UDP throughput before the attack was constant at 86 Mbps, and it went down to 38.5 Mbps during the attack. The RTT before the attack was 0.99ms, and it increased to 27.38ms during the attack. The packet loss result showed that there was no legitimate packet loss before the attack. During the attack, the packet loss value went up to 56.1%, 61.2%, 65.4%, and 75.2% when using the attack packet sizes of 512 Bytes, 1024 Bytes, 1280 Bytes, and 1512 Bytes, respectively. The CPU utilization of Windows Server 2012 before the attack was constant at 2%. During the attack, the CPU usage went up to 24%. The jitter value before the attack was 0.66ms. During the attack, the jitter values went up slightly when using the attack packet sizes between 64, 128, and 256 Bytes (at 0.92ms, 1.04ms, and 1.061ms, respectively), and it went up significantly to 41.13ms when using the attack packet size of 1518 Bytes.

After using defenses on Windows Server 2012, the results showed that the performance of Windows OS was increased. The Hybrid Method and Threshold Limit were the most effective defenses against the UDP flood attack in all studies, whereas, the Software Firewall and Network Load Balancing were the least effective defenses (IP Verify, and ACLs were average defenses). The Hybrid Method and Threshold Limit could increase the TCP throughput from 0.19 Mbps (no defense) to 86.30 Mbps. ACLs and IP Verify moderately increased the TCP throughput to 18.48 Mbps and 15.67 Mbps, respectively. On the other hand, Network Load Balancing and the Software Firewall only increased the TCP throughput from 0.19 Mbps (no defense) to 0.28 Mbps, and 0.25 Mbps, respectively. The UDP throughput result showed that the Hybrid Method and Threshold Limit significantly increased the UDP throughput from 38.7 Mbps (no defense) to 86.5 Mbps, while ACLs and IP Verify moderately increased the UDP throughput to 66.01 Mbps and 61.96 Mbps, respectively. On the other hand, Network Load Balancing and the Software Firewall only increased the UDP throughput from 38.7 Mbps (no defense) to 43.35 Mbps and 39.1 Mbps, respectively. The RTT result showed that the Hybrid Method and Threshold Limit were the most effective defenses in this study. Both solutions could decrease the average RTT from 27.38ms (no defense) to 26.30ms and 26.39ms, respectively. ACLs and IP Verify could reduce the RTT to 26.67ms, and 26.74ms, respectively. On the other hand, Network Load Balancing increased the RTT from 27.38ms (no defense) to 28.97ms. The packet loss result showed that the Hybrid Method and Threshold Limit could significantly reduce the packet loss value from 56.1% (no defense) to 0.1%. IP Verify and ACLs moderately decreased the packet loss value to 28.5%, and 23.1%, respectively. The most ineffective defenses in this study were Network Load Balancing and the Software Firewall. Both defenses reduced the packet loss values insignificantly from 56.1% (no defense) to 51.1%, and 55.2%, respectively. The Hybrid Method and Threshold Limit could decrease the CPU utilization during the attack from 24% (no defense) to 2%. ACLs and IP Verify could reduce the CPU utilization to 16% and 18%, respectively. On the other hand, the Software Firewall consumed the CPU utilization more than other defenses, which was 25%.

The results for Linux Ubuntu 13 showed that the performance of Linux OS reduced during the UDP flood attack. The TCP throughput before the attack was constant at 94.1 Mbps, and dropped significantly to 0.36 Mbps during the attack. The UDP throughput value before the attack was constant at 95.7 Mbps and went down to 42.9 Mbps during the attack. The RTT before the attack was 0.62ms, and increased to 26.42ms during the attack. The packet loss result showed that there was no legitimate packet loss before the attack. During the attack the packet loss value went up to 54.9%, 60.1%, 63.2%, and 74.2% when using the attack packet sizes of 512 Bytes, 1024 Bytes, 1280 Bytes, and 1512 Bytes, respectively. The CPU utilization result showed that the CPU usage of Linux OS before the attack was constant at 0.3%, and went up to 25% during the attack. The jitter value before the attack was 0.065ms and went up significantly to 29.28ms. During the attack, the jitter values went up slightly when using the attack packet sizes between 64, 128, and 256 Bytes (at 0.133, 0.128ms, and 0.139ms, respectively) and went up significantly to 29.28ms when using the attack packet size of 1518 Bytes.

After using defenses on Linux Ubuntu 13, the results showed that the performance of Linux OS was increased. Hybrid Method and Threshold Limit were the most effective defenses against the UDP flood attack in most studies (except the RTT result) whereas the Software Firewall was the least effective defense (IP Verify, and ACLs were average defenses). The Hybrid Method and Threshold Limit could increase the TCP throughput from 0.36 Mbps (no defense) to 94 Mbps. ACLs and IP Verify moderately increased the TCP throughput to 53.37 Mbps and 46.93 Mbps, respectively. On the other hand, the Software Firewall only increased the TCP throughput from 0.19 Mbps (no defense) to 0.64 Mbps. The UDP throughput result showed that the Hybrid Method and Threshold Limit could increase the UDP throughput from 43.04 Mbps (no defense) to 95.6 Mbps. ACLs, IP Verify, and Network Load Balancing moderately increased the UDP throughput from 43.04 Mbps (no defense) to 68.55 Mbps, 65.35 Mbps, and 64.15 Mbps, respectively. On the other hand, the Software Firewall insignificantly increased the UDP throughput from 43.04 Mbps (no defense) to 44.07 Mbps. The RTT result showed that IP Verify was the most effective defense in this study, as it reduced the average RTT from 26.42ms (no defense) to 25.60ms, while the Hybrid Method reduced the RTT to 25.87ms. The Threshold Limit and ACLs could reduce the RTT from 26.42 (no defense) to 25.98ms, and 26.0ms, respectively. On the other hand, Network Load Balancing increased the RTT from 26.42ms (no defense) to 26.48ms. The packet loss result showed that the Hybrid Method and Threshold Limit could significantly reduce the packet loss value from 54.9% (no defense) to 0.1%. IP Verify and ACLs moderately decreased the packet loss value to 31.5% and 28.7%, respectively. The most ineffective defense in this study was the Software Firewall which reduced the packet loss value insignificantly from 54.9% (no defense) to 54.1%. The CPU utilization result showed that the Hybrid Method and Threshold Limit could decrease the CPU utilization during the attack from 25% (no defense) to 2%. IP Verify and ACLs could reduce the CPU utilization to 8% and 13%, respectively. On the other hand, the Software Firewall insignificantly reduced the CPU usage from 25% to 21%.

When comparing the performance between the two operating systems, it can be said that Linux Ubuntu 13 could withstand the UDP flood attack better than Windows Server 2012. The result showed that the TCP throughput before the attack on Linux platform was constant at 94 Mbps, which was higher than Windows by 8 Mbps. During the attack, the throughput values of both operating systems dropped significantly to 0.36 Mbps (Linux) and 0.19 Mbps (Windows). The UDP throughput before the attack on Linux platform was constant at 95.7 Mbps, which was higher than Windows by 10 Mbps. During the attack, the UDP throughput of Linux and Windows OS dropped moderately to 43.9 Mbps and 38.3 Mbps, respectively. Without the attack, the average RTT of Windows platform was 0.99ms, which was higher than Linux platform at 0.36ms. During the attack, the average RTT of Windows Server 2012 went up to 27.38ms, while the RTT of Linux Ubuntu 13 increased to 26.42ms. Windows Server 2012 utilized the CPU resource before the attack at about 2%, while Linux Ubuntu 13 utilized the CPU resource at about 0.5%. During the attack, Linux Ubuntu 13 demonstrated better performance in terms of stability; the CPU utilization remained steady at around 23.5%, while the CPU utilization of Windows Server 2012 fluctuated between 19% and 24%.

When comparing defenses between the two operating systems, it can be said that the Hybrid Method and Threshold Limit were the most effective defenses against the UDP flood attack for both operating systems, whereas the Software Firewall was the least effective defense (IP Verify, and ACLs were average defenses). The Hybrid Method and Threshold Limit defenses could increase the TCP throughput from 0.36 Mbps to 94 Mbps (for Linux Ubuntu 13) and from 0.19 Mbps to 86.4 Mbps (for Windows Server 2012). When comparing the UDP throughput, the Hybrid Method and Threshold Limit could increase the UDP throughput from 43.04 Mbps to 95.6 Mbps (Linux Ubuntu 13), and from 38.7 Mbps to 86.5 Mbps (Windows Server 2012). It can be noted that throughputs increased more for Linux OS when all defenses were implemented. The RTT results after using defenses showed that the Hybrid Method was the most effective defense for Windows OS, and reduced the RTT from 27.38ms to 26.30ms, while IP Verify was the most effective defense for Linux OS, and reduced the RTT from 26.42ms to 25.60ms. The Hybrid Method and Threshold Limit could reduce the CPU usage of Windows OS from 24% (no defense) to 2%, and from 25% (no defense) to 2% for Linux Ubuntu 13. Based on all the results, it can be concluded that Linux Ubuntu 13 withstood the UDP flood attack better than Windows Server 2012, while the Hybrid Method and Threshold Limit were the most effective defenses against the UDP flood attack for both Windows and Linux platforms.

8.1 Future Work

This study provided a testbed study of the impact of a UDP flood attack on computers using the latest versions of Windows and Linux platforms, namely, Window Server 2012 and Linux Ubuntu 13. This study also compared the performance of six defense mechanisms that can be adopted on networks to defend against the attack. The following is future studies that can be done:

8.1.1 Different Types of DDoS Attacks and Defenses

There are other attacks and defenses that can be evaluated in a testbed environment. For example, ARP Request Attacks, ICMP Flood Attacks, Smurf Attacks, and TCP SYS Flood Attacks. These attacks can be evaluated on the latest version of operating systems such as Windows 8.1, Linux Fedora 20, Linux GNOME 3.13, and Red Hat Enterprise Linux 7.

8.1.2 IPv6 Router Advertisement Attack and Defenses

We used a testbed based on IPv4 in this thesis, as IPv6 is being implemented; the testbed can be set up using IPv6 and various attacks and defense mechanisms evaluated. For example, IPv6 Router Advertisement (RA) Attack can be evaluated. The attack occurs when the attacker floods the network with a large number of Router Advertisement packets and forcing operating systems to create IPv6 addresses in response to every packet it receives. This will cause 100% CPU usage on the victim computer, preventing to process other application requests.

8.1.3 DDoS Attacks on Client with Mobility

A mobile phone can also be a target of DDoS attacks (Hossain, Atiquzzaman, & Ivancic, 2011). Further study can include impact of DDoS attacks and defenses on a mobile client.

APPENDIX

Appendix A: Hardware Specifications

Router

Cisco 2811	Specification
Data Link Protocol	Fast Ethernet and Ethernet
Management Protocol	SNMP3
Network/Transport Protocol	IPsec
Ports	2 router ports/9 switching ports
RAM	DDR SDRAM 256MB/768MB (Maximal)
Flash Memory	64MB/256MB(Maximal)
Voltage Required	AC 120/230V
Width	17.2 Inches
Height	1.8 Inches
Weight	14.1 lbs.
Features	Firewall protection, VPN support, MPLS support, hardware encryption, and Quality of Service (QoS)

Table 1: Cisco 2811 Specifications

Switch

Cisco SG 200-08	Specification
Switching Capacity	13.6 Gigabits per second
CPU Memory	32 MB
Packet Buffer	4 Mb
Flash Memory	8 MB
Security	802.1X
Ports	8 Gigabit Ethernet
Standards	Fast Ethernet and Ethernet
Dimensions (Weight x Height x Depth)	4.45 x 1.06 x 5.12 Inches
Voltage Required	100V-240V
Features	Port grouping, Quality of Service, VLAN, and Voice VLAN

Table 2: Cisco SG 200-08 Specifications

REFERENCES

Anderson, T., Roscoe, T., & Wetherall, D. (2004). Preventing Internet Denial-of-Service with Capabilities. *ACM SIGCOMM Computer Communication Review, 34*(1), 39-44. doi: 10.1145/972374.972382

Anstee, D., Bussiere, D., & Sockrider, G. (2012). *Arbor Special Report: Worldwide Infrastucture Security Report*. Retrieved from http://pages.arbornetworks.com/rs/arbor/images/wisr2012_en.pdf

Arora, K., Kumar, K., & Sachdeva, M. (2011. Characterizing DDoS Attack Distributions from Emulation Based Experiments on DETER Testbed. *Proceedings of the International Conference on Advanced Computing, Networking and Security.* (pp. 541-550). Surathkal, India.

Backtrack. (2013). Introduction of BackTrack Linux. Retrieved from http://www.backtrack-linux.org

Bai, Y., & Yang, Y. (2006, September). *An Approximate Performance Analysis and Measurement of the Equivalent Model of Parallel Queues for a Web Cluster with a Low Rejection*. Paper presented at the 14th IEEE International Conference on Networks, Singapore. doi: 10.1109/ICON.2006.302659

BalanceNG. (2014). The Software Load Balancer. Retrieved from http://www.inlab.de/balanceng/

Beethink. (2014). Anti DDoS Guardian: Protect Your Business from DDoS Attacks. Retrieved from http://www.anti-ddos.net/

Bogdanoski, M., & Risteski, A. (2011). Wireless Network Behavior under ICMP Ping Flood DoS Attack and Mitigation Techniques. *Journal of Communication Networks and Information Security, 3*(1), 17-24. Retrieved from http://www.ijcnis.org

Bradner, S., & McQuaid, J. (1999). "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, March 1999.

Bruce, H. (2011). *Packet Guide to Core Network Protocols*, California, USA: O'Reilly Media.

Case, J., & Light, G. (2011). Emerging Methodologies in Engineering Education Research. *Journal of Engineering Education, 100*(1), 186-210. doi: 10.1002/j.2168-9830.2011.tb00008.x

Chaba, Y., Singh, Y., & Aneja, P. (2009). Performance Analysis of Disable IP Broadcast Technique for Prevention of Flooding-Based DDoS Attack in MANET. *Journal of Networks, 4*(3), 178-183. doi: 10.4304/jnw.4.3.178-183

Chang, R. (2002). Defending Against Flooding-Based Distributed Denial-of-Service Attacks. *IEEE Communications Magazine, 40*(10), 42-51. doi: 10.1109/MCOM.2002.1039856

Charvat, J. (2003). *Project Management Methodologies: Selecting, Implementing and Supporting Methodologies and Processes for Projects*. New Jersey: John Wiley & Sons, Inc.

Chen, Y., Hwang, K., & Ku, W. (2007, August). *Distributed Change-Point Detection of DDoS Attacks: Experimental Results on DETER Testbed*. Paper presented at the DETER Community Workshop on Cyber Security Experimentation and Test, Massachusetts. Retrieved from http://www.dl.acm.org

Ciampa, M. (2012). *Security+ Guide to Network Security Fundamentals* (4 ed.). Baston, MA: Course Technology.

Cisco. (2013a). Catalyst Switches for Microsoft Network Load Balancing Configuration Example. Retrieved from http://www.cisco.com/en/US/products/hw/switches/ps708/products_configuration_example09186a0080a07203.shtml

Cisco. (2013b). Configuring Committed Access Rate. Retrieved from http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfcar_ps1835_TSD_Products_Configuration_Guide_Chapter.html

Cisco. (2013c). Understanding Unicast Reverse Path Forwarding. Retrieved from http://www.cisco.com/web/about/security/intelligence/unicast-rpf.html

Cisco. (2013d). Access Control Lists: Overview and Guidelines. Retrieved from http://www.cisco.com/c/en/us/td/docs/ios/12_2/security/configuration/guide/fsecur_c/scfacls.html

David, D. (2007). Prevent IP spoofing with the Cisco IOS. Retrieved from http://www.techrepublic.com/article/prevent-ip-spoofing-with-the-cisco-ios

Douligeris, C., & Mitrokotsa, A. (2004). DDoS Attacks and Defense Mechanisms: Classification and State-of-the-Art. *Journal of Computer and Telecommunications Networking, 44*(5), 643-666. doi: 10.1016/j.comnet.2003.10.003

Erickson, J. (2008). *Hacking the Art of Exploitation* (2 ed.). San Francisco, CA: William Pollock.

Exist, E., & Postel, J. (1981). “Internet Standard”, RFC 791, September 1981.

Forouzan, A. (2000). *TCP/IP: Protocol Suite* (1 ed.). New Delhi, India: Tata McGraw-Hill Publishing Company Limited.

Galtsev, A., & Sukhov, A. (2011, August). *Network Attack Detection at Flow Level*. Paper presented at the 4th Conference on Smart Spaces, Petersburg. doi:10.1007/978-3-642-22875-9_30

Garber, L. (2000). Denial-of-service Attacks Rip the Internet. *IEEE Computer Society 33*(4), 12-17. doi: 10.1109/MC.2000.839316

Gates, C., Collins, M., Duggan, M., Kompanek, A., & Thomas, K. (2004, November). *More Netflow Tools: For Performance and Security*. Paper presented at the Conference on System Administration, Atlanta. Retrieved from http://www.dl.acm.org

Gupta, B., Joshi, R., & Misra, M. (2010). Distributed Denial of Service Prevention Techniques. *Journal of Computer and Electrical Engineering, 2*(2), 268-276.
Hping. (2014). Hping3 Security Tool. Retrieved from http://www.hping.org

Hossain, M., Atiquzzaman, M., & Ivancic, W. (2011, March). *Security Vulnerabilities and Protection Mechanisms of Mobility Management Protocols.* Paper presented at the IEEE Aerospace Conference, Big Sky, Montana. doi: 10.1109/AERO.2011.5747350

Hutt, A., Hoyt, D., & Bosworth, S. (2002). *Computer Security Handbook* (4 ed.). New York: John Wiley & Sons, Inc.

Kaur, D., Sachdeva, M., & Kumar, K. (2012). Study of DDoS Attacks Using DETER Testbed. *Journal of Computing and Business Research, 3*(2), 1-13.

Kavisankar, L., & Chellappan, C. (2011). CNoA: Challenging Number Approach for Uncovering TCP SYN Flooding Using SYN Spoofing Attack. *Journal of Network Security and Its Applications, 3*(5), 191-202. doi:10.5121/ijnsa.2011.3515

Kolahi, S.S., Li, P. (2011). Evaluating IPv6 in Peer-to-Peer 802.11n Wireless LANs. *IEEE Internet Computing*, *15*(4), 70-74. doi: 10.1109/MIC.2011.89

Kolahi, S.S, Rico, C., & Hong, C. (2013, November). *Bandwidth-IPSec Security Trade-off in IPv4 and IPv6 in Windows 7 Environment.* Paper presented at the Second International Conference on Future Generation Communication Technology, London. doi: 10.1109/FGCT.2013.6767214

Kumar, S. (2007, July). *Smurf-based Distributed Denial of Service (DDoS) Attack Amplification in Internet*. Paper presented at the International Conference on Internet Monitoring and Protection, San Jose, CA. doi: 10.1109/ICIMP.2007.42

Kumar, S., Azad, M., Gomez, O., & Valdez, R. (2006, February). *Can Microsoft's Service Pack2 (SP2) Security Software Prevent SMURF Attacks.* Paper presented at the Advanced International Conference on Telecommunications. doi: 10.1109/AICT- ICIW.2006.60

Kumar, S., & Gomez, O (2010). Denial of Service Due to Direct and Indirect ARP Storm Attacks in LAN Environment. *Journal of Information Security, 1*(2), 88-94. doi: 10.4236/jis.2010.12010

Kurose, J., & Ross, W. (2010). *Computer Networking: A Top-Down Approach* (5 ed.). Boston, MA: Pearson Education.

Lad, D., Alghalbi, A., & Ahmed, S. (2013). "Comparison of Various Defense Mechanisms Against Distributed Denial of Service Attacks". 1st Semester Project Report. Dept of Computing, Unitec Institute of Technology.

Le, A., Boutaba, R., & Shaer, E. (2008, September). *Correlation-Based Load Balancing for Network Intrusion Detection and Prevention Systems*. Paper presented at the 4th International Conference on Security and Privacy in Communication Networks, Istanbul, Turkey. doi: 10.1145/1460877.1460880

Lemon, J. (2002, February). *Resisting SYN Flooding DoS Attacks with a SYN Cache*. Paper presented at the BSD Conference 2002 on BSD Conference, California. Retrieved from http://www.dl.acm.org

Loshin, P. (2003). *TCP/IP Clearly Explained* (4 ed.). San Francisco, California: Elsevier Science.

Lu, W., Gu, W., & Yu, S. (2009). One-Way Queuing Delay Measurement and Its Application on Detecting DDoS Attack. *Journal of Network and Computer Applications, 32*(2), 367-376. doi: 10.1016/j.jnca.2008.02.018

Marti, P., Fuertes, J. M., Fohler, G., & Ramamritham, K. (2001, December). *Jitter Compensation for Real-Time Control Systems*. Paper presented at the 22nd IEEE Real-Time Systems Symposium. doi: 10.1109/REAL.2001.990594

Masoud, B., Shahram, J., & Gholam, S. (2012). Defense against SYN-Flood Denial of Servial of Service Attacks Based on LeamingAutomata. *Journal of Computer Science, 9*(3), 514-520.

McCabe, J. (2010). *Network Analysis, Architecture, and Design* (3 ed.). Burlington, MA Elsevier Science.

Microsoft. (2014). Network Load Balancing Technical Overview. Retrieved from http://technet.microsoft.com/en-us/library/bb742455.aspx

Mirkovic, J., Hussain, A., Fahmy, S., Reiher, P., & Thomas, R. (2009). Accurately Measuring Denial of Service in Simulation and Testbed Experiments. *IEEE Transactions, 6*(2), 81-95. doi: 10.1109/TDSC.2008.73

Mirkovic, J., Sonia, F., Reiher, P., & Thomas, K. (2009, March). *How to Test DoS Defenses.* Paper presented at the Cybersecurity Applications & Technology Conference For Homeland Security, Washington. doi: 10.1109/CATCH.2009.23

Mohd, G., & Rosilah, H. (2011). Flooding Distributed Denial of Service Attacks-A Review. *Journal of Computer Science, 7*(8), 1218-1223. Retrieved from http://www.thescipub.com

Mohd, Z., Idris, Y., Hussain, K., Stiawan, D., & Awan, K. (2011, November). *Protocol Share Based Traffic Rate Analysis (PSBTRA) for UDP Bandwidth Attack*. Paper presented at the International Conference on Informatics Engineering and Information Science, Kuala Lumpur. doi: 10.1007/978-3-642-25327-0_24

Muijs, D. (2011). *Doing Quantitative Research in Education with SPSS* (2 ed.). London: SAGE Publications Ltd.

Openmaniak. (2009). Iperf. Retrieved from http://openmaniak.com/iperf.php

Orebaugh, A., Ramirez, G., Rurke, J., Morris, G., Pesce, L., & Wright, J. (2007). *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. Massachusetts: Syngress Publishing, Inc.

Oriyano, S., & Gregg, M. (2010). *Hacker Techniques, Tools, and Incident Handling*. Massachusetts: Jones & Bartlett Learning.

Pack, G., Yoon, J., Collins, E., & Estan, C. (2006, August). *On Filtering of DDoS Attacks Based on Source Address Prefixes*. Paper presented at the Securecomm and Workshops, Baltimore. doi: 10.1109/SECCOMW.2006.359537

Ping over a TCP Connection (2013). Retrieved from http://www.elifulkerson.com/projects/tcping.php

Rahman, J., Saha, S., & Hasan, F. (2012, December). *A New Congestion Control Algorithm for Datagram Congestion Control Protocol (DCCP) Based Real-Time Multimedia Applications*. Paper presented at the International Conference on Electrical and Computer Engineering, Dhaka. doi: 10.1109/ICECE.2012.6471605

Rajesh, S. (2013). Protection from Application Layer DDoS Attacks for Popular Websites. *International Journal of Computer and Electrical Engineering, 5*(6), 555-558. doi: 10.7763/IJCEE.2013.V5.771

Rao, S. (2011). *Denial of Service Attacks and Mitigation Techniques: Real Time Implementation with Detailed Analysis*. Retrieved from http://www.sans.org

Rui, X., Li, M., & Ling, Z. (2009, August). *Defending against UDP Flooding by Negative Selection Algorithm Based on Eigenvalue Sets*. Paper presented at the International Conference on Information Assurance and Security, Xi'an. doi: 10.1109/IAS.2009.280

Shrivastava, G., Sharma, K., & Rai, S. (2010, December). *The Detection & Defense of DoS & DDoS Attack: A Technical Overview*. Paper presented at the International Conference on Computer Engineering and Technology, New Delhi. Retrieved from http://jietjodhpur.com

Singh, A., & Juneja, D. (2010). Agent Based Preventive Measure for UDP Flood Attack in DDoS Attacks. *International Journal of Engineering Science and Technology, 2*(8), 3405-3411. Retrieved from http://www.ijest.info

Sosinsky, B. (2009). *Networking Bible.* Indiana: Wiley Publishing Inc.

Stephen, M., & Ruby, B. (2004, September). *Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures*. Paper presented at the 17th International Conference on Parallel and Distributed Computing Systems, New Jersey. Retrieved from http://palms.ee.princeton.edu

Subramani, R. (2011). Denial of Service Attacks and Mitigation Techniques: Real Time Implementation with Detailed Analysis. Retrieved from http://www.sans.org/reading-room/whitepapers/detection/denial-service-attacks-mitigation-techniques-real-time-implementation-detailed-analysi-33764?show=denial-service-attacks-mitigation-techniques-real-time-implementation-detailed-analysi-33764&cat=detection

Vidya, S., & Bhaskaran, R. (2011). ARP Storm Detection and Prevention Measures. *International Journal of Computer Science Issues, 8*(2), 456-460. Retrieved from http://www.ijcsi.org

Wang, J. (2009). *Computer Network Security Theory and Practice*. Lowell, Massachusetts: Springer.

Webstress. (2014). Website Performance, Stress, and Load Testing. Retrieved from http://www.paessler.com/webstress

Wireshark. (2014). Introduction of Wireshark. Retrieved from http://www.wireshark.org

Xie, Y., & Yu, S. (2009). Monitoring the Application-Layer DDoS Attacks for Popular Websites. *IEEE Transactions on Networking, 17*(1), 15-25. doi: 10.1109/TNET.2008.925628

Yaar, A., Perrig, A., & Song, D. (2004, May). *SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks*. Paper presented at the IEEE Symposium on Security and Privacy. doi: 10.1109/SECPRI.2004.1301320

York, D. (2010). *Seven Deadliest Unified Communications Attacks*. Massachusetts: Syngress Publishing Inc.