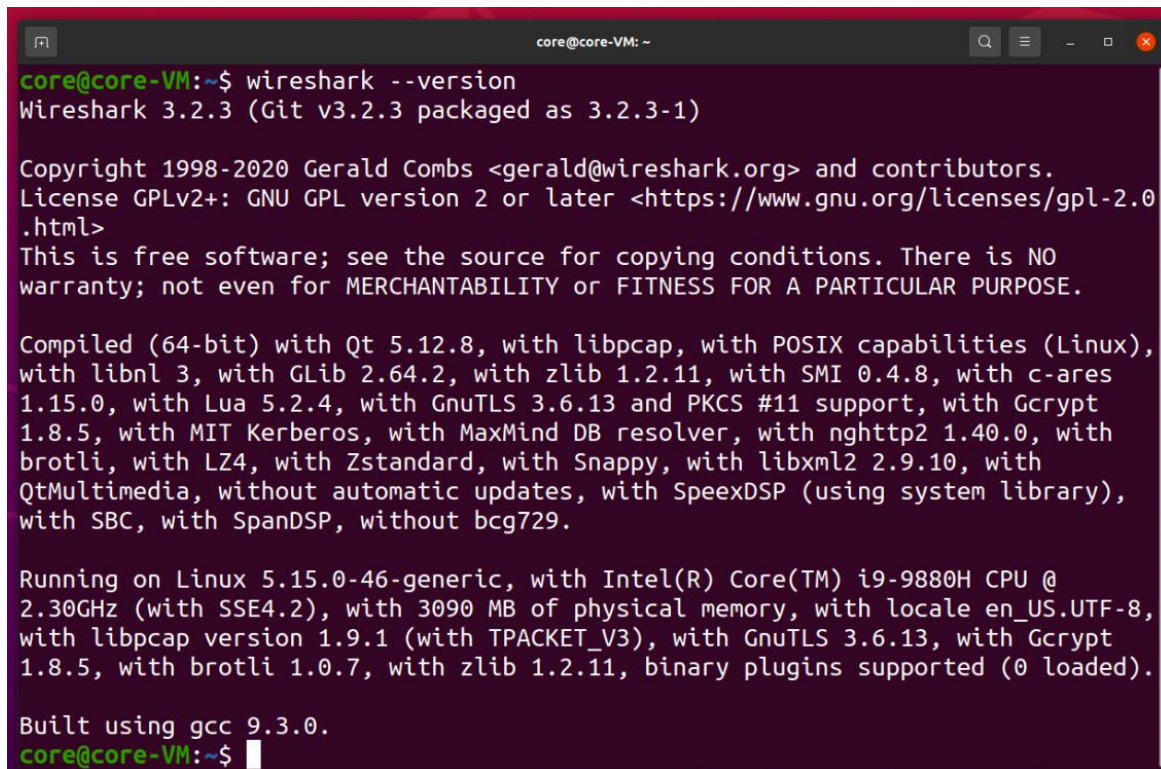


Investigate and Characterize the Impact of DoS Attacks from Attackers on the Webserver using Wireshark, the Protocol Analyzer

Install Wireshark

To install Wireshark, I used the following commands, `sudo apt-get update`, `sudo apt-get install wireshark` and `wireshark --version`.

A terminal window titled 'core@core-VM: ~' with a dark background and light green text. The window shows the output of the command 'wireshark --version'. The output includes the version number 'Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1)', copyright information for Gerald Combs, the GPL license, a disclaimer, a detailed list of compiled dependencies (Qt, libpcap, POSIX, libnl, GLib, zlib, SMI, c-ares, Lua, GnuTLS, PKCS #11, Gcrypt, MIT Kerberos, MaxMind DB, nghttp2, brotli, LZ4, Zstandard, Snappy, libxml2, QtMultimedia, SpeexDSP, SBC, SpanDSP), system information (Linux 5.15.0-46-generic, Intel(R) Core(TM) i9-9880H CPU @ 2.30GHz, 3090 MB memory, locale en_US.UTF-8), and the compiler used (gcc 9.3.0).

```
core@core-VM:~$ wireshark --version
Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1)

Copyright 1998-2020 Gerald Combs <gerald@wireshark.org> and contributors.
License GPLv2+: GNU GPL version 2 or later <https://www.gnu.org/licenses/gpl-2.0.html>
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Compiled (64-bit) with Qt 5.12.8, with libpcap, with POSIX capabilities (Linux),
with libnl 3, with GLib 2.64.2, with zlib 1.2.11, with SMI 0.4.8, with c-ares
1.15.0, with Lua 5.2.4, with GnuTLS 3.6.13 and PKCS #11 support, with Gcrypt
1.8.5, with MIT Kerberos, with MaxMind DB resolver, with nghttp2 1.40.0, with
brotli, with LZ4, with Zstandard, with Snappy, with libxml2 2.9.10, with
QtMultimedia, without automatic updates, with SpeexDSP (using system library),
with SBC, with SpanDSP, without bcg729.

Running on Linux 5.15.0-46-generic, with Intel(R) Core(TM) i9-9880H CPU @
2.30GHz (with SSE4.2), with 3090 MB of physical memory, with locale en_US.UTF-8,
with libpcap version 1.9.1 (with TPACKET_V3), with GnuTLS 3.6.13, with Gcrypt
1.8.5, with brotli 1.0.7, with zlib 1.2.11, binary plugins supported (0 loaded).

Built using gcc 9.3.0.
core@core-VM:~$
```

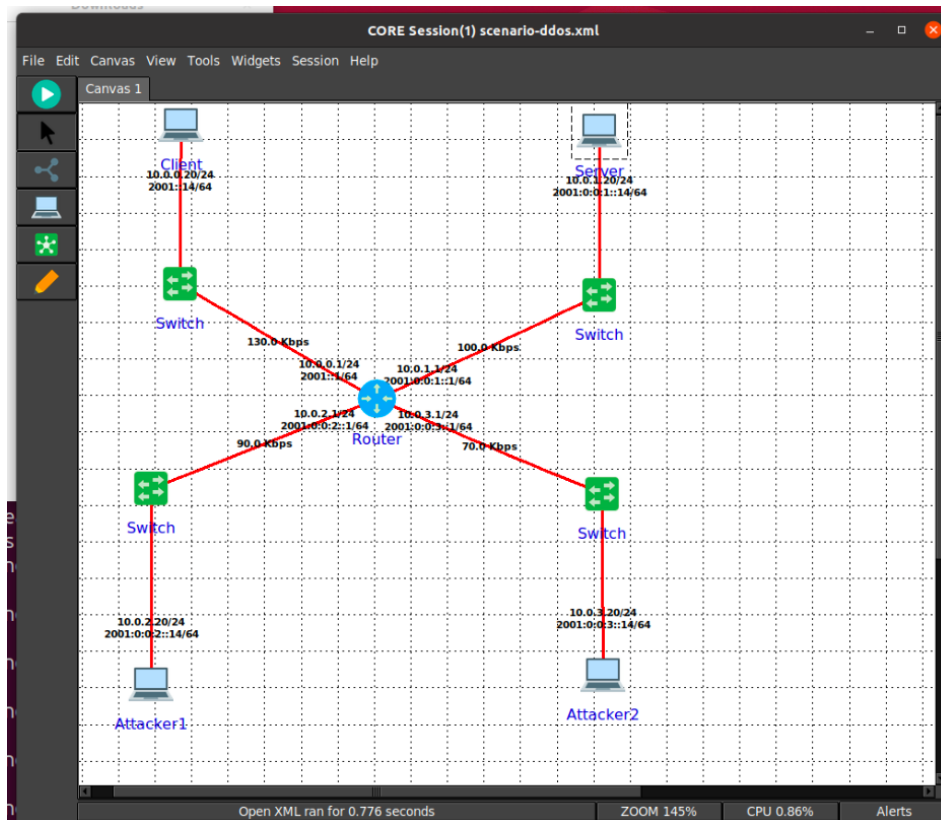
Setting Up the CORE Scenario

I first ensured the CORE daemon was running to set up the core scenario. To check that, I used the command, `systemctl status core-daemon`

```
core@core-VM: ~/hw1/part_1
core@core-VM: ~/hw1/part_1$ systemctl status core-daemon
● core-daemon.service - Common Open Research Emulator Service
   Loaded: loaded (/lib/systemd/system/core-daemon.service; enabled; vendor p
   Active: active (running) since Wed 2024-09-04 09:26:48 EDT; 1h 31min ago
   Main PID: 656 (core-daemon)
     Tasks: 11
    Memory: 73.1M
    CGroup: /system.slice/core-daemon.service
            └─656 /home/core/.cache/pypoetry/virtualenvs/core-mpDUMUHI-py3.8/b>

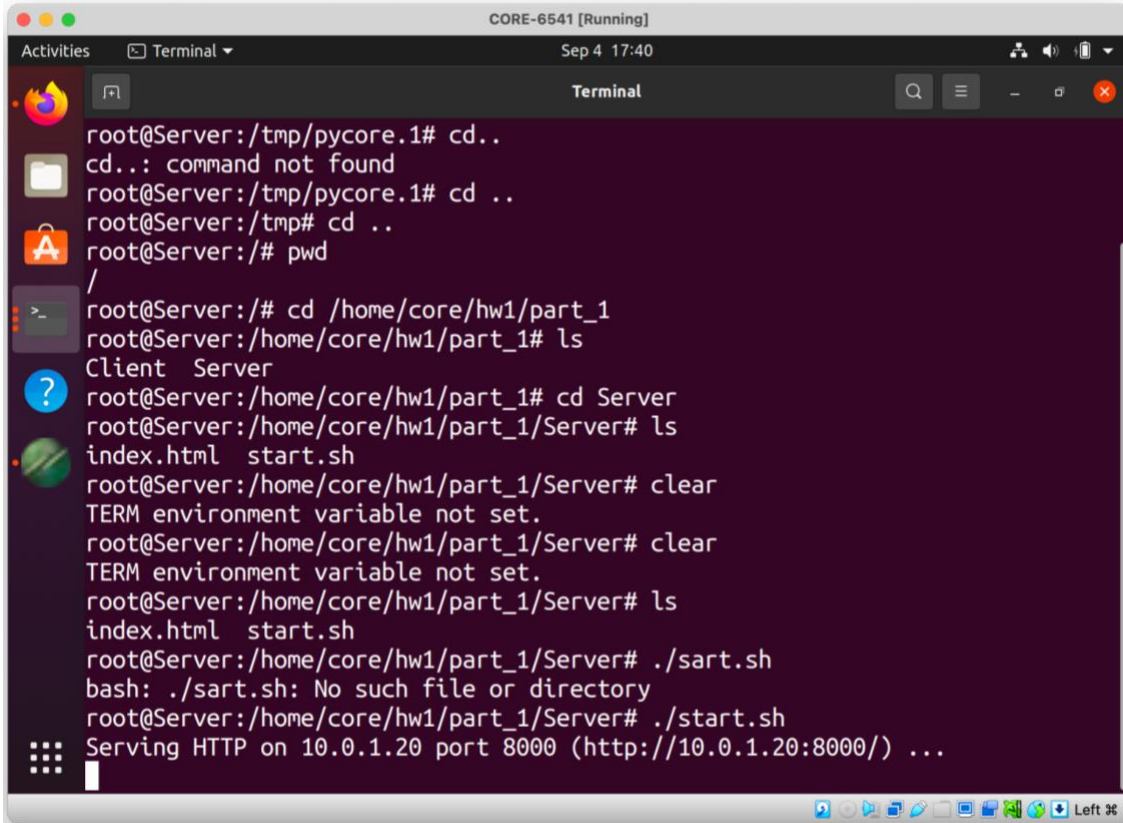
Sep 04 09:55:25 core-VM core-daemon[656]: 2024-09-04 09:55:25,476 - INFO - core>
Sep 04 09:55:25 core-VM core-daemon[656]: 2024-09-04 09:55:25,476 - INFO - sess>
Sep 04 09:55:25 core-VM core-daemon[656]: 2024-09-04 09:55:25,477 - INFO - core>
Sep 04 09:55:25 core-VM core-daemon[656]: 2024-09-04 09:55:25,477 - INFO - sess>
Sep 04 09:55:25 core-VM core-daemon[656]: 2024-09-04 09:55:25,477 - INFO - core>
Sep 04 09:55:25 core-VM core-daemon[656]: 2024-09-04 09:55:25,477 - INFO - sess>
Sep 04 09:55:25 core-VM core-daemon[656]: 2024-09-04 09:55:25,477 - INFO - core>
Sep 04 09:55:25 core-VM core-daemon[656]: 2024-09-04 09:55:25,478 - INFO - sess>
Sep 04 09:55:25 core-VM core-daemon[656]: 2024-09-04 09:55:25,479 - INFO - core>
Sep 04 09:55:25 core-VM core-daemon[656]: 2024-09-04 09:55:25,479 - INFO - sess>
lines 1-19/19 (END)
```

Then, I loaded the file “scenario-ddos.xml” in the CORE GUI.



Running the Scenarios

First, I ran the scenario by clicking the green play button. Then, I ran `start.sh` in the server node using the command, `./start.sh`. Next, I ran the `run_curl.sh` in the client node using the command, `./run_curl.sh`. Followed I ran the `man hping3` command to read the manual of the `hping3`.



```
root@Server:/tmp/pycore.1# cd..
cd..: command not found
root@Server:/tmp/pycore.1# cd ..
root@Server:/tmp# cd ..
root@Server:/# pwd
/
root@Server:/# cd /home/core/hw1/part_1
root@Server:/home/core/hw1/part_1# ls
Client  Server
root@Server:/home/core/hw1/part_1# cd Server
root@Server:/home/core/hw1/part_1/Server# ls
index.html  start.sh
root@Server:/home/core/hw1/part_1/Server# clear
TERM environment variable not set.
root@Server:/home/core/hw1/part_1/Server# clear
TERM environment variable not set.
root@Server:/home/core/hw1/part_1/Server# ls
index.html  start.sh
root@Server:/home/core/hw1/part_1/Server# ./sart.sh
bash: ./sart.sh: No such file or directory
root@Server:/home/core/hw1/part_1/Server# ./start.sh
Serving HTTP on 10.0.1.20 port 8000 (http://10.0.1.20:8000/) ...
```

```
CORE-6541 [Running]
Sep 4 17:41
Terminal
<b> Legit Server </b>
</html>
--Wed 04 Sep 2024 05:41:41 PM EDT--
<html>
  <b> Legit Server </b>
</html>
--Wed 04 Sep 2024 05:41:44 PM EDT--
<html>
  <b> Legit Server </b>
</html>
--Wed 04 Sep 2024 05:41:46 PM EDT--
<html>
  <b> Legit Server </b>
</html>
--Wed 04 Sep 2024 05:41:48 PM EDT--
<html>
  <b> Legit Server </b>
</html>
```

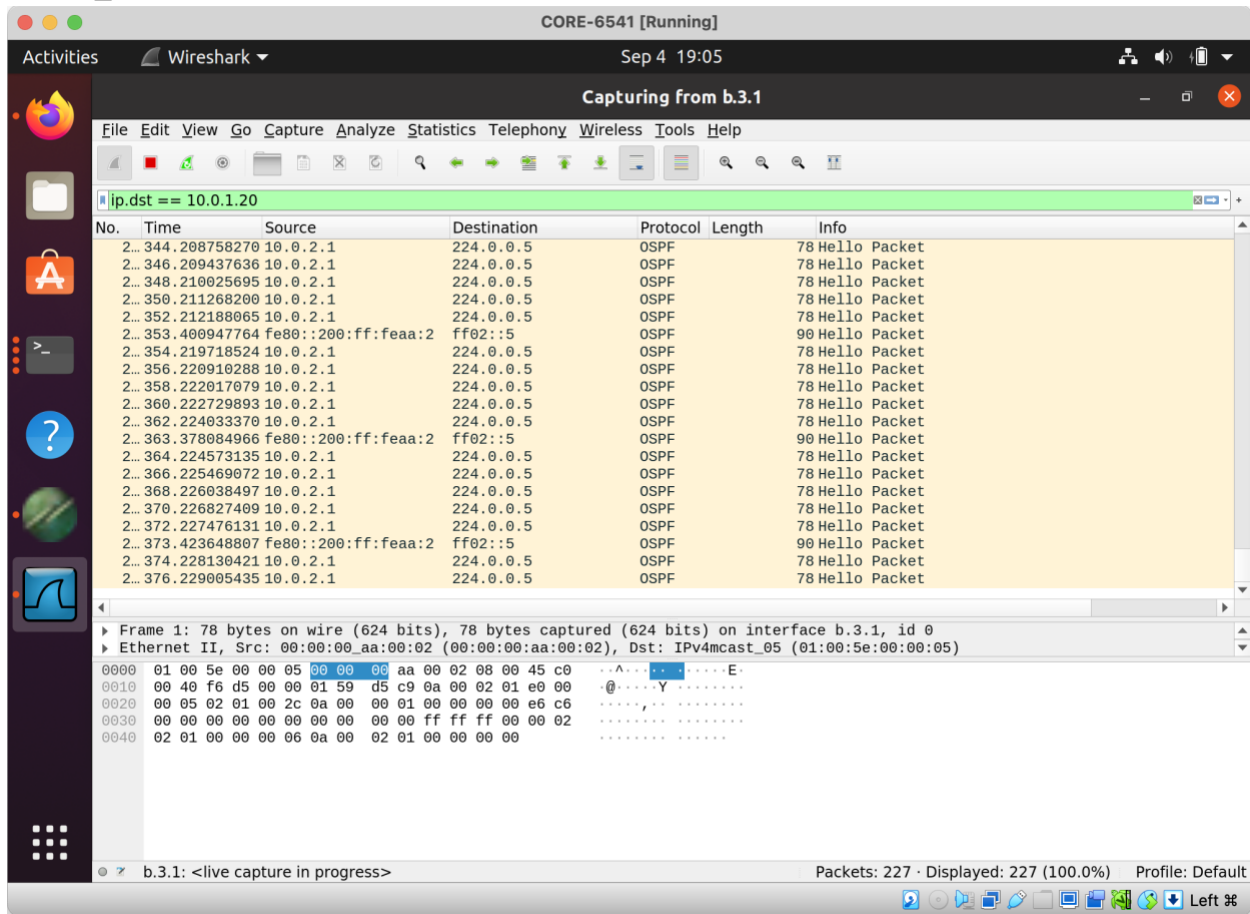
```
Terminal
HPING3(8)      System Manager's Manual      HPING3(8)

NAME
    hping3 - send (almost) arbitrary TCP/IP packets to network hosts

SYNOPSIS
    hping3 [ -hvnqVDzZ012WrfxykQbFSRPAUXYjJBuTG ] [
    -c count ] [ -i wait ] [ --fast ] [ -I interface ] [
    -9 signature ] [ -a host ] [ -t tll ] [ -N ip id ] [
    -H ip protocol ] [ -g fragoff ] [ -m mtu ] [ -o tos ] [
    -C icmp type ] [ -K icmp code ] [ -s source port ] [
    -p[+][+] dest port ] [ -w tcp window ] [ -O tcp offset ] [
    -M tcp sequence number ] [ -L tcp ack ] [ -d data size ] [
    -E file-al page hping3(8) line 1 (press h for help or q to quit)
```

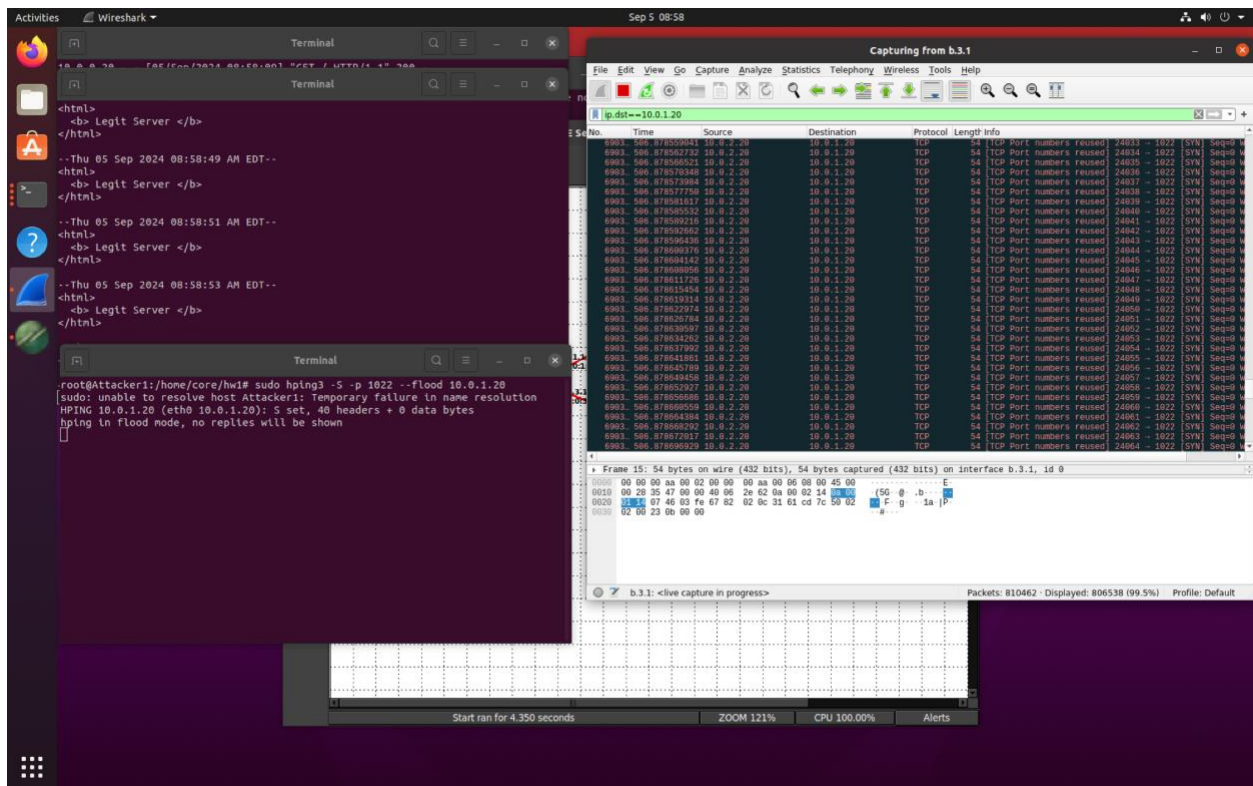

1. Run the Wireshark:

First, run the Wireshark with the filter `ip.dst==10.0.1.20` and then run the `./run_curl.sh`, command to see the traffic through the client and server using Wireshark.



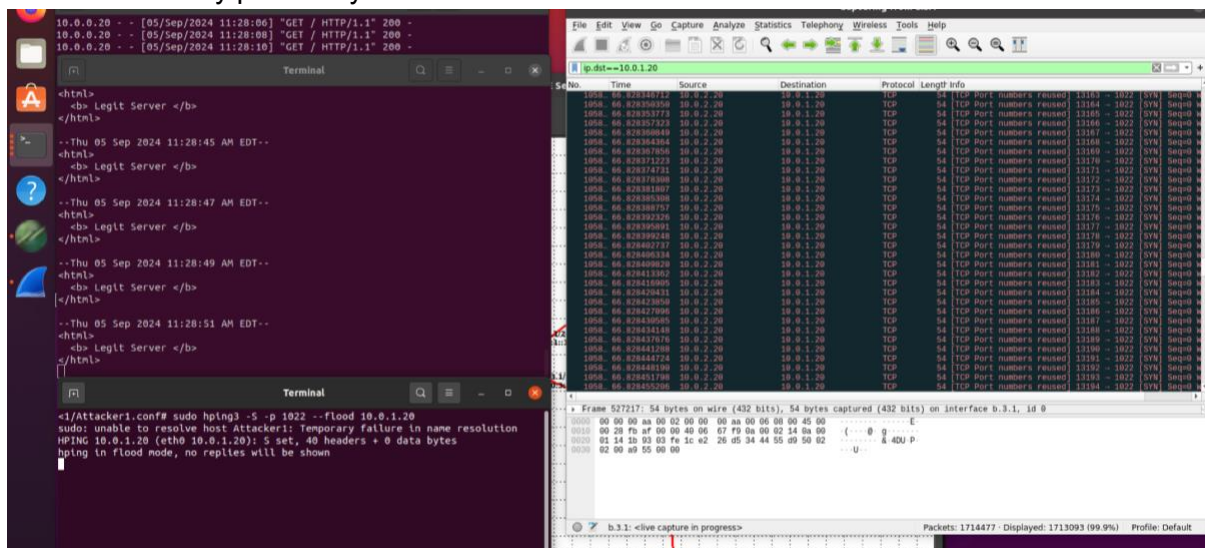
a) SYN Flood DoS Attack

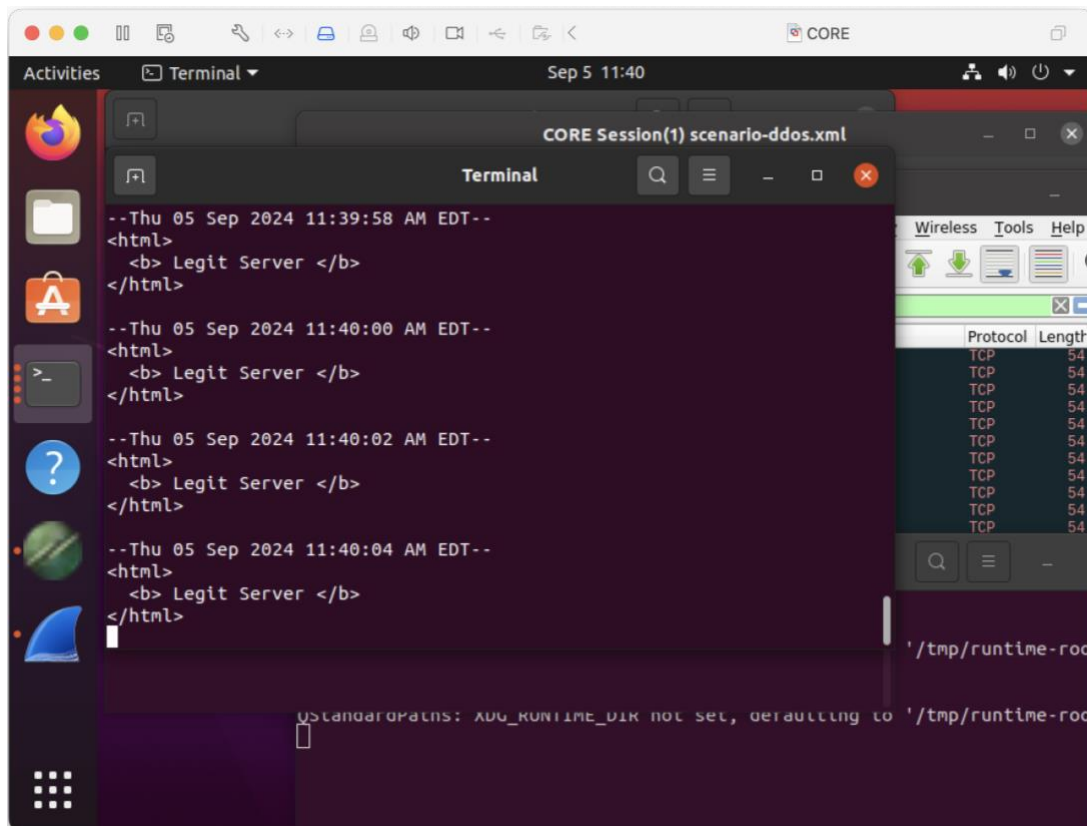
Did an SYN flood DoS attack on the server from attacker 1 using the command, `sudo hping3 -S -p 1022 --flood 10.0.1.20`. The `-S` flag is used to provide SYN packets, and `-p` sets the port i.e. 1022.



b) Is the Attack Effective or Not?

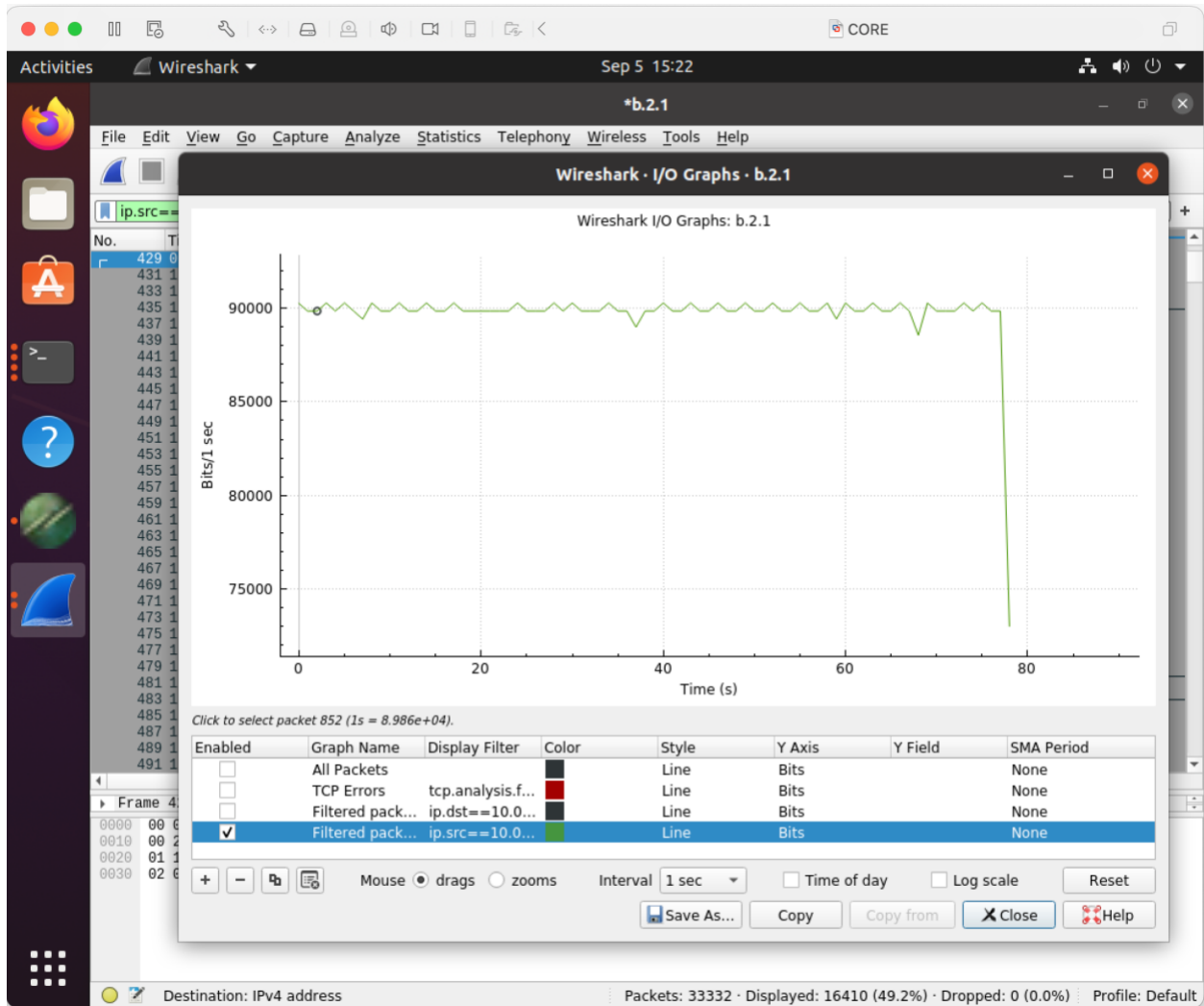
The attack is not effective because the threshold of the server that it can accommodate is higher than the packets being sent. As a result, the Client kept receiving the response for its curl command. The document said to stop the attack after 1 minute but I stopped after 10 mins to see if there's any possibility for the attack to be effective but it's not effective.





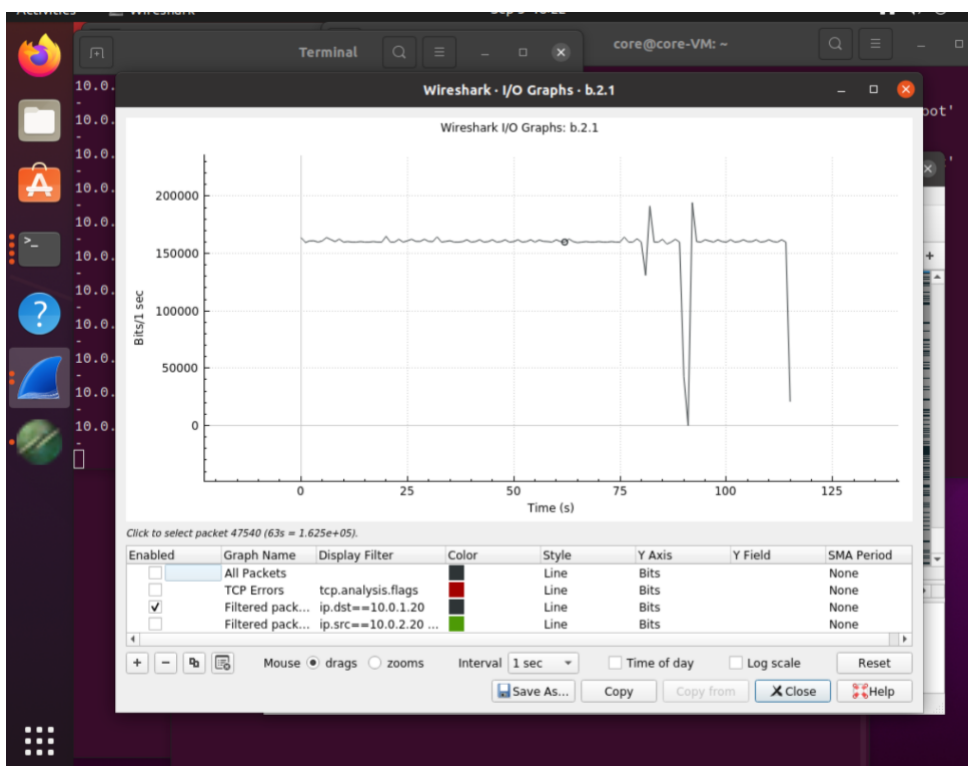
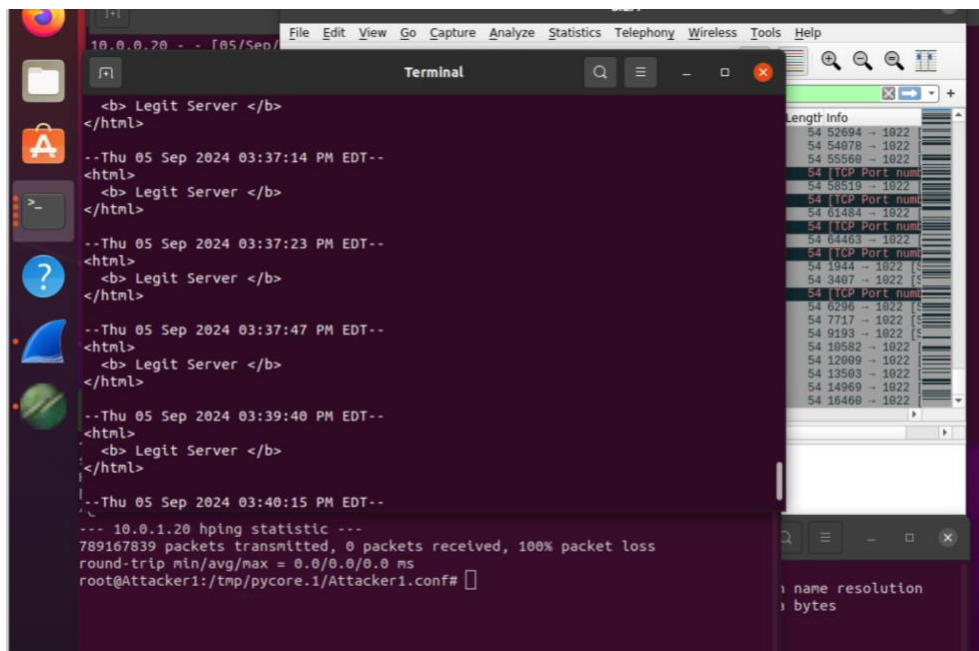
c) Magnitude of the Attack

To find the magnitude of the attack, we've to find the peak value on the Y-axis. This represents the highest bandwidth consumption of the attack. From, the graph the values range between 75000 bps to 90000 bps. The peak value on the graph seems to be close to 90000 bps. This means the magnitude of the attack is approximately 90 kbps. Therefore, the attack is consuming approximately 90000 bps/90 kbps at its peak as shown in the graph.



d) SYN Flood Attack from Attacker 1 and Attacker 2

I did a SYN Flood attack on the server node but this time from both Attacker 1 and Attacker 2 nodes. As a result, the attack was effective and the client node stopped receiving responses from the server for almost 2 minutes because we made a SYN Flood attack (DoS) on the server. The graph shows a range between 0 - 200000 bps and a stable reliable line of traffic of around 150000 bps with some fluctuations. This concludes the magnitude of the attack is approximately 200000 bps/200 kbps.



e) ICMP Flooding Attack

This time I did an ICMP Flood attack on the server from Attacker 1 and the attack was not effective because the client kept receiving the response from the server. The graph ranges

between 0 and 100000 bps and shows a stable line of traffic pattern at approximately 80000 bps i.e. 80 kbps and also the peak traffic rate during the attack is approximately the same. Therefore, the magnitude of the attack is 80000 bps/80 kbps.

The screenshot displays a network traffic analysis tool interface. The top pane shows a list of network packets, primarily ICMP Echo (ping) requests, with columns for No., Time, Source, Destination, Protocol, Length, and Info. The bottom pane shows a terminal window with the following commands and output:

```
<1/Attacker1.conf# sudo hping3 -S -p 1022 --flood 10.0.1.20
sudo: unable to resolve host Attacker1: Temporary failure in name resolution
HPING 10.0.1.20 (eth0 10.0.1.20): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.1.20 hping statistic ---
789167839 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
<1/Attacker1.conf# sudo hping3 --icmp--flood 10.0.1.20
sudo: unable to resolve host Attacker1: Temporary failure in name resolution
hping3: unrecognized option '--icmp--flood'
Try hping3 --help
<1/Attacker1.conf# sudo hping3 --icmp --flood 10.0.1.20
sudo: unable to resolve host Attacker1: Temporary failure in name resolution
HPING 10.0.1.20 (eth0 10.0.1.20): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

The screenshot shows a terminal window with a continuous stream of HTML tags, likely generated by a script during the attack:

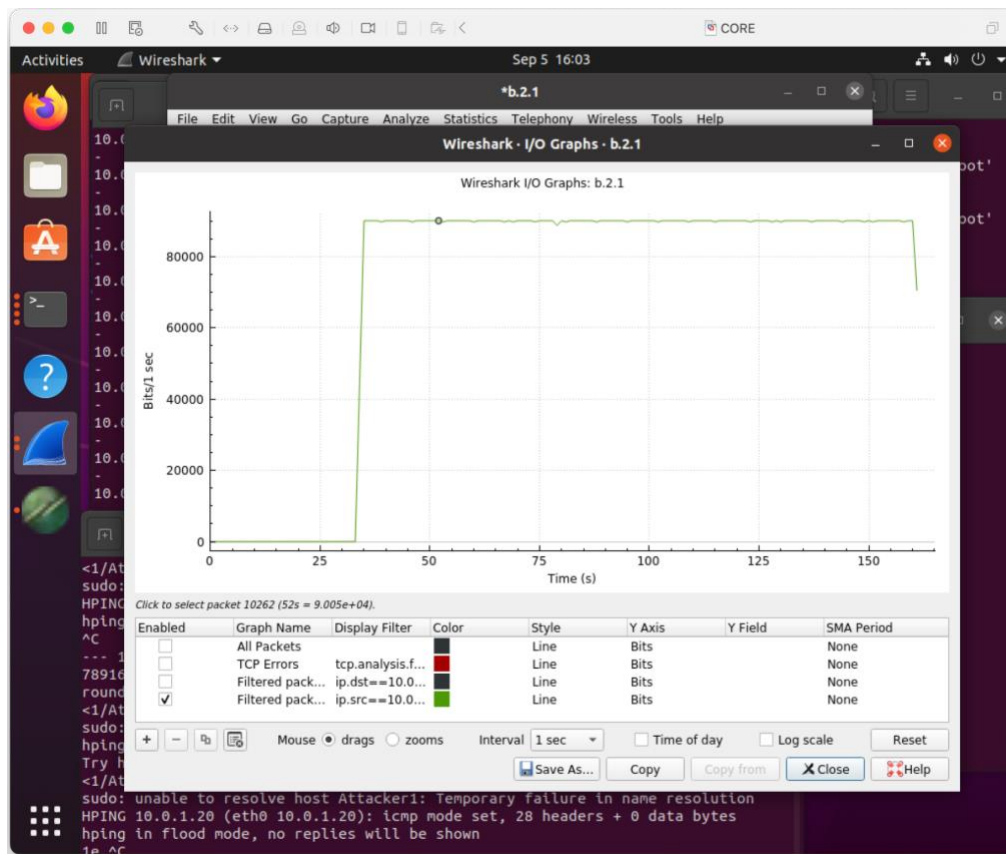
```
--Thu 05 Sep 2024 04:02:18 PM EDT--
<html>
  <b> Legit Server </b>
</html>

--Thu 05 Sep 2024 04:02:20 PM EDT--
<html>
  <b> Legit Server </b>
</html>

--Thu 05 Sep 2024 04:02:22 PM EDT--
<html>
  <b> Legit Server </b>
</html>

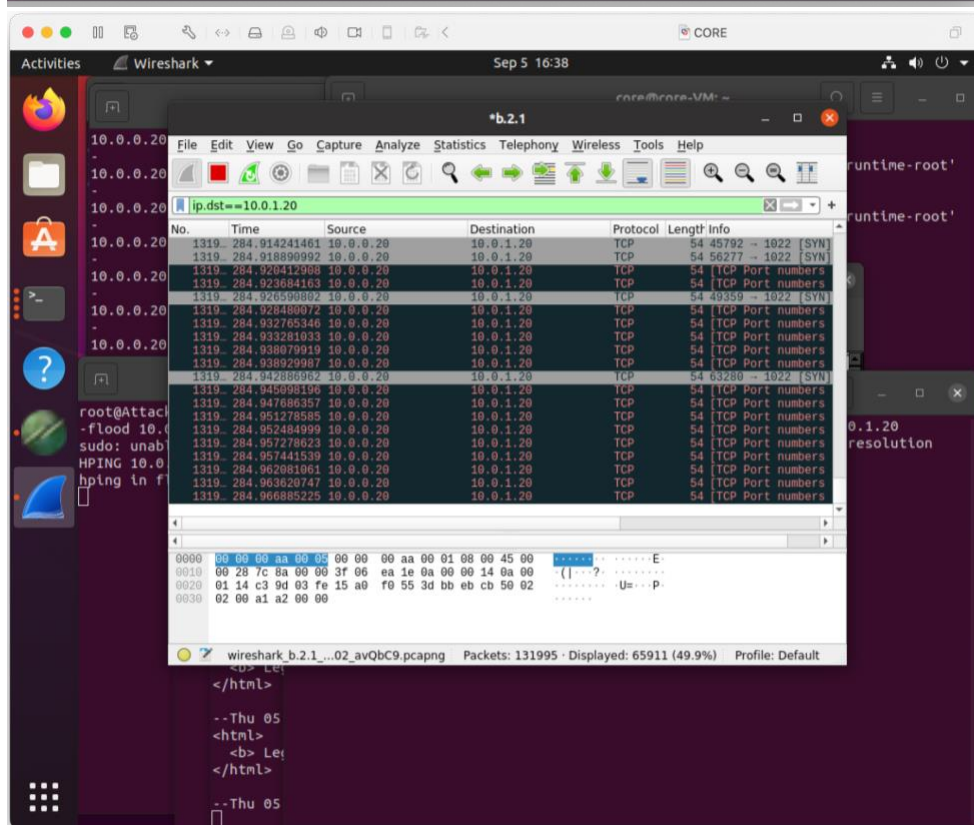
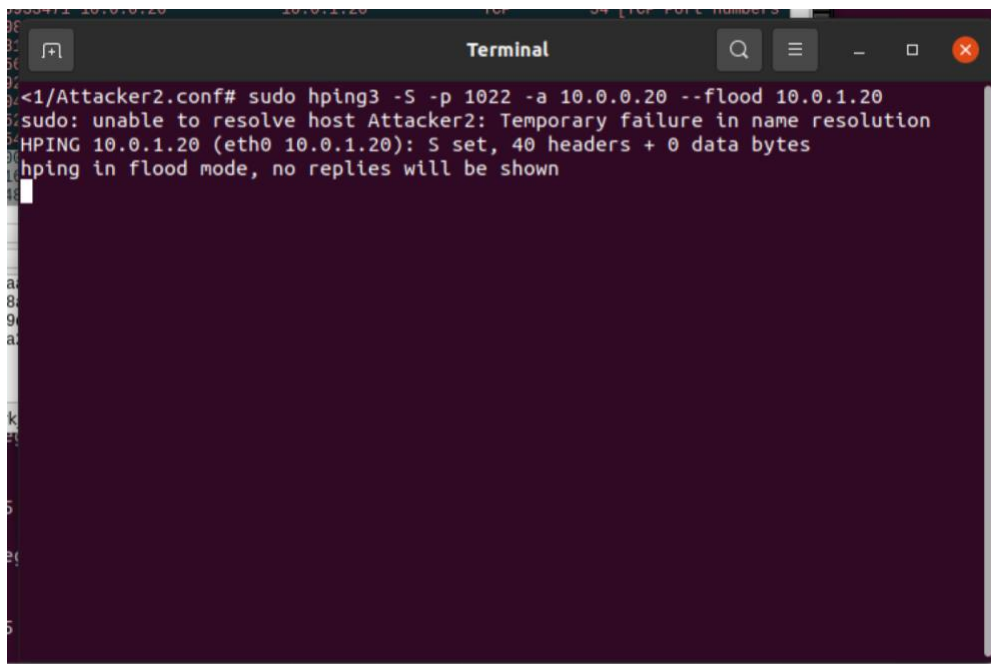
--Thu 05 Sep 2024 04:02:24 PM EDT--
<html>
  <b> Legit Server </b>
</html>

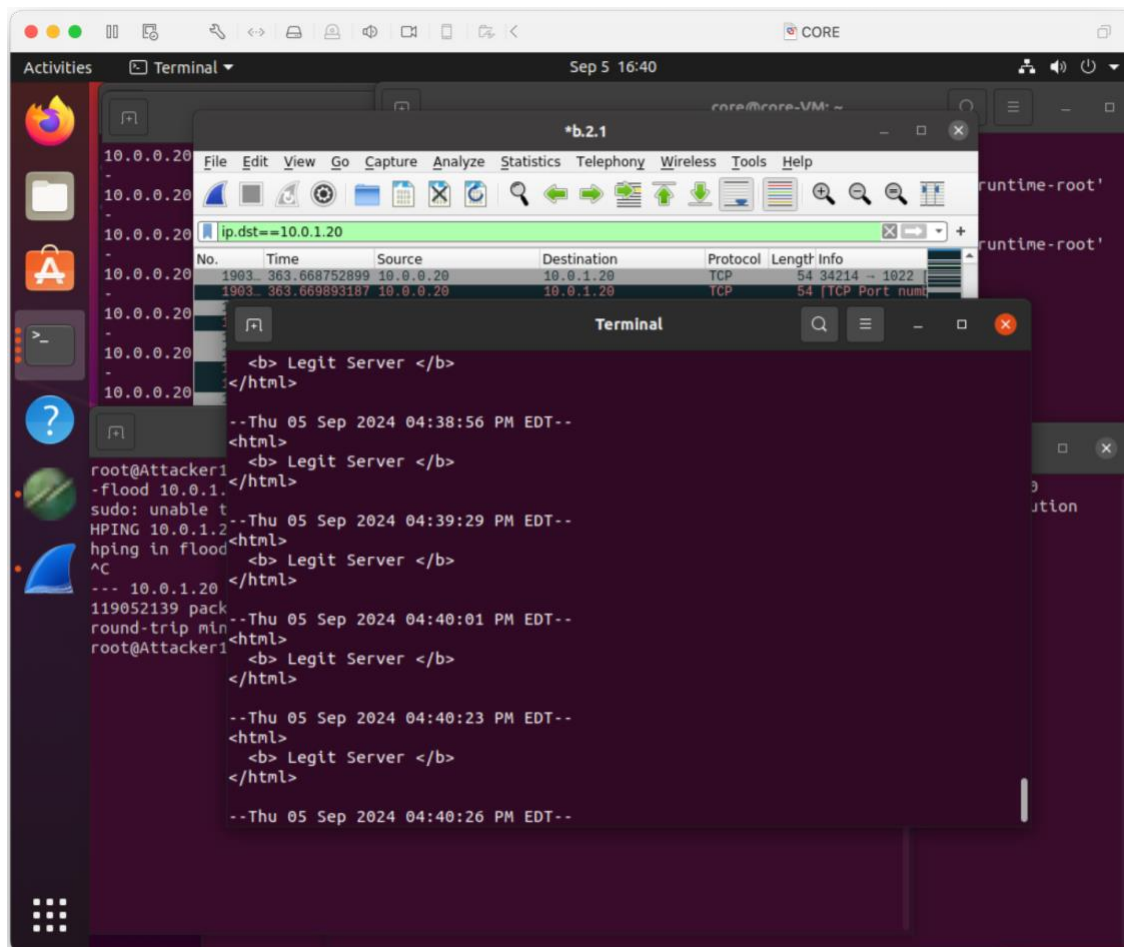
--Thu 05 Sep 2024 04:02:26 PM EDT--
<html>
  <b> Legit Server </b>
</html>
```



f) SYN Flooding Attack with Address Spoofing

I did the SYN Flood attack on the server but changed the command so that the attack reflects the source address as client but originally its source address is attacker 1 node and the same is done from attacker 2 node too. I used the command `sudo hping3 -S -p 1022 -a 10.0.0.20 --flood 10.0.1.20`. I can also see there is a lag in response to the client in 1-minute intervals which means the attack is effective.





The IO graph shows traffic pattern ranges between 80000 bps to 160000 bps. Initially, Attacker 1 node is spoofed to the client's IP, generating traffic at approx. 90000 bps/90 kbps. After some time, Attacker 2 node is spoofed to the client's IP, and this resulted in a significant increase in traffic with the combined attack reaching a peak i.e. magnitude of 160000 bps/160 kbps. See the graph below for a better view.

