Denial-of-Service Attacks using Socket Programming

by

Aditya Mandeep Vakani(01FB16EEC020), Akshar Deepankar Athreya(01FB16EEC028), Anurag Vinayak Muttur (01FB16EEC042), Ashwin Kumar Singh(01FB16EEC052)

Department of Electronics and Communication Engineering, PES University, Bangalore 560085

Supervised by

Professor M. Rajasekar

Department of Electronics and Communication Engineering, PES University, Bangalore 560085

GitHub Repository:

https://www.github.com/akshar-Athreya97/CN-Project

Introduction

This project demonstrates a Denial-of-Service (DoS) Attack from one end system to another with the help of Python and Socket Programming. A Denial-of-Service attack is a cyber-attack in which the attacker seeks to make a machine or network resource unavailable to its intended users by temporarily disrupting services of a victim connected to the internet. In this project we have demonstrated different types of DoS attacks - SYN Flood attack and UDP Flood attack using Socket Programming, RST Flood attacks and ICMP Flood attack using hping3 command line tool. The incoming packets have been captured and analyzed using Wireshark which is a free and open-source network packet analyzer. The effects of the attacks on the computer have been shown using Windows Task Manager.

The two main types of DoS attacks are classified as:

1. **Bandwidth Depletion:**
   This type of attack involves an attacker sending large amounts of traffic to a victim's system , usually causing the system to slow down, crash or preventing access to it by legitimate users, also known as packet flooding. The following two types have been demonstrated:
   a. **UDP Flood**
   b. **ICMP Flood**
2. **Resource Depletion:**
   This type of attack involves the attacker sending packets that misuse network protocol communications or are malformed, leading to network resources being tied up and none left for legitimate users.Two common resource depletion attacks that have been shown are:
   a. **TCP SYN flood**
   b. **TCP RST flood**

Software Used

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python's simple, easy to learn syntax emphasizes readability and therefore reduces cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. In this project we've mainly made use of the Socket module to write our code.

Wireshark

Wireshark is a free and open-source packet analyzer which is used for network troubleshooting, analysis, protocol development and for educational purposes. It is a cross platform application that using pcap to capture packets. Wireshark "understands" the structure of different networking protocols and can parse and display the fields with their meanings as specified by different networking protocols.

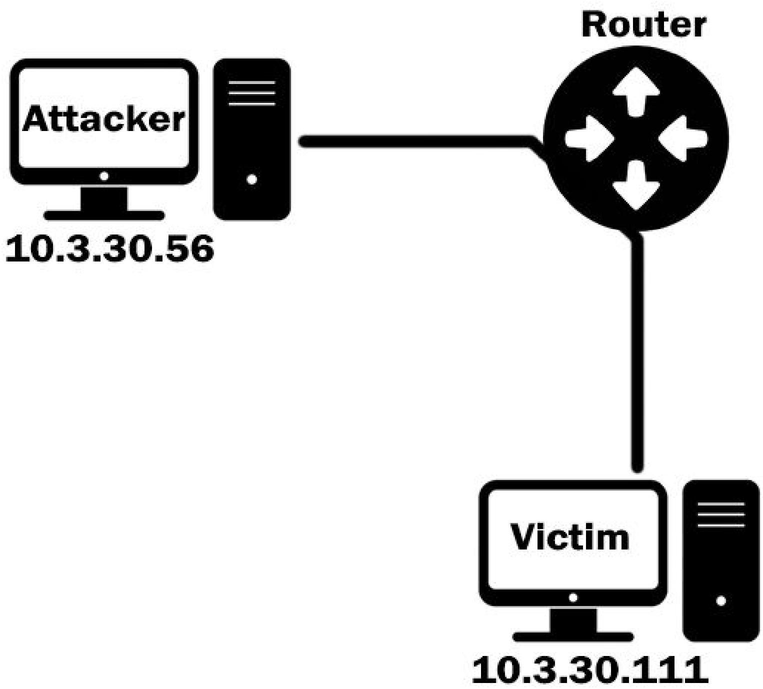
Photoshop

Graphic designing software which has been used here to make illustrations and label diagrams.

The Setup:

We will be using 2 end systems on a private network.

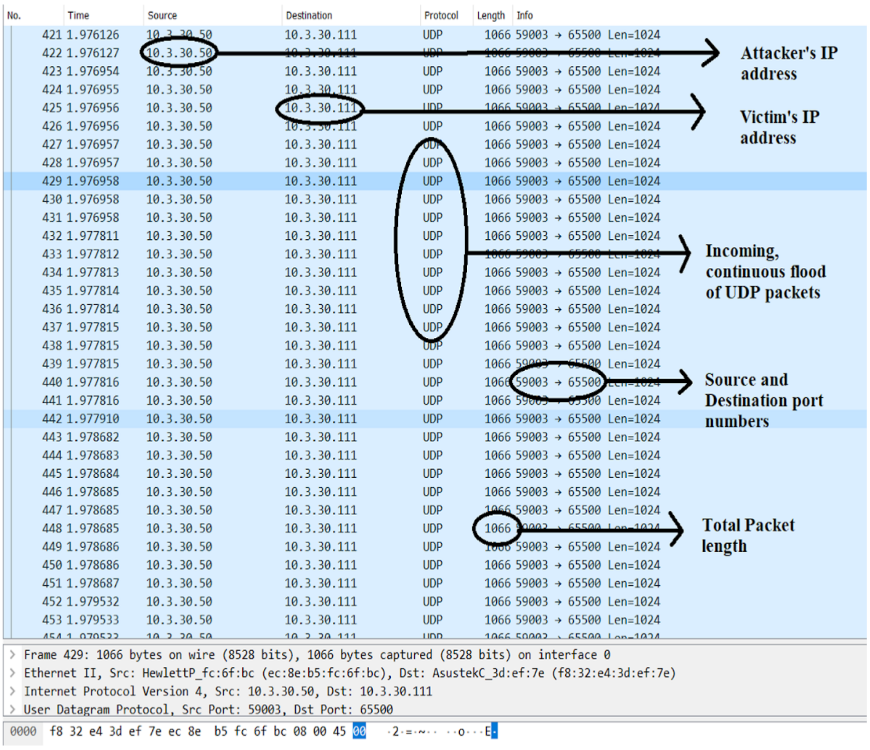
Victim's system IP during test -> *10.3.30.111 and 10.3.30.56* (**Windows 10 64bit)**
Attacking system IP during test -> *10.3.30.50* (**Ubuntu 16.04LTS)**



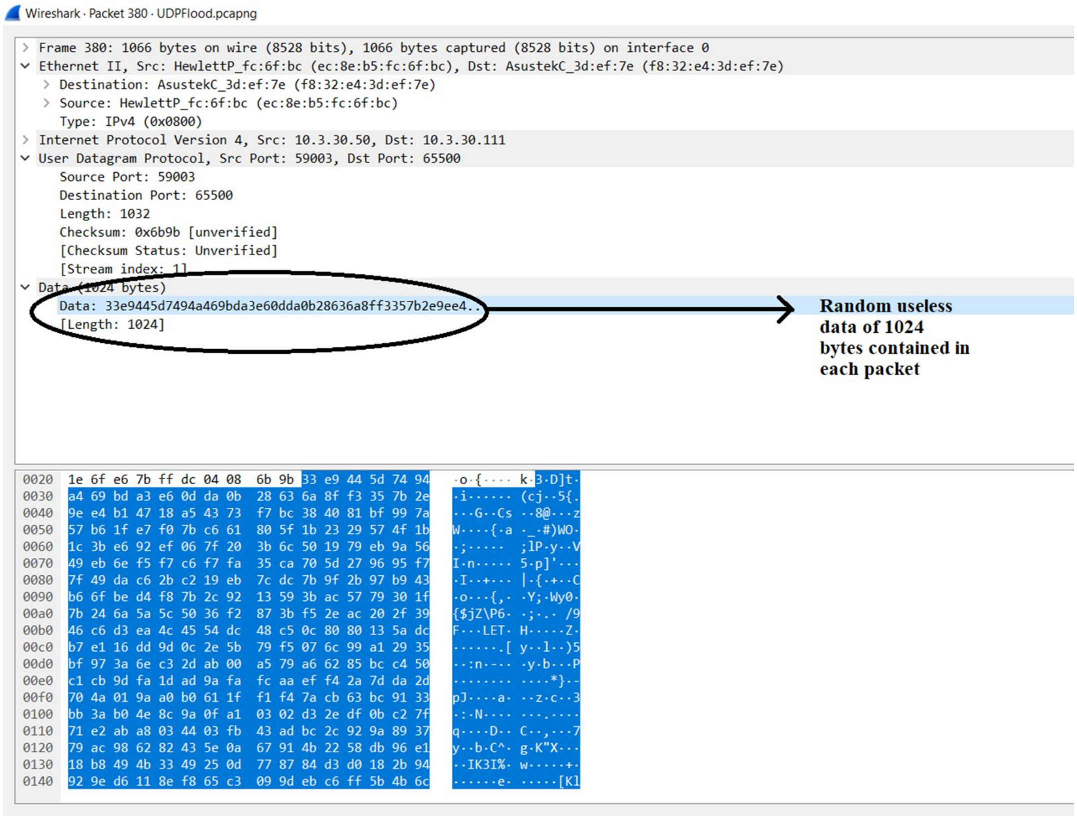
UDP Flooding using Socket Programing:

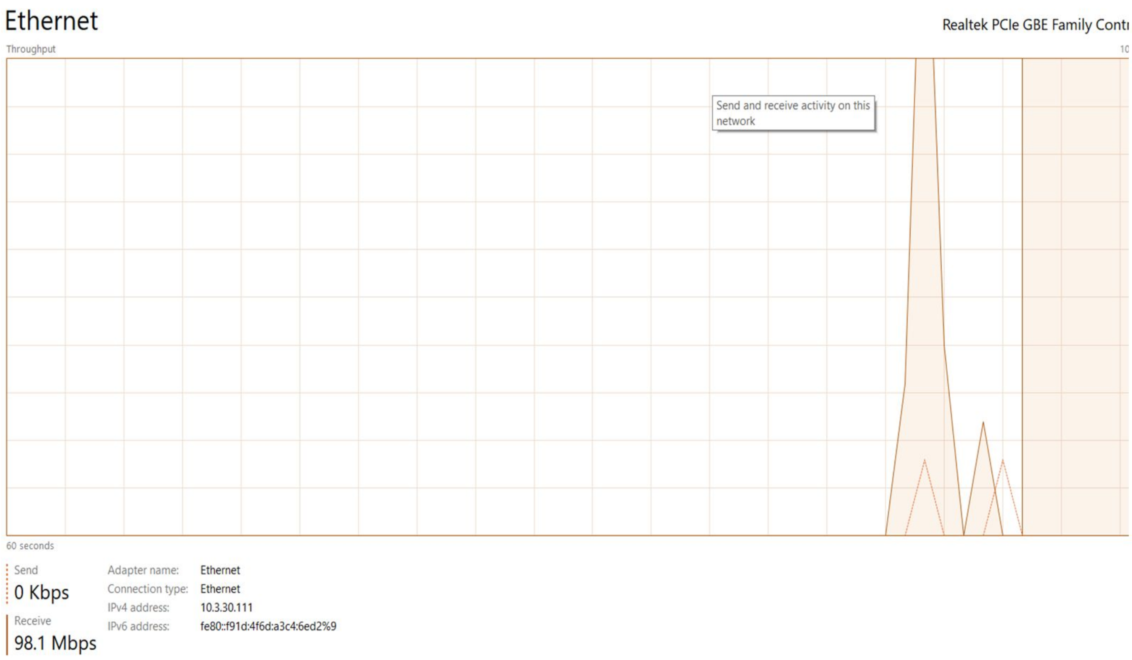
Illustrations:

WIRESHARK PACKET CAPTURE:



1. Source and Destination IP address (source - attacker's ip, Destination - victim's ip)
2. The protocol using which the packet has been sent: UDP
3. Port Numbers: Source port: 59003; Destination port: 65500
4. Packet length:1066 Bytes





If ,

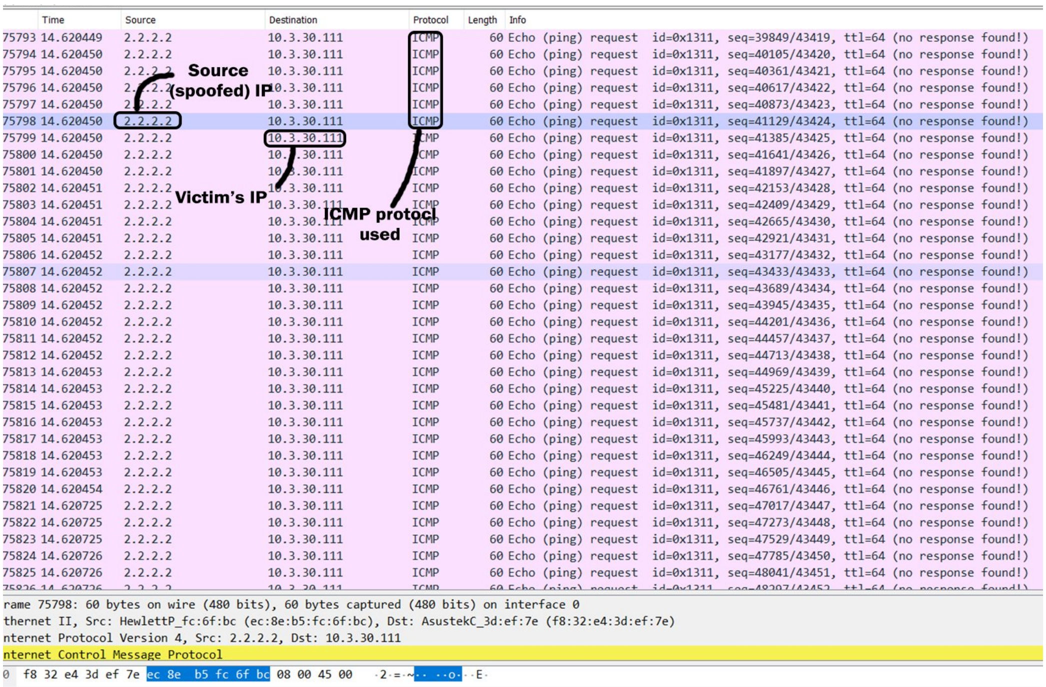
We can see that the victims system is being spammed with random junk packets which do not contain any useful data from the same ip address (attackers ip).

The incoming data rates shoots up from few Kbps to 98Mbps which indicates that the victims bandwidth resources are being attacked

First we create a UDP socket within a try-except block, assign a random packet size.
We specify the end time for the attack using the ‘endtime’ variable.
Until this end time is encountered, the packets are sent using a while loop , with the ‘sendto’ function containing packet size, target IP and port number as parameters.
The program is called from command line using ->
$: sudo python2 dos.py udp [target IP] [port] [duration]

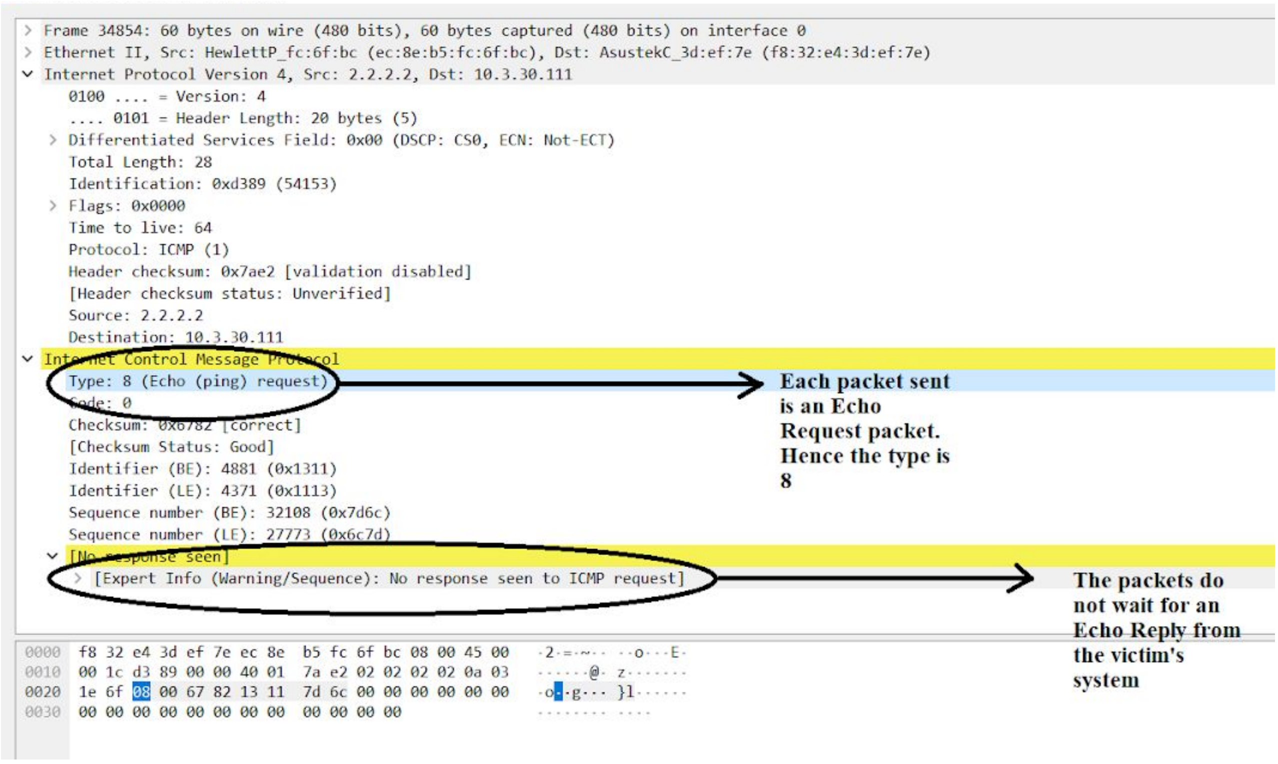
ICMP Flood (Ping Flood attack using hping3):

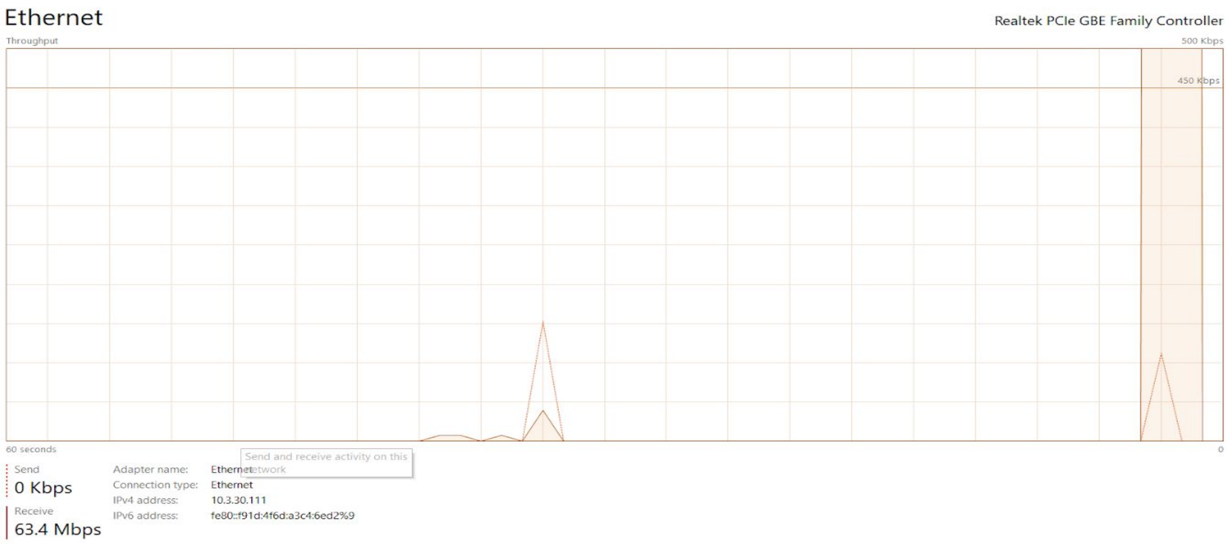
Illustrations:
Wireshark monitor:



In the above illustration we can observe the following:

1. Source IP: 2.2.2.2(attackers spoofed IP address)
2. Destination IP :10.3.30.111 (victims IP address)
3. Protocol using which packets have been sent:ICMP(Internet control Message Protocol)
4. Packet Length:60
5. Each packet is an Echo Request type packet.

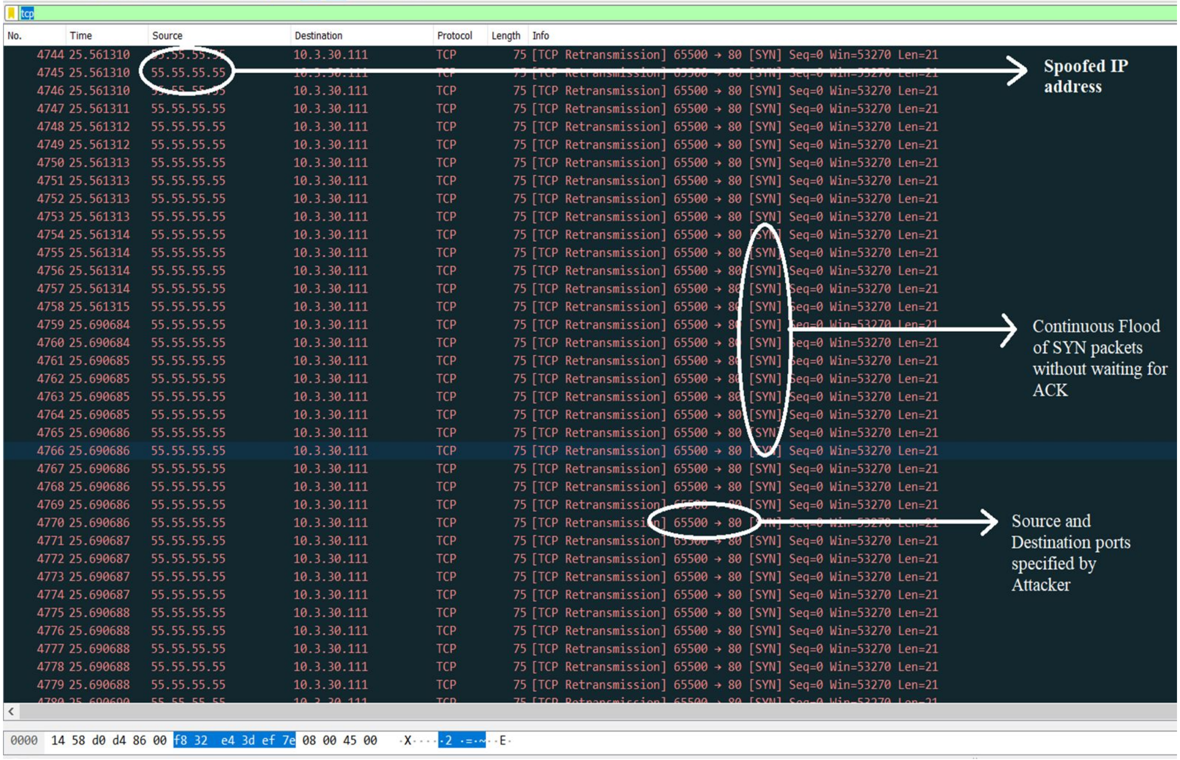




From the above snapshots we can see that the attacker floods the victim's network with request packets(Echo Request).This strains both the incoming and outgoing channels of the network, consuming significant bandwidth and resulting in a denial of service.
The outgoing data rate goes from a few Kbps to 63Mbps.

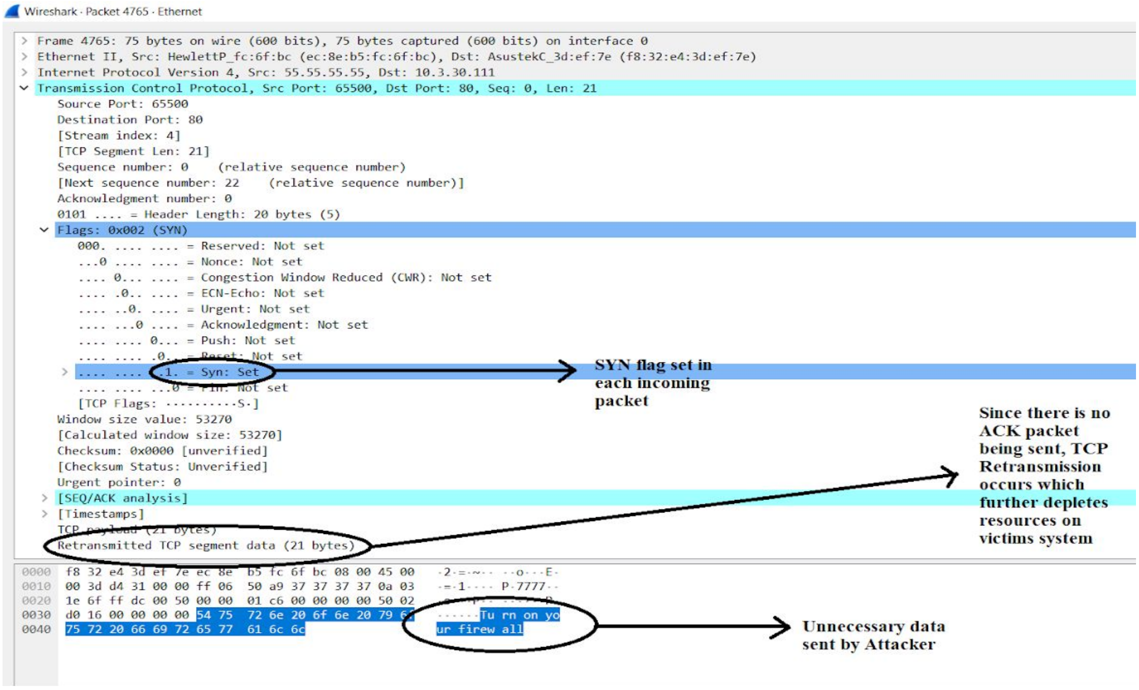
TCP SYN FLOOD

Illustrations:

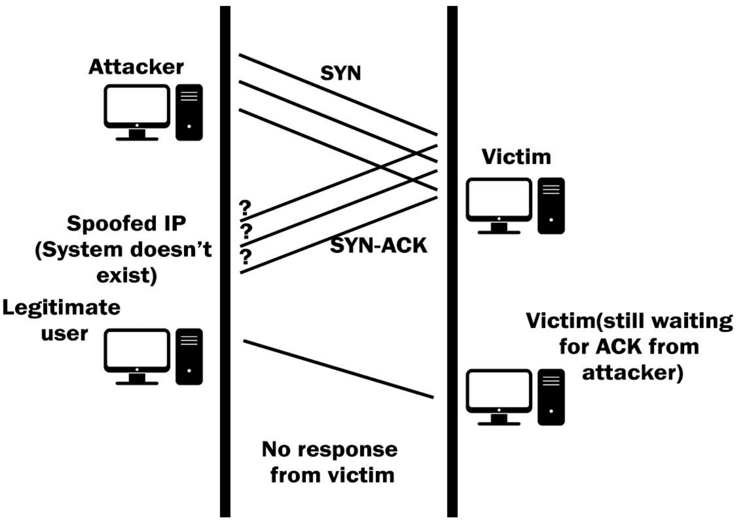


In the above illustrations we can observe the following:

1. Source IP: 55.55.55.55(attackers spoofed IP)
2. Destination IP :10.3.30.111(Victims IP address)
3. Protocol used to send the packets:TCP
4. There is a continuous flood of SYN packets without waiting for Acknowledgement (ACK)
5. Source port:65500
6. Destination port:80
7. Packet Length:75

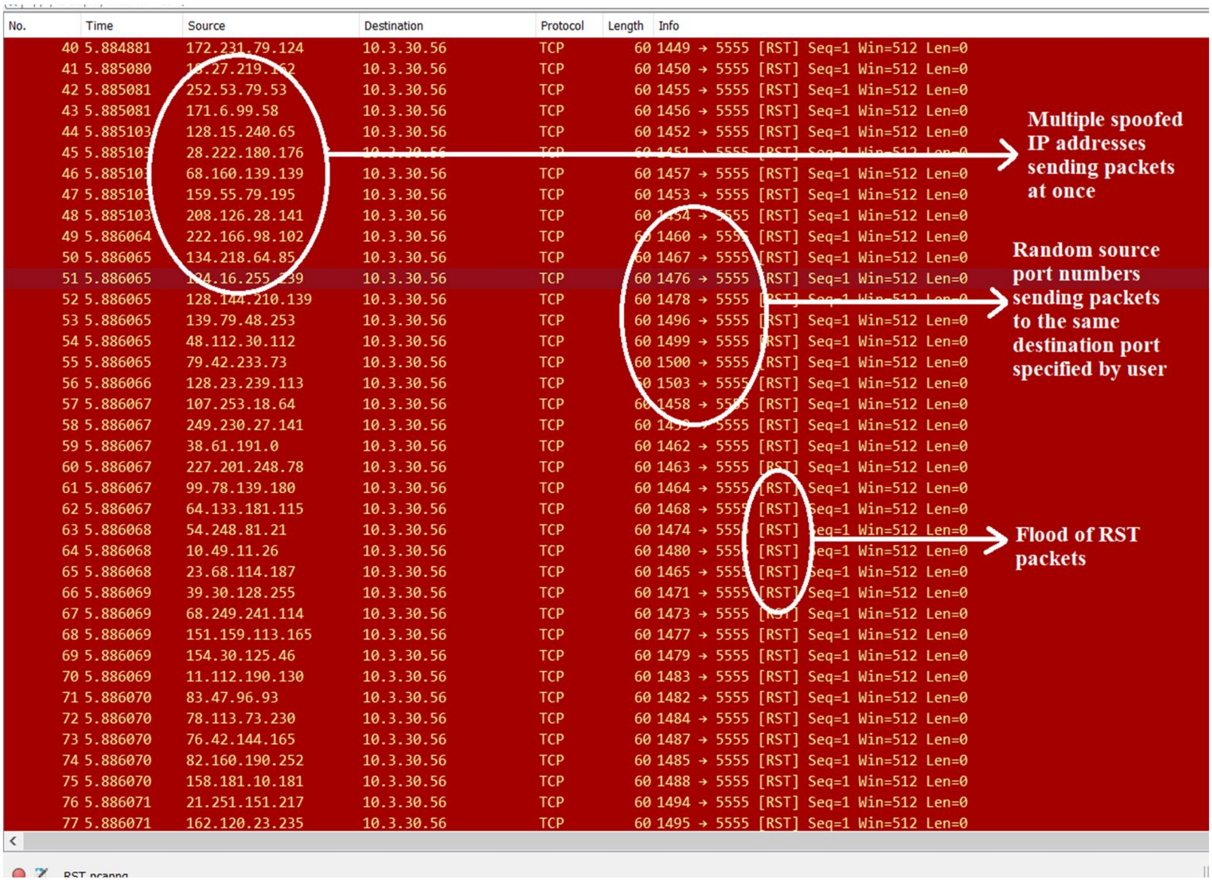


The attacker rapidly sends continuous SYN packets to the victim from a spoofed IP address, The victim attempts to send a SYN ACK packet back to the attacker but since its a spoofed IP address it is not received by the attacker so no ACK packet is generated and the Victim keeps waiting for an ACK reply.
Since there is no reply from the attacker the victims system requests retransmission.This process takes place for every SYN packet sent, and so it depletes the victims resources greatly



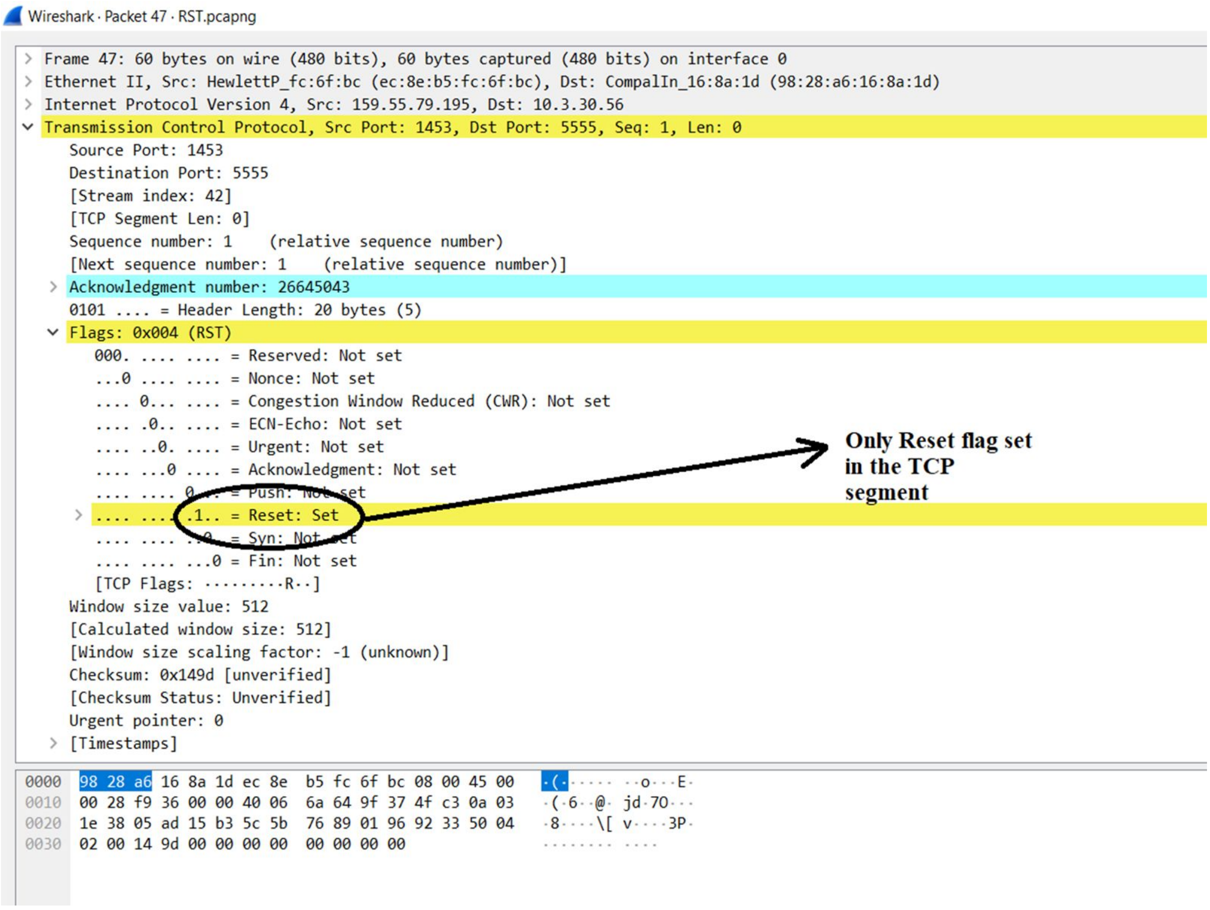
TCP RST FLOOD

Illustrations:



In the above illustrations we can observe the following:

1. Source IP:Randomly generated IP's (Attacker uses a new spoofed IP with every new packet sent)
2. Destination IP: 10.3.30.56(Victims IP address)
3. Protocol used to send the packets :TCP
4. There is a continuous flood of TCP packets with RST flag set
5. Source port:randomly generated port numbers for each packet sent
6. Destination port: 5555
7. Packet length:60



In this attack the attacker sends TCP packets with the RST flag set.
RESET is a flag in TCP packets to indicate that the connection is not longer working. So, if any of the two participants in a TCP connection send a packet contains such a RESET flag, the connection will be closed immediately.
Thus it can be use to attack TCP connections once the attacker can forge TCP packets from any of the two parties if he or she know their IPs, ports and the sequence number of current TCP connection.The attack can be used to make certain users to fail to use certain network services based on TCP if we know the information above.

Contributions by each student:

The main tasks needed to complete the project were researching different types of DoS attacks, understanding socket programming and the different functions of the socket module in python, writing the python script, capturing and analyzing packets in Wireshark and finally writing the project report. The four of us worked together in making this project possible taking equal responsibility in each of the tasks.

Conclusion and Future Work:

This project shows that, with a little understanding of Python and networks, a user can successfully carry out a Denial-Of-Service attack against an unprotected victim. Hence, now more than ever, there is a need for Network Security to prevent these attacks and to stay safe. We further would like to extend this project by working towards providing security by designing a Firewall that is capable of preventing such attacks from a wide range of attacking sources.

References:

Cryptography and Network Security:Principles and Practise by William Stallings

https://www.geeksforgeeks.org/socket-programming-python/

https://docs.python.org/2/howto/sockets.html

https://www.incapsula.com/ddos/ddos-attacks.html

https://www.github.com/akshar-Athreya97/CN-Project

https://www.wikipedia.org

https://www.google.com