Network Security – CSCI_6541_80

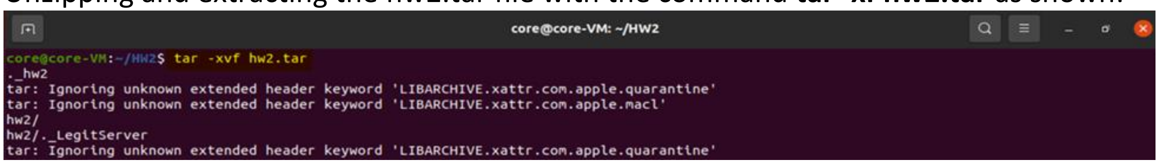
Namana Y Tarikere – G21372717

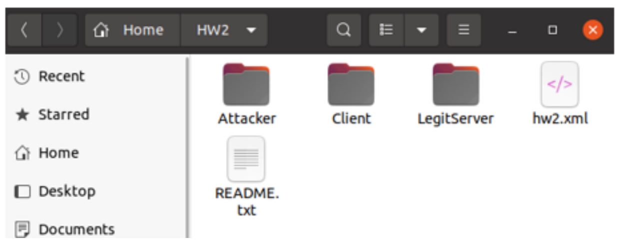
Homework Assignment – 2

ARP poisoning and Man in the Middle (5pts)

Unzip the file hw2.tar: tar -xf hw2.tar. This has a CORE scenario and scripts to run on Client, Attacker, and Server.

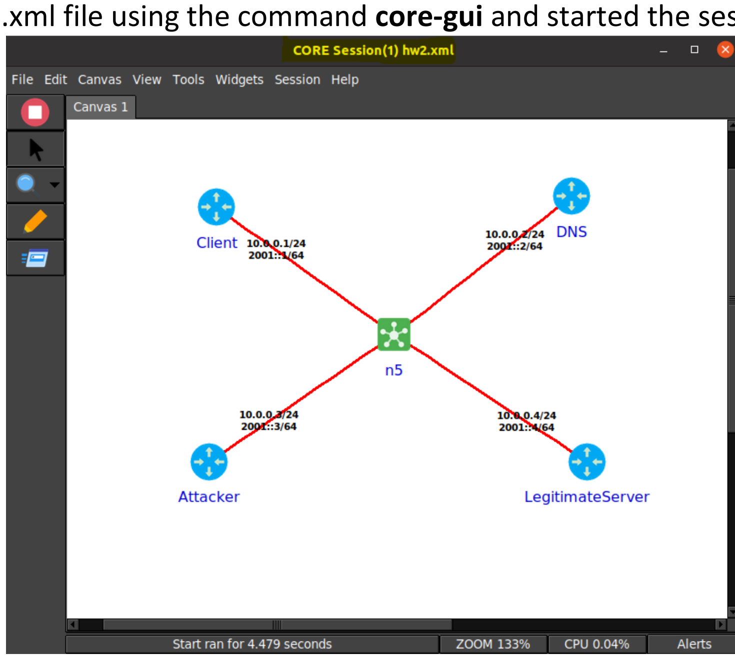
Unzipping and extracting the hw2.tar file with the command **tar -xf hw2.tar** as shown:





Load the file named hw2.xml. You should see the following CORE scenario. The DNS node in the figure below will not be used for the exercise.

Edited the XML file by removing the lines 31 and 57 that said, “service name = HTTP”, loaded the hw2.xml file using the command **core-gui** and started the session as shown:

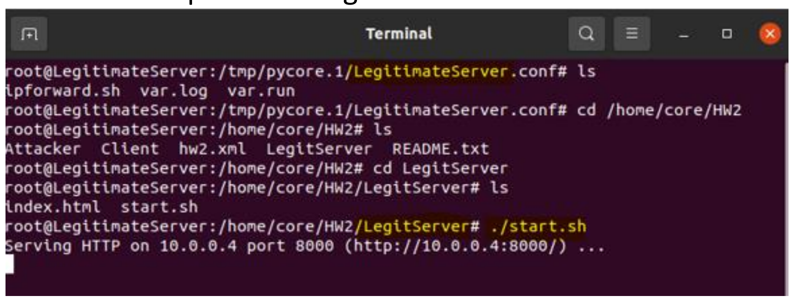


We are going to show the effect of ARP poisoning on this network.

1) **Run an HTTP server on the Server node: LegitServer/start.sh. Inspect the content of the script to understand what it does (very similar to HW1).**

   The script contains the command: **python3 -m http.server -d . -b 10.0.0.4 8000** This command launches the HTTP server with Python3, showing the files from the index.html file and hosting it on port 8000 of the 10.0.0.4 IP address.

   Running the **start.sh** script on the LegitimateServer Node terminal as shown:

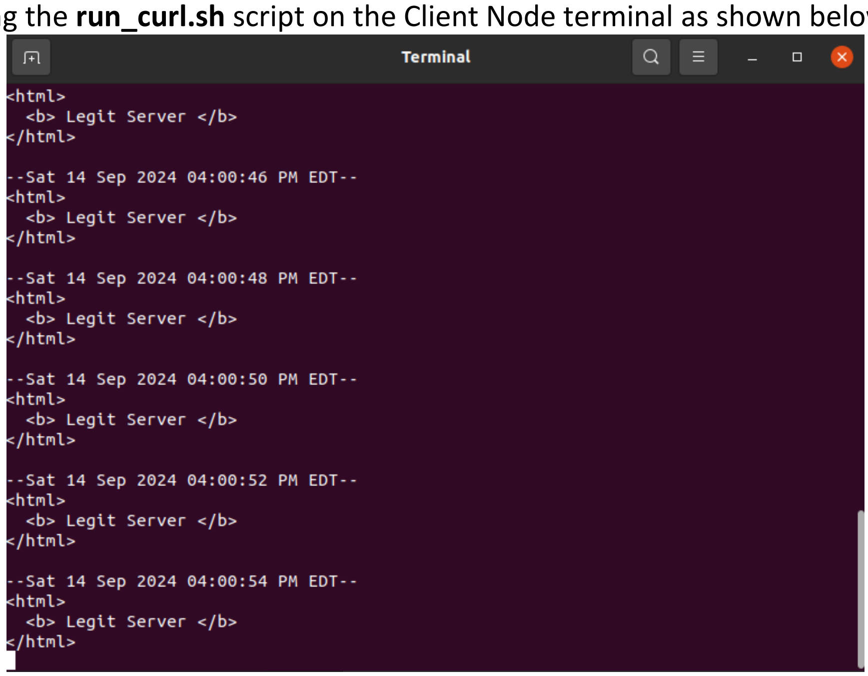


```
Terminal
root@LegitimateServer:/tmp/pycore.1/LegitimateServer.conf# ls
ipforward.sh  var.log  var.run
root@LegitimateServer:/tmp/pycore.1/LegitimateServer.conf# cd /home/core/HW2
root@LegitimateServer:/home/core/HW2# ls
Attacker  Client  hw2.xml  LegitServer  README.txt
root@LegitimateServer:/home/core/HW2# cd LegitServer
root@LegitimateServer:/home/core/HW2/LegitServer# ls
index.html  start.sh
root@LegitimateServer:/home/core/HW2/LegitServer# ./start.sh
Serving HTTP on 10.0.0.4 port 8000 (http://10.0.0.4:8000/) ...
```

2) **Use the Client/run_curl.sh script to request the front page of from the server. Inspect the content of the script to understand what it does (very similar to HW1).**

   This script uses the **curl** command to make an HTTP request to port 8000 on IP address 10.0.0.4. Then it waits for 2 seconds before displaying the HTTP response content along with the current date and time.

   Running the **run_curl.sh** script on the Client Node terminal as shown below:

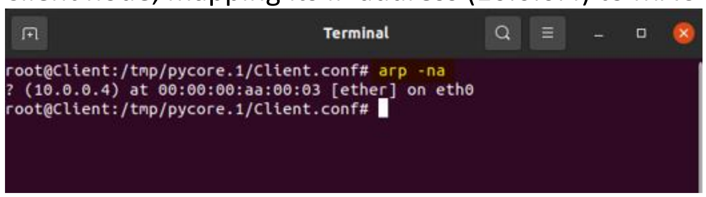


```
Terminal
<html>
  <b> Legit Server </b>
</html>

--Sat 14 Sep 2024 04:00:46 PM EDT--
<html>
  <b> Legit Server </b>
</html>

--Sat 14 Sep 2024 04:00:48 PM EDT--
<html>
  <b> Legit Server </b>
</html>

--Sat 14 Sep 2024 04:00:50 PM EDT--
<html>
  <b> Legit Server </b>
</html>

--Sat 14 Sep 2024 04:00:52 PM EDT--
<html>
  <b> Legit Server </b>
</html>

--Sat 14 Sep 2024 04:00:54 PM EDT--
<html>
  <b> Legit Server </b>
</html>
```

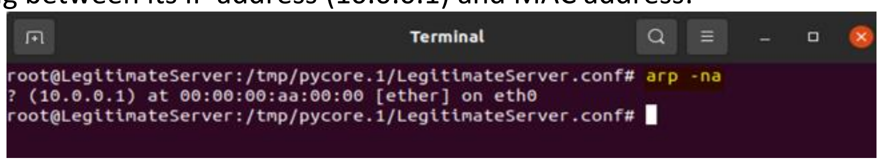
3) **(1 point) The command: arp -na, shows the content of the ARP mapping of IP address to MAC address. Show a screenshot of the ARP entry at the Client that maps the Server's IP to its MAC address.**

The **arp -na** command shows the MAC address linked to the IP address 10.0.0.4. It works by populating the ARP table by sending a query to the local network to find out which device has the MAC address associated with that IP address.

Running the command **arp -na** on the Client node, we can see the ARP entry of the Server at the Client node, mapping its IP address (10.0.0.4) to MAC address.

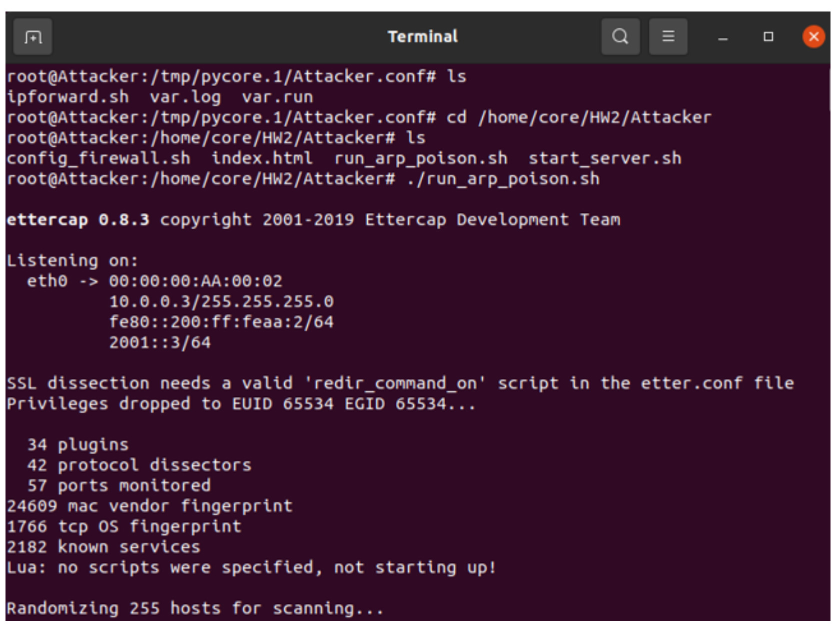


Similarly, here we can see the ARP entry of the Client at the Server node, showing the mapping between its IP address (10.0.0.1) and MAC address:



4) **(1 point) Now run the ARP poison attack on the Attacker using the script: Attacker/run_arp_poison.sh.**

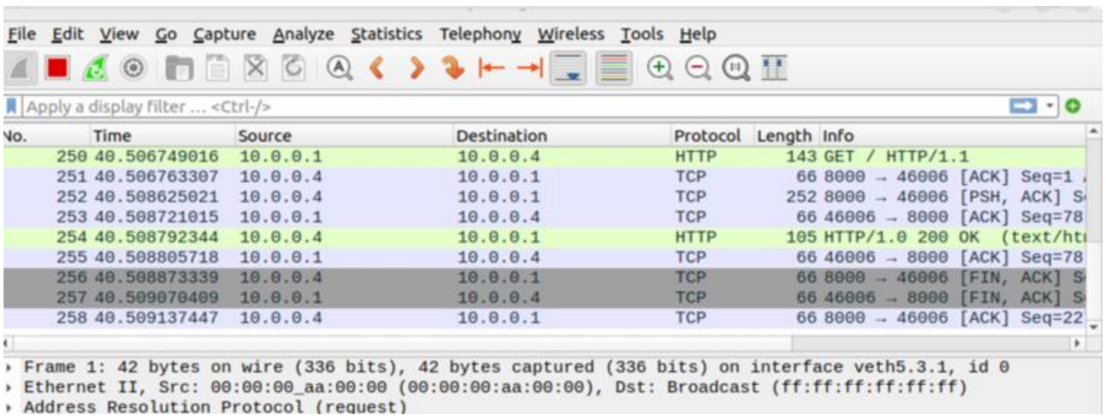
Running the **run_arp_poison.sh** script on the Attacker terminal to conduct ARP poison attack as shown below:



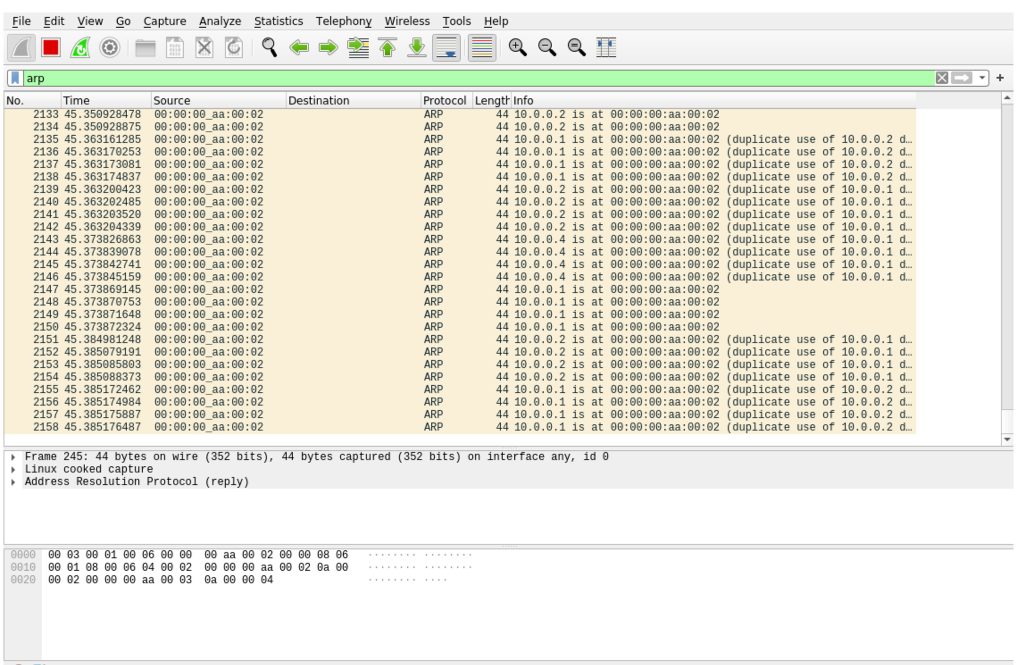
What this does is send a lot of forged ARP packets mapping the IP address of the Server to the MAC address of the Attacker. The Client will accept it and update its local mapping table. So now all the traffic from the Client will go to the Attacker first. For now, the Attacker will happily forward these packets out to the Server without changing anything, so the Client will be getting the HTTP page back. However, the Attacker is in a position to inspect all the traffic between the Client and the Server.

Using Wireshark, show screenshots of forged ARP packets sent by the Attacker.

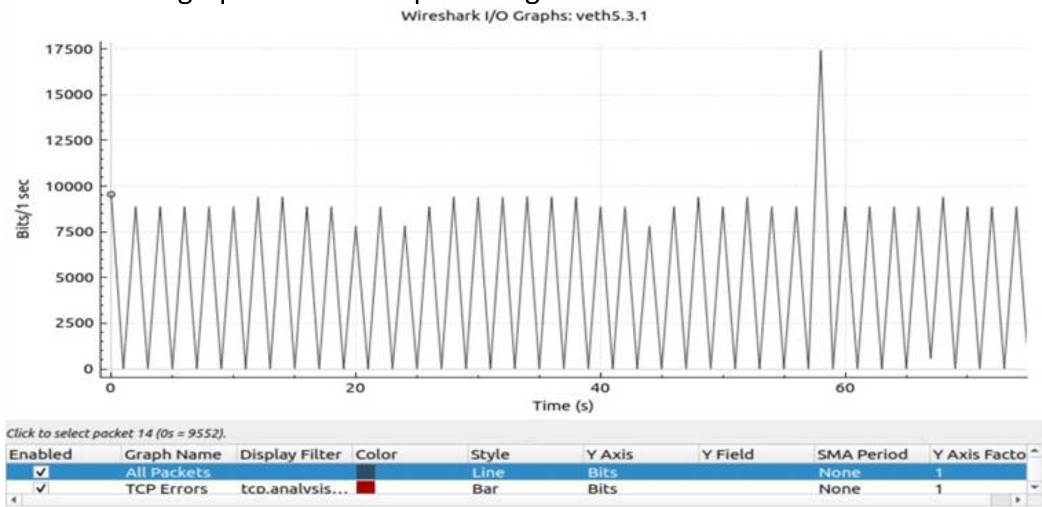
Wireshark interface before the ARP poisoning attack:



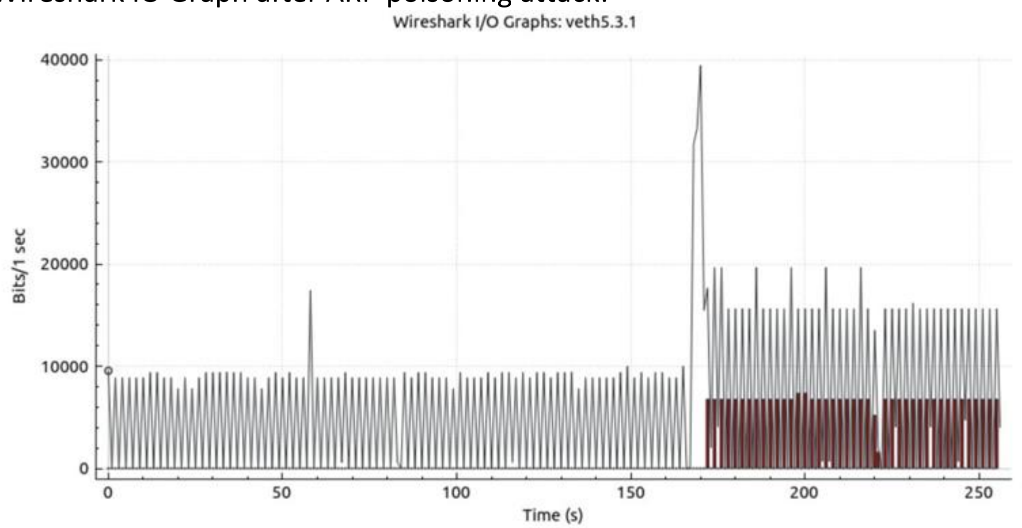
In Wireshark interface, we can see the Attacker sending forged ARP requests with the MAC address 00:00:00:aa:00:02 after the attack:



Wireshark IO graph before ARP poisoning attack:

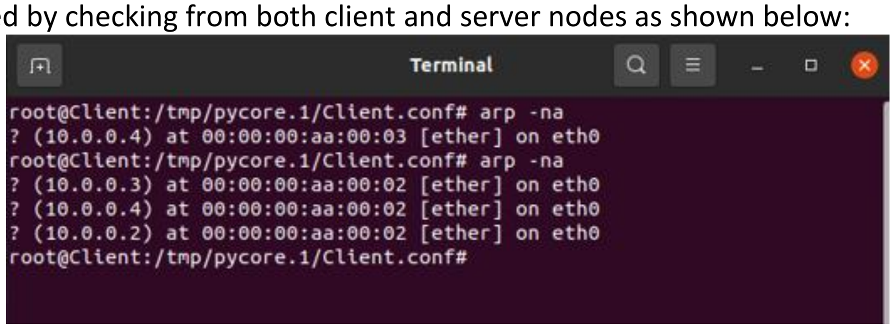


Wireshark IO Graph after ARP poisoning attack:



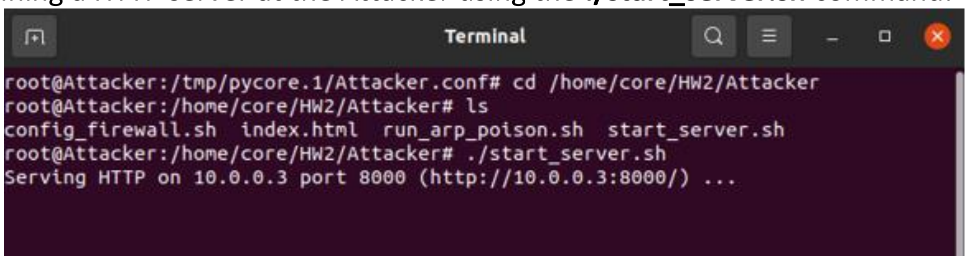
5) (1 point) Repeat 3, notice the mapping changed.

We can now observe multiple ARP entries **mapping different IP addresses to the same MAC address**, which belongs to the attacker (00:00:00:aa:00:02). This is verified by checking from both client and server nodes as shown below:



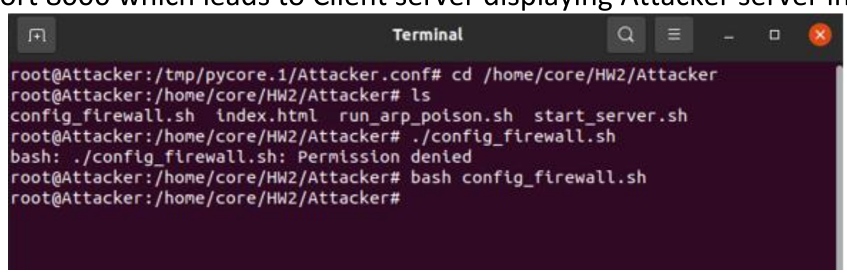
6) **Now run an HTTP server at the Attacker: Attacker/start_server.sh.**

Running a HTTP server at the Attacker using the **./start_server.sh** command:

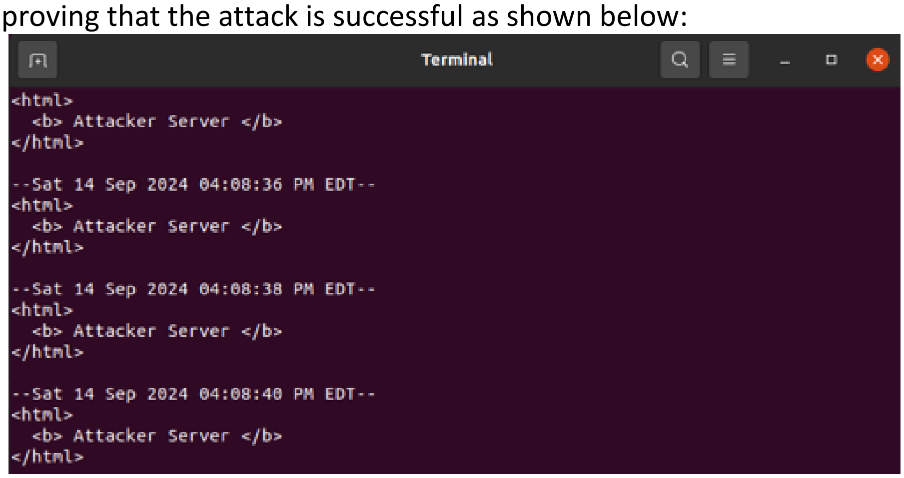


7) **(1 point) Now run config_firewall.sh. What this will do is install a firewall rule on the Attacker that forces the Client's HTTP request to go to the Attacker's HTTP server as opposed to the Server. You should notice now the page you get back is different. Show a screenshot.**

Running the **confirg_firewall.sh** script on the Attacker node redirects TCP traffic from port 8000 which leads to Client server displaying Attacker server info:



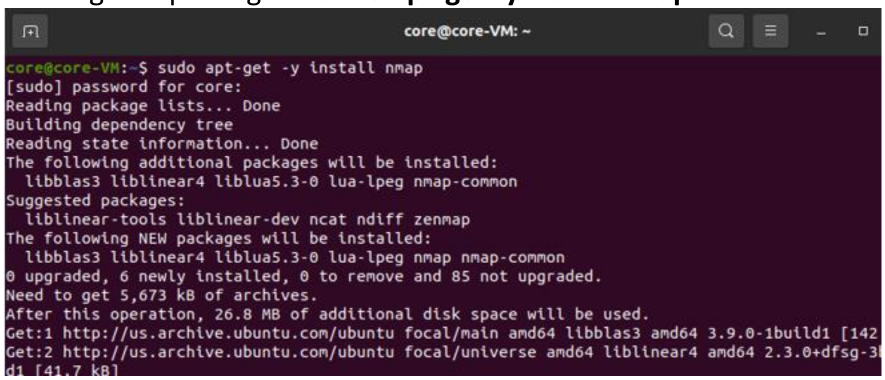
Now, we can see that the client's HTTP request is re-routed to the attacker's HTTP server proving that the attack is successful as shown below:



8) **(1 point) Perform an OS fingerprinting scan**

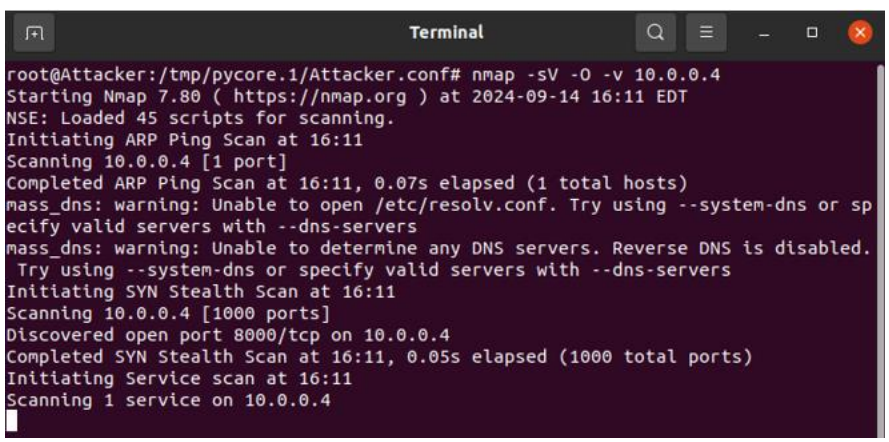
   a. **Install nmap: sudo apt-get -y install nmap (assumes your host is connected to the Internet and the VM interface that is configured to use NAT is connected).**

      Installing nmap using the **sudo apt-get -y install nmap** command:



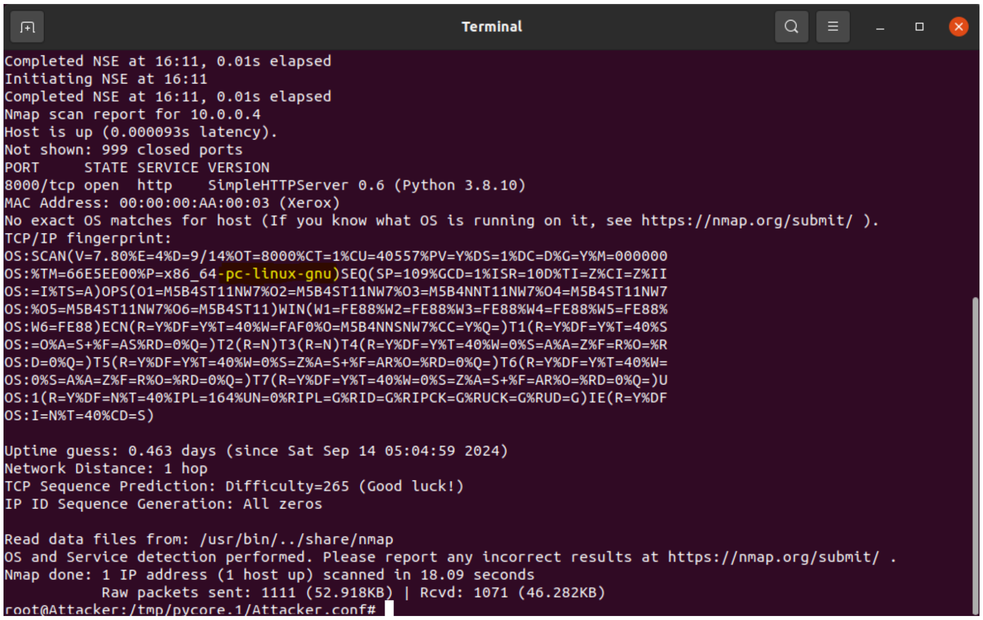
   b. **You can run the following command to perform an OS fingerprinting scan from attacker on the server: nmap –sV –O –v 10.0.0.4.**

      Performing the **OS Fingerprinting Scan** from the Attacker node on the Server using the **nmap -sV -O -v 10.0.0.4** command as shown below:



   c. **Was this scan able to guess the OS of the server?**

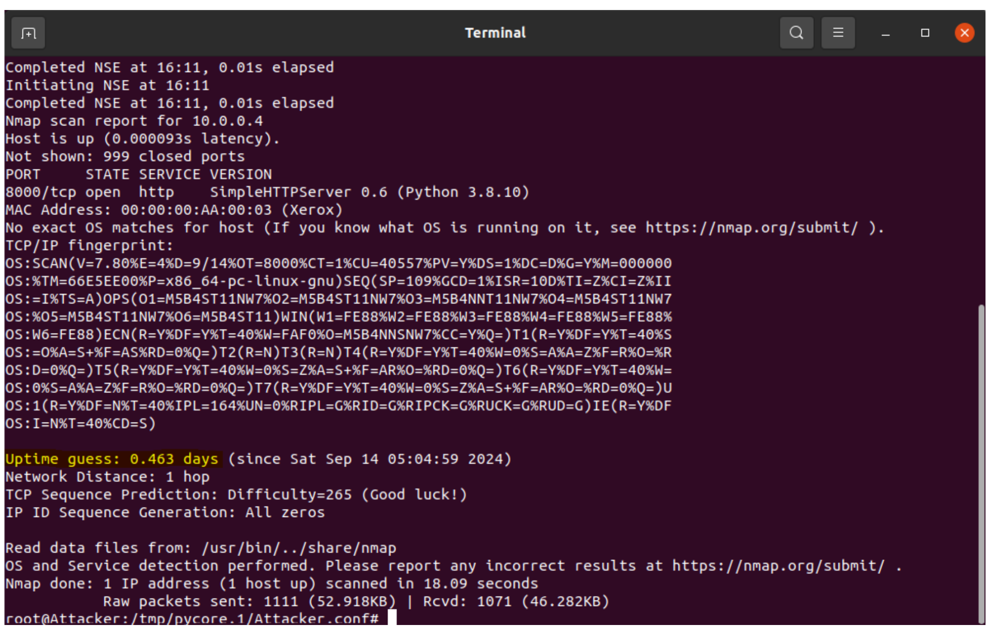
      The nmap scan couldn't identify an exact OS of the server since there is only one open port, but the TCP/IP fingerprint indicates that it could be a Linux system as shown below:



```
Completed NSE at 16:11, 0.01s elapsed
Initiating NSE at 16:11
Completed NSE at 16:11, 0.01s elapsed
Nmap scan report for 10.0.0.4
Host is up (0.000093s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE VERSION
8000/tcp open  http    SimpleHTTPServer 0.6 (Python 3.8.10)
MAC Address: 00:00:00:AA:00:03 (Xerox)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.80%E=4%D=9/14%OT=8000%CT=1%CU=40557%PV=Y%DS=1%DC=D%G=Y%M=000000
OS:%TM=66E5EE00%P=x86_64-pc-linux-gnu)SEQ(SP=109%GCD=1%ISR=10D%TI=Z%CI=Z%II
OS:=I%TS=A)OPS(O1=M5B4ST11NW7%O2=M5B4ST11NW7%O3=M5B4NNT11NW7%O4=M5B4ST11NW7
OS:%O5=M5B4ST11NW7%O6=M5B4ST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%
OS:W6=FE88)ECN(R=Y%DF=Y%T=40%W=FAF0%O=M5B4NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S
OS:=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%R
OS:D=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=
OS:0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U
OS:1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DF
OS:I=N%T=40%CD=S)

Uptime guess: 0.463 days (since Sat Sep 14 05:04:59 2024)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=265 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.09 seconds
           Raw packets sent: 1111 (52.918KB) | Rcvd: 1071 (46.282KB)
root@Attacker:/tmp/pycore.1/Attacker.conf#
```

d. What is the "uptime of the server"? What does that mean? How can this information be used by an attacker?

The uptime of the server shows the amount of time that has passed since the server was last started and it resets after each update or patchwork. The uptime of the server here is **0.463 days** (since Sat, Aug 14, 2024). This means that the system has been stable and functional for 0.463 days without any rebooting.



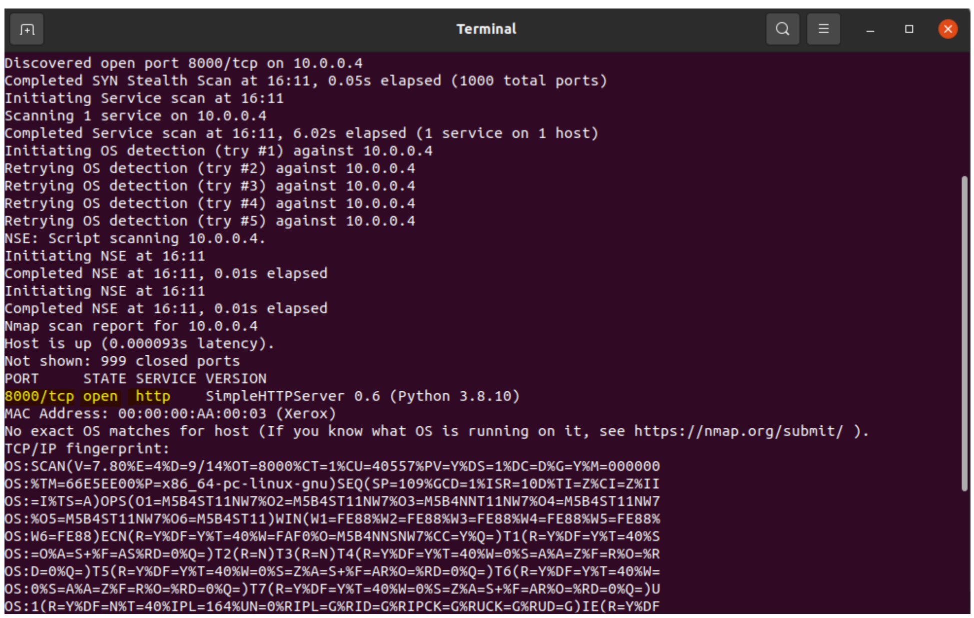

```
Completed NSE at 16:11, 0.01s elapsed
Initiating NSE at 16:11
Completed NSE at 16:11, 0.01s elapsed
Nmap scan report for 10.0.0.4
Host is up (0.000093s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE VERSION
8000/tcp open  http    SimpleHTTPServer 0.6 (Python 3.8.10)
MAC Address: 00:00:00:AA:00:03 (Xerox)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.80%E=4%D=9/14%OT=8000%CT=1%CU=40557%PV=Y%DS=1%DC=D%G=Y%M=000000
OS:%TM=66E5EE00%P=x86_64-pc-linux-gnu)SEQ(SP=109%GCD=1%ISR=10D%TI=Z%CI=Z%II
OS:=I%TS=A)OPS(O1=M5B4ST11NW7%O2=M5B4ST11NW7%O3=M5B4NNT11NW7%O4=M5B4ST11NW7
OS:%O5=M5B4ST11NW7%O6=M5B4ST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%
OS:W6=FE88)ECN(R=Y%DF=Y%T=40%W=FAF0%O=M5B4NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S
OS:=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%R
OS:D=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=
OS:0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U
OS:1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DF
OS:I=N%T=40%CD=S)

Uptime guess: 0.463 days (since Sat Sep 14 05:04:59 2024)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=265 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.09 seconds
           Raw packets sent: 1111 (52.918KB) | Rcvd: 1071 (46.282KB)
root@Attacker:/tmp/pycore.1/Attacker.conf#
```

An attacker could use system uptime details to plan their attack more effectively by:

- Choosing a time when a reboot or maintenance is least likely, like immediately after a reboot.
- Gauging for any potential vulnerabilities in the system. A long uptime in the system can suggest that the server has not been updated or patched recently, making it more vulnerable to attacks.
- Since uptime indicates how long the server has been running, an attacker might use this information to exploit the server.

e. What ports were identified as open on the server? Were the corresponding services identified?

Only the **port 8000/tcp** was identified as open on the server which shows us the service and version running on the port. Here, the server running on **port 8000 is HTTP** and its version is **SimpleHTTPServer 0.6 running on Python 3.8.10** as shown in the screenshot below:




```
Discovered open port 8000/tcp on 10.0.0.4
Completed SYN Stealth Scan at 16:11, 0.05s elapsed (1000 total ports)
Initiating Service scan at 16:11
Scanning 1 service on 10.0.0.4
Completed Service scan at 16:11, 6.02s elapsed (1 service on 1 host)
Initiating OS detection (try #1) against 10.0.0.4
Retrying OS detection (try #2) against 10.0.0.4
Retrying OS detection (try #3) against 10.0.0.4
Retrying OS detection (try #4) against 10.0.0.4
Retrying OS detection (try #5) against 10.0.0.4
NSE: Script scanning 10.0.0.4.
Initiating NSE at 16:11
Completed NSE at 16:11, 0.01s elapsed
Initiating NSE at 16:11
Completed NSE at 16:11, 0.01s elapsed
Nmap scan report for 10.0.0.4
Host is up (0.000093s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE VERSION
8000/tcp open  http    SimpleHTTPServer 0.6 (Python 3.8.10)
MAC Address: 00:00:00:AA:00:03 (Xerox)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.80%E=4%D=9/14%OT=8000%CT=1%CU=40557%PV=Y%DS=1%DC=D%G=Y%M=000000
OS:%TM=66E5EE00%P=x86_64-pc-linux-gnu)SEQ(SP=109%GCD=1%ISR=10D%TI=Z%CI=Z%II
OS:=I%TS=A)OPS(O1=M5B4ST11NW7%O2=M5B4ST11NW7%O3=M5B4NNT11NW7%O4=M5B4ST11NW7
OS:%O5=M5B4ST11NW7%O6=M5B4ST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%
OS:W6=FE88)ECN(R=Y%DF=Y%T=40%W=FAF0%O=M5B4NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S
OS:=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%R
OS:D=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=
OS:0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U
OS:1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DF
```