Alexandria University
Faculty of Engineering

Master of Computer Science program

Network Security

Network attack using TCP protocol for performing DoS and DDoS attacks

Project Report
Spring 2021

Yousef Mohamed Zook

Contents

Introduction

In this project we are discussing a new DoS attack proposed by this paper. The Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks pose a severe problem in the Internet, whose impact has been well exhibited in the computer network literature.

The main goal of a DDoS is the disruption of services by attempting to reduce access to a machine or service instead of depraving the service itself. In DDoS attack, the attacker exploits any vulnerability in the protocols at different layers. In this way it compromises different systems. These systems are called zombies or bots. DDoS attacks are comprised of packet streams from disparate sources. The DDoS tools do not require technical knowledge to execute them. Hence DDoS are becoming effortless to launch and difficult to detect.

DDoS traffic creates a heavy congestion in the internet and hinders all Internet users whose packets cross congested routers. In the paper, we studied DDoS attack types at various TCP/IP protocols, application-level DDoS attack tools, compare existing GUI tools so that we know the trend of attacking method used by the attackers to launch an attack and various defense mechanisms. DDoS attacks never try to break the victim's system, thus making any old security defense mechanism inefficient. The main goal of a DDoS attack is to cause destruction on a victim either for personal reasons, either for material gain, or for popularity. Application-level attacks overflow a computer with such a high volume of connection requests, that all available operating system resources are disbursed and the computer can no longer process legitimate user requests

Problem

In this attack we are trying to overcome the new defense mechanisms exist nowadays to mitigate the DoS attacks.

The desired outcome is to make the SYN-Flooding DoS attack effective even on networks that contains defense mechanisms like SYN-Cookies and SYN proxy.

TCP Protocol

The TCP protocol is a network protocol that forms the layer 3 of the network stack. It is responsible for transferring the messages from and to different IP addresses.

The TCP protocol is the main channel that is exploited to make DoS attacks like SYN-Flooding attacks. And this is done by making a TCP half-open connection with the victim machine to overflow its requests queue.

Attack new approach

The idea of this new attack is to complete the TCP connection to overcome the defense mechanisms strategies that exists nowadays.

TCP Protocol

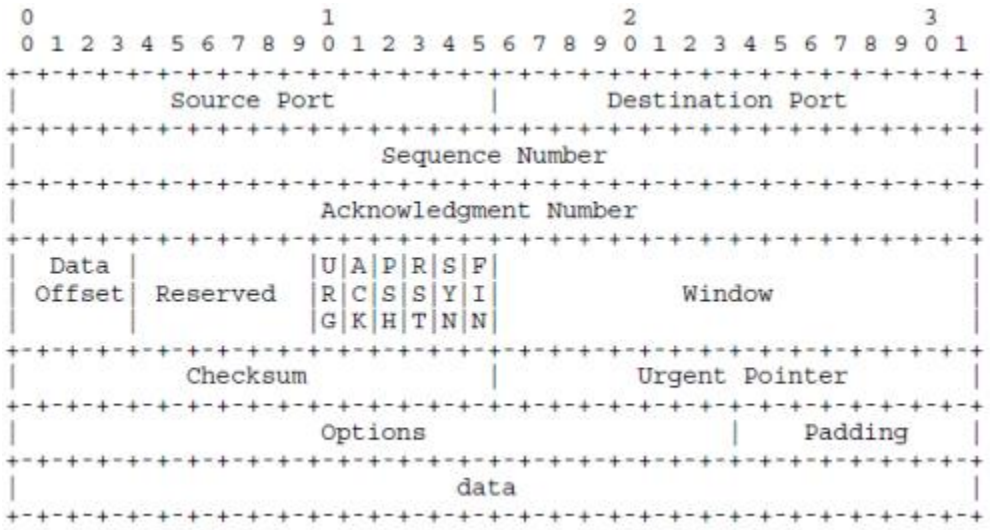
In this part we will discuss the TCP protocol which is as mentioned above the main layer for making SYN-Flooding attacks.

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications (see Fig. 2 – TCP header). The TCP provides for reliable inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks.

The protocol has a well-known structure for messages that contains flags to specify which type of message is it and other header parameters for different usages.

TCP structure

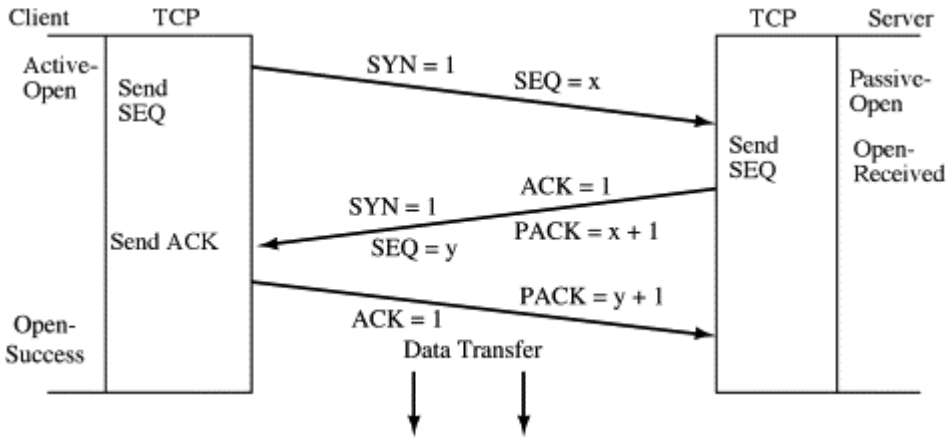
As shown below, the TCP message headers contains several sections. There are sections to specify source and destination IP addresses, and a sequence number field that is used between the sender and receiver to assure synchronization between messages.



Also, there is a flags section that identifies the message type, whether it is a SYN message, ACK message … etc.

Three-Way handshake

In a typical TCP connection, each device maintains its state and the corresponding sequence number to track the incoming sequences of packets. When an end-host device receives a new packet, it will send back an ACK (acknowledgement) packet, containing an acknowledgement number, which means the device has successfully received the data and is awaiting further incoming data as the number indicates. In the establishment of a TCP connection between a client and a server, a TCP three-way handshake process is performed. The process is as illustrated in the following figure:



1. The client sends a SYN (synchronize) packet to the server, which has a random sequence number.

2. The server sends back a SYN-ACK packet, containing a random sequence number and an ACK number acknowledging the client's sequence number.

3. The client sends an ACK number to the server, acknowledging the server's sequence number.

4. The sequence numbers on both ends are synchronized. Both ends can now send and receive data independently.

The three-way handshake happened in the TCP protocol is the main part which is used to make half-open connections for the DoS attacks.

SYN Flood attacks

In this part we will discuss SYN-Flood attacks.

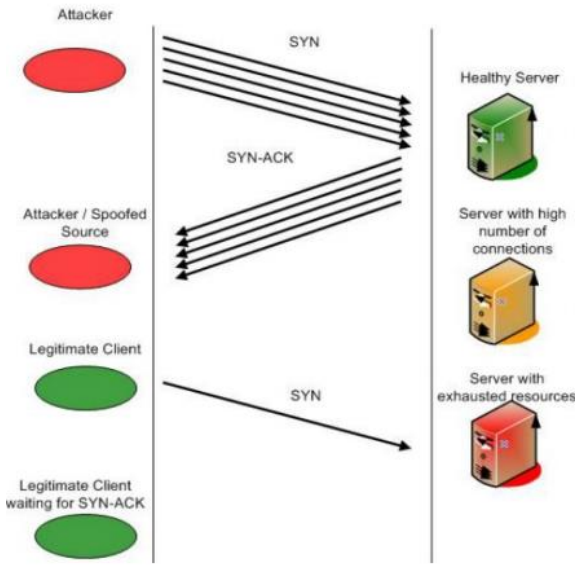
How attack works

The idea of the attack is to make initiate the TCP Three-Way handshake with the victim machine. However, the initiated TCP handshake is never completed from the attacker machine. Which then is repeated for many TCP connections till that make the victim machine unable to respond anymore even for legitimate Users.

This is done by two different ways:

1. The malicious client will keep sending the SYN packets which are usually of 64 bytes. And when the victim server responses by sending the SYN-ACK messages as an acknowledgment to the malicious client, the malicious attacker client just ignores the SYN-ACK message and continue to send the new SYN messages.

2. In other method, the malicious client spoofs the source IP address and sends the SYN packets to the victim server. The Victim server sends the SYN-ACK to the spoofed source address and waits for the ACK response. Since the spoofed sources did not originally send the SYN packets, they never respond back. And the Victim server holds up the connection. In this way many connections are initiated to the server, the server waits for the response back from the client and keeps the connections in its connection table. This exhausts the resources of the server, resulting in the denial of service. As the legitimate users will not be responded or get the service from server due to unavailability of the resources.



This figure shows the process of the SYN-Flood attack.

Defense mechanisms

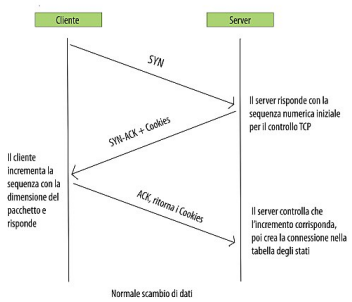
There are different mechanisms to mitigate or prevent the effects of the DoS attack. We are discussing two defense mechanisms of them is this section.

1- SYN-Cookies

According to the main paper mentioned, the key to the SYN flood attack is the filling of the victim's SYN queue. The SYN cookies defense allows the victim to continue accepting connection requests when the SYN queue is full. The SYN cookies approach is to detect when the SYN queue is full and if full, create a cryptographic cookie (a 32-bit number) from information in the SYN segment and then drop the SYN segment. The cookie will be used as the initial sequence number in the SYN-ACK sent to the client. The cookie (plus one) will be returned to the server as the acknowledgement number in the ACK from a legitimate client. The returned cookie can be validated and the most important parts of the SYN segment can be reconstructed from the cookie thus allowing the attacked port to bypass the SYNRECEIVED state (the SYN queue) and to allocate a TCB (Transmission Control Block) for a validated (legitimate) connection directly in the ESTABLISHED state. Since the spoofed clients of the SYN flood never send ACKs, no TCBs are allocated for them in any state when SYN cookies are in use. Thus, the SYN cookies defense allows the port to accept new connections when the SYN queue is full, defeating the SYN flood attack.

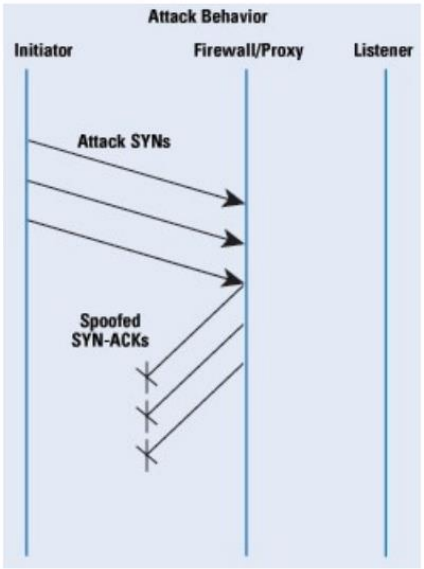
The key to not allocating storage while waiting for the ACK from the client is in the construction of the cookie. Since the cookie will be returned to the server in the client's ACK, part of the cookie encodes

some recoverable information from the SYN segment so that the server will not need to allocate a TCB to hold that information while waiting for the ACK.



2- SYN Proxy

As shown in the following diagram, the basic operation of a firewall/proxy that spoofs SYN-ACKs to the initiator. If the initiator is legitimate, the firewall/proxy sees an ACK and then sets up a connection between itself and the listener, spoofing the initiator's address. The firewall/proxy splits the end-to-end connection into two connections to and from itself. This splitting works as a defense against SYN flooding attacks, because the listener never sees SYNs from an attacker. As long as the firewall/proxy implements some TCP-based defense mechanism such as SYN cookies or a SYN cache, it can protect all the servers on the network behind it from SYN flooding attacks.



New approach for the attack

As mentioned in the introduction part, the new approach for the DoS attack proposed by the paper depends on the idea of completing the TCP connection handshake.

Idea

The idea is to complete the TCP connection handshake through many fake IP addresses to overcome the problem of the syn-cookies and syn-proxy defending mechanisms. let's imagine that attacker is able to complete three-way handshake and establish the connection correctly. If he establishes multiple connections without consequent closing it, the queue will soon get exhausted.

This is done by 3 main modules, the first module is a module to spoof many number of IP addresses in the network that is not reserved to a specific mac address.

The second is to send SYN packets to the victim machine with the source IP address used as the spoofed IPs. And the final module is to send the SYN-ACK message to the victim machine after receiving the sequence number send to the spoofed IP address.

ARP spoofing Module

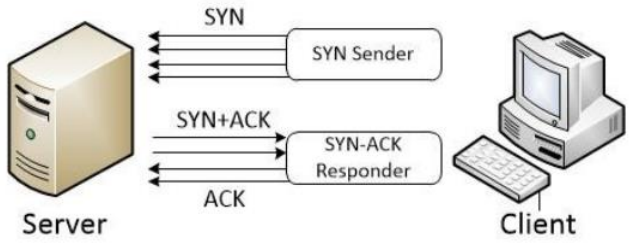
This module just spoofs different IP addresses in the network to be used as a slave machines to send SYN packets to the victim.

Sender module

This module just sends TCP packets opening new connection with SYN flag and defined source and destination addresses.

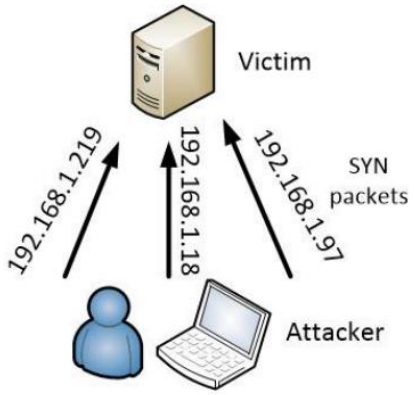
SYN-ACK responder

This module gathers all incoming packets in the network and responds. Packets with SYN+ACK flags with defined source and destination addresses are filtered then, a response packet with ACK flags is sent back to the victim to finish the 3-way handshake. This module cooperates with the first module.



So to run the attack the following steps should be done:

1. To scan the network, discover and collect unused (available) IP addresses.
2. Using e.g. ARP spoofing to redirect all traffic going back to these available addresses, e.g. from the router.
3. After redirecting traffic using SYN Sender to send a few SYN packets to the victim. The sender IP address is set to one of the available address, which isn't assigned to any host.
4. After obtaining SYN+ACK packet it's necessary to send ACK packet to establish a new connection stored in the server's queue.



Implementation

The implementation of this attack is discussed here. We have used Python3 to implement this attack with Scapy library.

We have implemented the modules to be run on an attacker machine with IP address 192.168.1.108 and a victim machine with IP address 192.168.1.104.

The first module done in the ARP-Spoofing module, the module gets the IPs range start and end to spoof and continuously spoof the given victim IP address till stopped by Ctrl+C signal.

The module sends ARP packets to the victim machine telling it that the IP address #x is at Mac address of the attacker machine.

The second and third modules are implemented together in a single python program because of the need of passing the sequence number obtained from the SYN-Sender module to the SYN-ACK responder module.

The SYN-Sender module sends SYN packets to the victim machine with a source ip address equals the fake IP addresses used in the ARP spoofing module, then takes the response sequence number and pass it to the SYN-ACK responder module.

Which sends an ACK packet contains the spoofed sequence number + 1 to complete the TCP connection handshake with the victim machine with each spoofed IP address.

After this step the victim should be flooding with many several TCP connections from different spoofed IP addresses. Which will make the victim not able to respond to legitimate users.

Results

The attack victim is a HTTP server that is running on the victim machine 192.168.1.104 on port number 9000.

We have started by running the ARP spoofing attack with ip ranges from 180 to 220 which represents 40 fake machines.

The following figure shows the attacker arp spoofing module running:

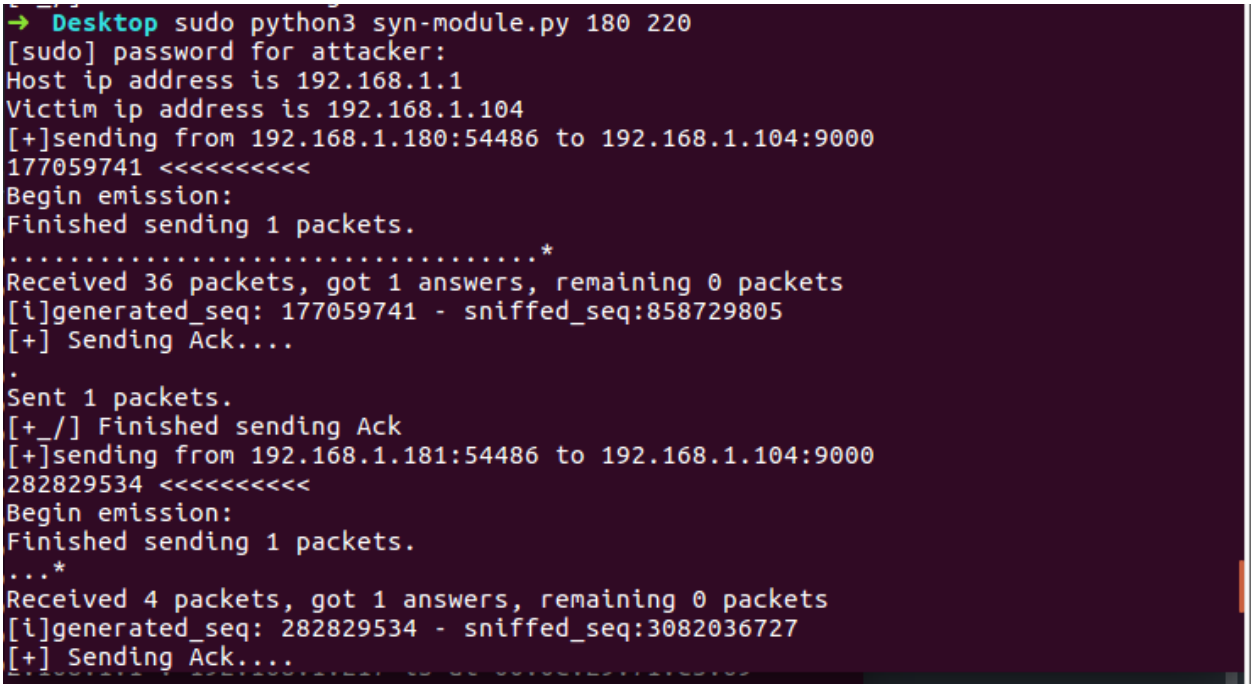
```
sudo python3 arp.py 180 220
[+] 09:50:02 - Sent to 192.168.1.104 : 192.168.1.211 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.1 : 192.168.1.212 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.104 : 192.168.1.212 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.1 : 192.168.1.213 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.104 : 192.168.1.213 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.1 : 192.168.1.214 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.104 : 192.168.1.214 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.1 : 192.168.1.215 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.104 : 192.168.1.215 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.1 : 192.168.1.216 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.104 : 192.168.1.216 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.1 : 192.168.1.217 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.104 : 192.168.1.217 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.1 : 192.168.1.218 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.104 : 192.168.1.218 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.1 : 192.168.1.219 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.104 : 192.168.1.219 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.1 : 192.168.1.220 is-at 00:0c:29:71:c3:69
[+] 09:50:02 - Sent to 192.168.1.104 : 192.168.1.220 is-at 00:0c:29:71:c3:69
[+] 09:50:03 - Sent to 192.168.1.1 : 192.168.1.180 is-at 00:0c:29:71:c3:69
[+] 09:50:03 - Sent to 192.168.1.104 : 192.168.1.180 is-at 00:0c:29:71:c3:69
[+] 09:50:03 - Sent to 192.168.1.1 : 192.168.1.181 is-at 00:0c:29:71:c3:69
[+] 09:50:03 - Sent to 192.168.1.104 : 192.168.1.181 is-at 00:0c:29:71:c3:69
```

The following image shows part of the victim machine ARP table:

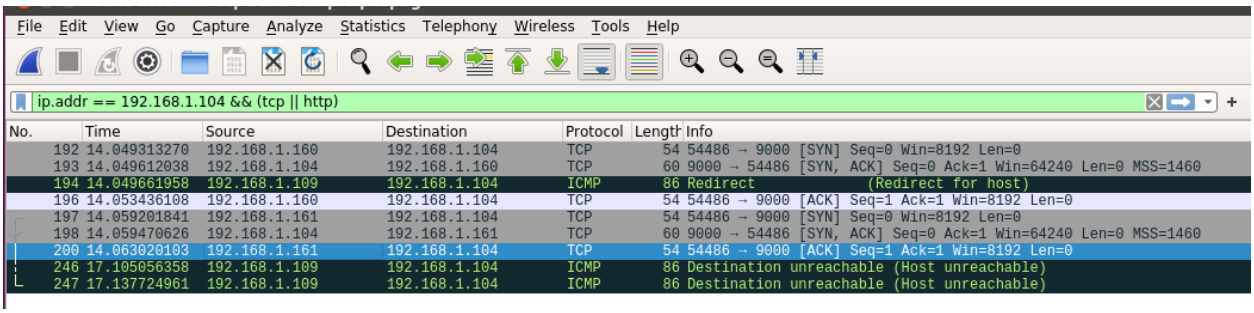
```
victim@ubuntu:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.1.182            ether   00:0c:29:71:c3:69   C                     ens33
192.168.1.161            ether   00:0c:29:71:c3:69   C                     ens33
192.168.1.181            ether   00:0c:29:71:c3:69   C                     ens33
192.168.1.160            ether   00:0c:29:71:c3:69   C                     ens33
192.168.1.180            ether   00:0c:29:71:c3:69   C                     ens33
192.168.1.1              ether   74:da:88:5b:44:ca   C                     ens33
192.168.1.109            ether   00:0c:29:71:c3:69   C                     ens33
192.168.1.106            ether   9c:da:3e:fd:16:05   C                     ens33
```

You can notice that the mac address of the attacker machine “00:0c:29:71:c3:69” is repeated and mapped to several IP addresses which are spoofed by the first module.

After this step we started the second and third module program. The following image shows the module running and logging the sniffed IP addresses:



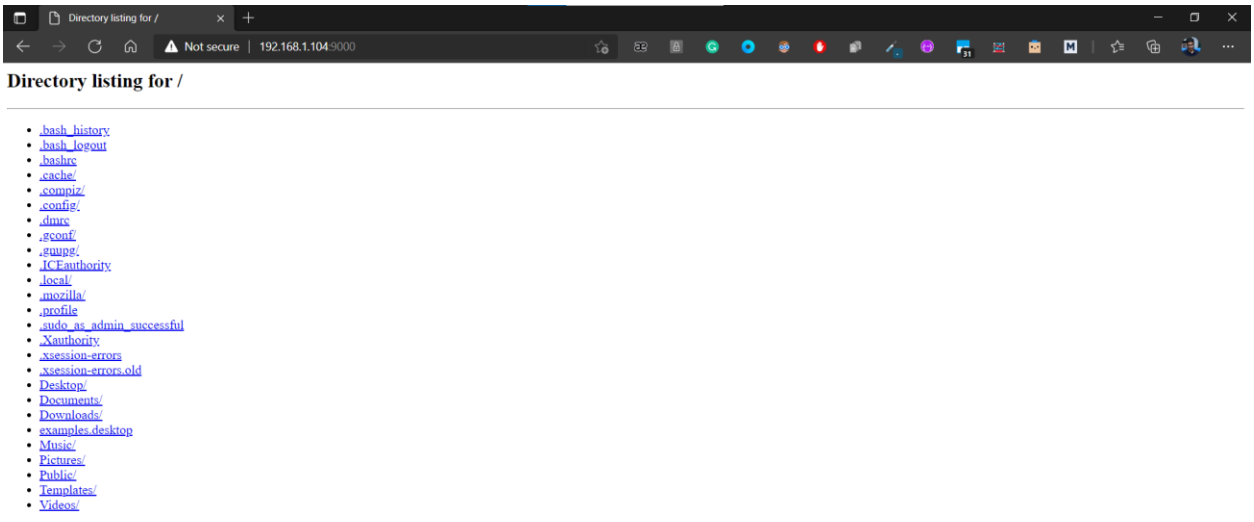
Also you can see the connection completing through wireshark sniffer. As shown in the following image, a connection using 2 fake IP addresses 160,161 were used and established successfully.



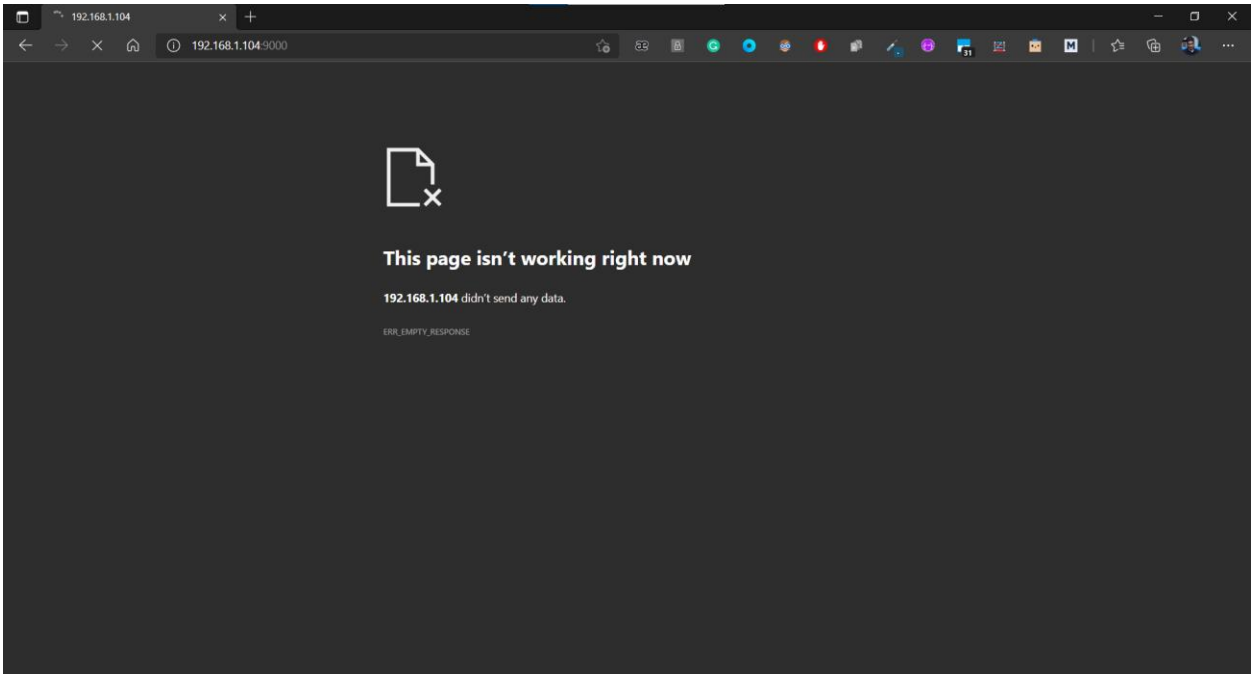
After that we checked to access the server running on the victim machine with browser but it doesn't respond for more than 2 minutes, which indicates that the attack runs successfully.

The following 2 images shows the response before and after running the attack:

Before:



After:



The attack effects

The paper has been cited in Machine Learning Approaches for Combating Distributed Denial of Service Attacks in Modern Networking Environments. The paper has listed the different DoS attack approaches and mentioned this attack as one of the SYN-Flooding attacks. Also, the paper proposes a comparison between different Machine Learning algorithms to mitigate the risk of DoS attacks.

However, there are some challenges in the ML approaches, for ex. It's hard to get a good dataset to train a model for this task.

Suggested defense mechanisms

In these sections, we want to suggest some approaches to mitigate this attack effect.

The first suggestion is to prevent the attack using a MAC address recognition mechanism that is able to check the ARP table periodically or at failure points to check if there a same MAC address that is spoofing different IP addresses or not, and if found then block it.

The second suggestion is to set a timeout for each connection, which means that each client will need to initiate new connection after x amount of time. In this case the attacker will need to keep a continuous running attack to reopen the TCP connection each time it is ended and this will leads to that the attacker will need more resources to perform such continuous open TCP connections more than the power of the server resources.

References:

Network attack using TCP protocol for performing DoS and DDoS attacks | IEEE Conference Publication | IEEE Xplore

Machine Learning Approaches for Combating Distributed Denial of Service Attacks in Modern Networking Environments | IEEE Journals & Magazine | IEEE Xplore

SYN flood DDoS attack | Cloudflare

What is a SYN flood attack and how to prevent it? | NETSCOUT

Three-Way Handshake - an overview | ScienceDirect Topics

SYN cookies - Wikipedia

Python Code