Veermata Jijabai Technological Institute, Mumbai 400019

Experiment No.: 02

Aim: Perform two/three attacks from each layer of TCP/IP

Group : Kiran Patil - 211070904

Mayuresh Murudkar - 211070903

Pratiksha Sankhe – 201071049

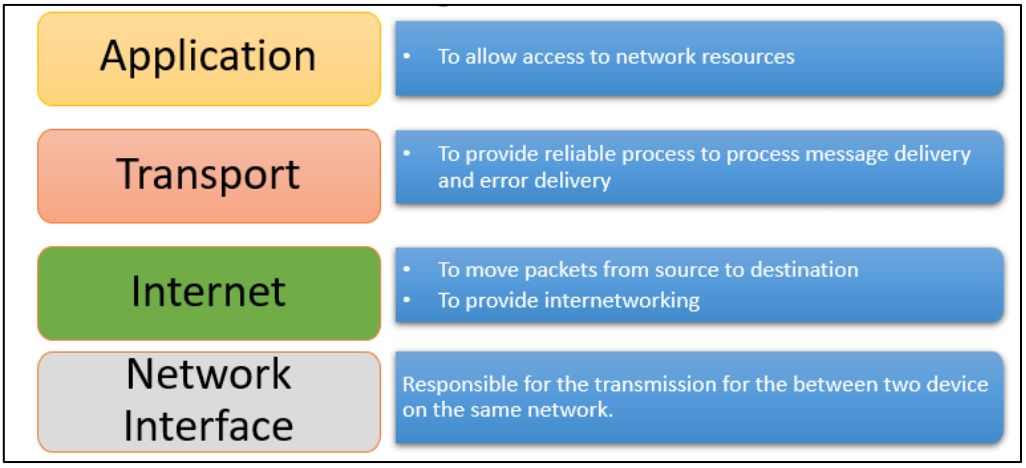
Branch: Computer Engineering

Batch: D

Theory:

TCP/IP Model helps you to determine how a specific computer should be connected to the internet and how data should be transmitted between them. It helps you to create a virtual network when multiple computer networks are connected together. The purpose of the TCP/IP model is to allow communication over large distances.

TCP/IP stands for Transmission Control Protocol/ Internet Protocol. TCP/IP Stack is specifically designed as a model to offer highly reliable and end-to-end byte stream over an unreliable internetwork.



1. Application Layer

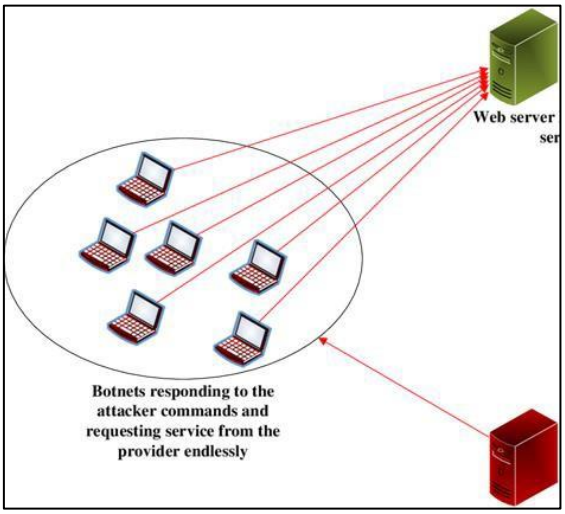
DNS spoofing or DNS cache poisoning is an attack in which altered DNS records are used to redirect users or data to a fraudulent website or link that is camouflaged as the actual destination.

1. **SQL Injection (SQLi):** Attackers inject malicious SQL queries into input fields of a web application to manipulate the database or gain unauthorized access.
2. **Cross-Site Scripting (XSS):** Malicious scripts are injected into web pages to steal session cookies, redirect users, or perform other malicious actions.
3. **Cross-Site Request Forgery (CSRF):** Users are tricked into performing actions on a website without their consent, potentially resulting in unauthorized changes.
4. **Man-in-the-Middle (MitM) Attacks:** Attackers intercept and possibly alter communication between two parties, exploiting vulnerabilities in application layer protocols.
5. **Distributed Denial of Service (DDoS):** Application layer services are overwhelmed with a high volume of traffic, causing them to become slow or unresponsive.
6. **Brute Force Attacks:** Attackers repeatedly try different username and password combinations to gain unauthorized access to an application.

7. **Phishing Attacks:** Users are tricked into divulging sensitive information through deceptive websites or emails.
8. **Session Hijacking:** Attackers steal or manipulate session tokens to impersonate a user and access their account.
9. **Application Layer DoS Attacks:** Overload application layer resources, making them unavailable by targeting vulnerabilities in application code.
10. **Buffer Overflow Attacks:** Attackers exploit vulnerabilities by overflowing buffer space with excessive data, potentially executing arbitrary code.

1) HTTP Flood DDoS Attack:

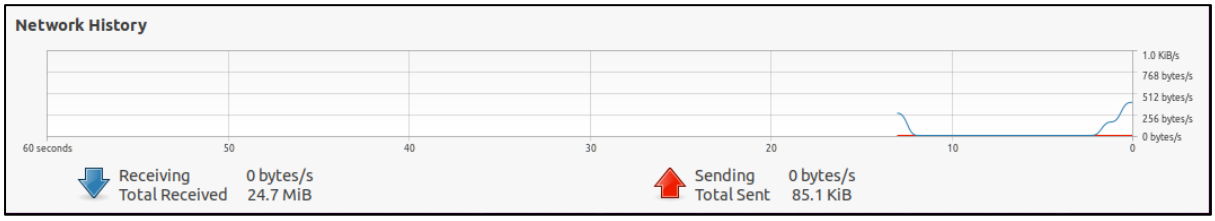
An HTTP flood DDoS attack utilizes what appear to be legitimate HTTP GET or POST requests to attack a web server or application. These flooding DDoS attacks often rely on a botnet, which is a group of Internet-connected computers that have been maliciously appropriated through the use of malware such as a Trojan Horse.



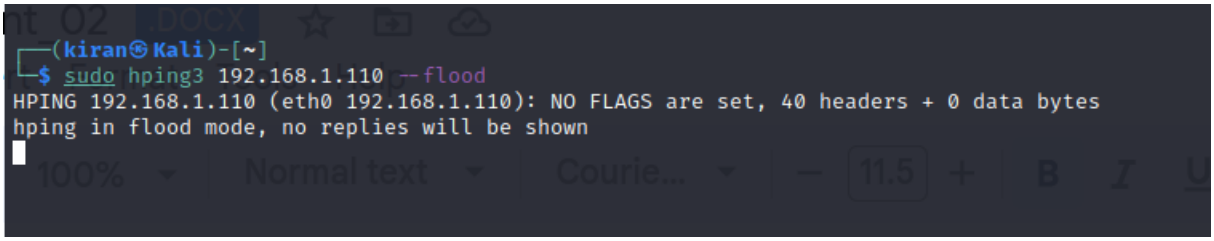
Victim IP address:

```
kiran@kiran-VirtualBox: ~
kiran@kiran-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.110  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::e296:4a3b:855a:2317  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:11:9a:c7  txqueuelen 1000  (Ethernet)
        RX packets 1539  bytes 1399079 (1.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 384  bytes 33945 (33.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 161  bytes 13755 (13.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 161  bytes 13755 (13.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

kiran@kiran-VirtualBox:~$
```

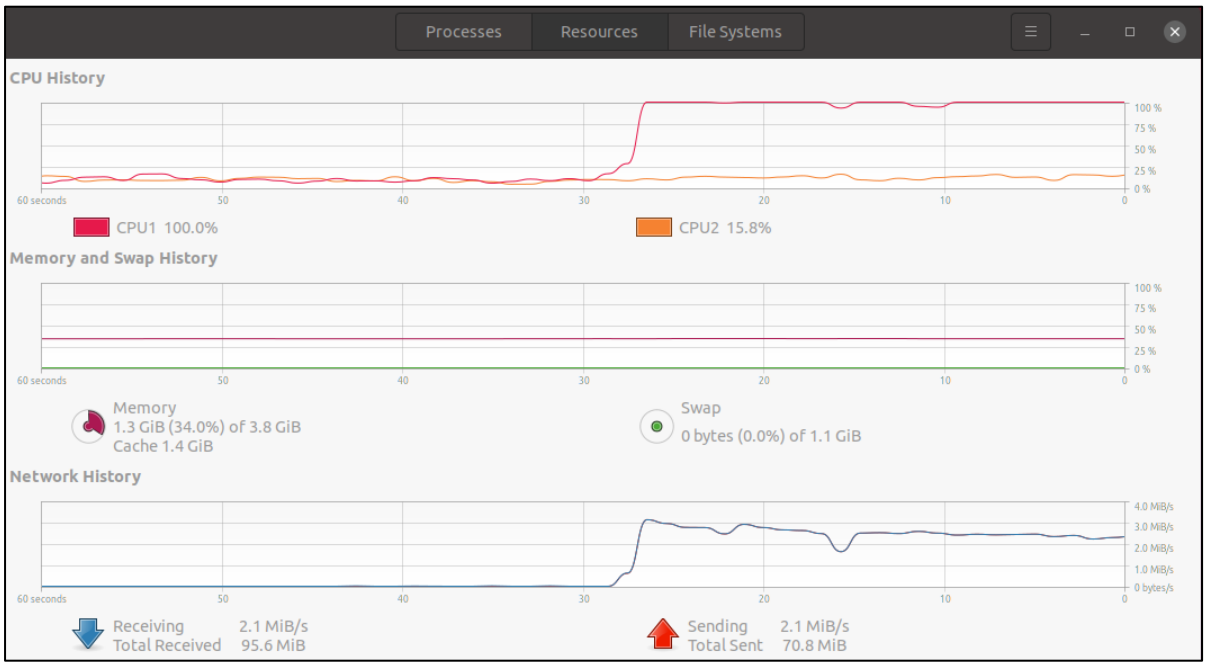
Network Traffic Before Attack:



Attack using hping:



Task manager after the attack:



These types of DDoS attacks are designed to cause the targeted server or application to allocate the most resources possible in direct response to each request. In this way, the attacker hopes to overwhelm the server or application, “flooding” it with as many process-intensive requests as possible.

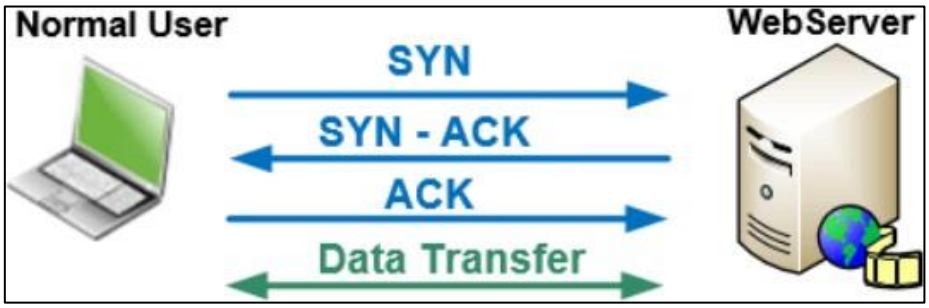
HTTP POSTs are often used because they involve complex server-side processing, while HTTP GET attacks are easier to create, thus lending themselves to botnet attacks which rely on scale to achieve the desired disruption.

2.Transport Layer

1) TCP SYN Flood :

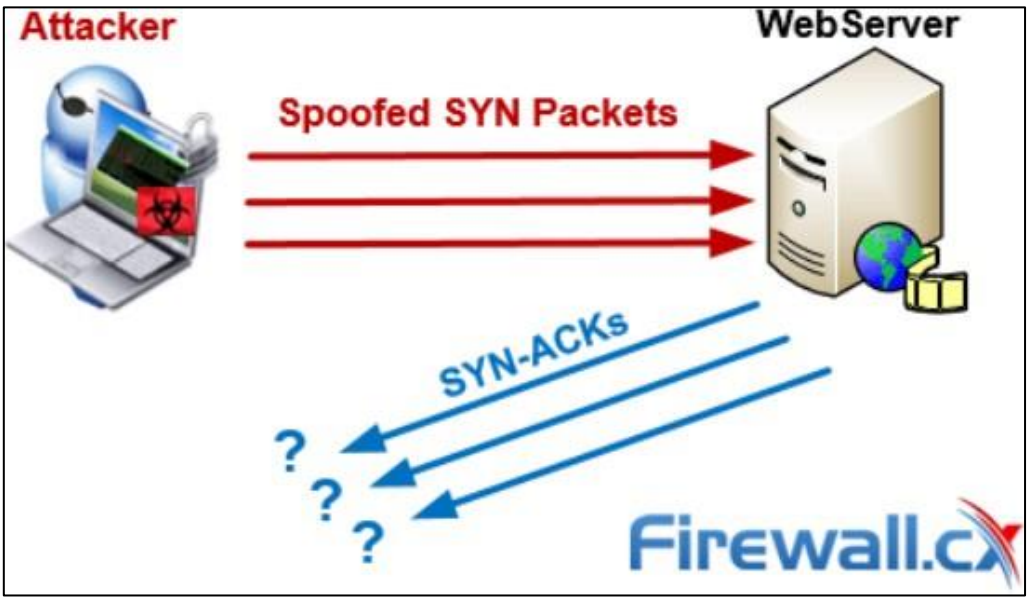
In a SYN flood attack, a malicious party exploits the TCP protocol 3-way handshake to quickly cause service and network disruptions, ultimately leading to an Denial of Service (DoS) Attack.

Normal TCP 3-way handshake:

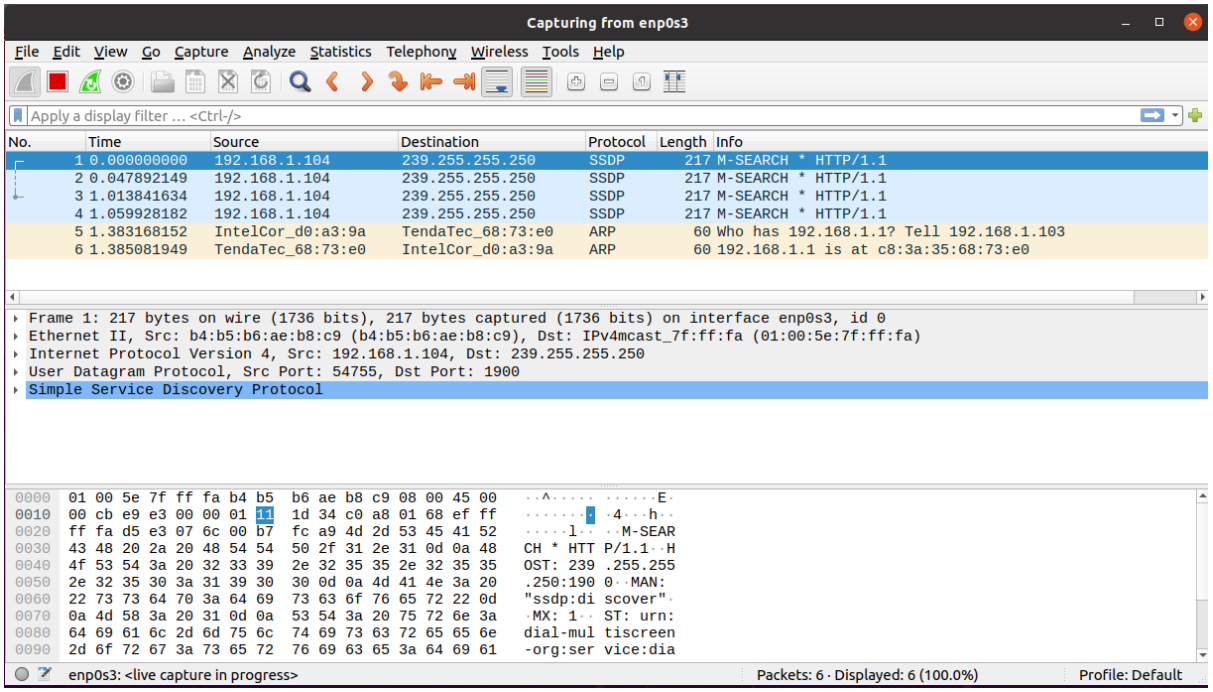


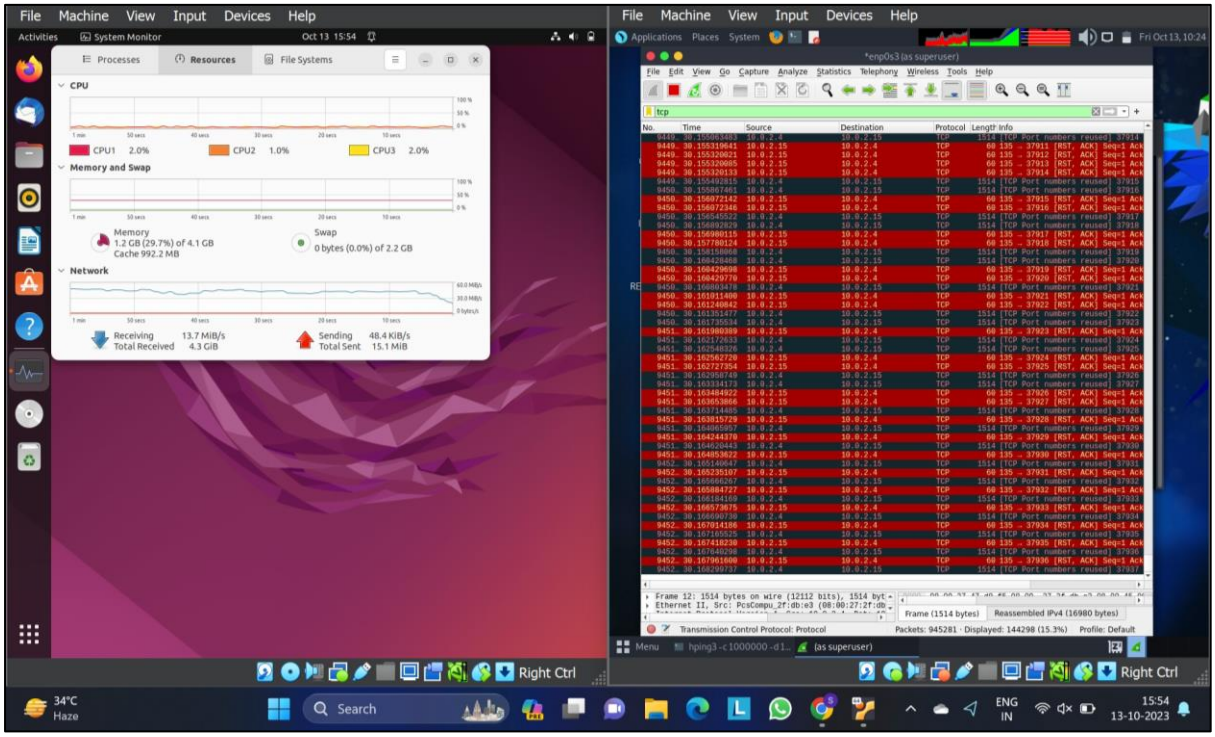
TCP SYN Flood Attack:

The attacker sends a high volume of SYN packets to the server using spoofed IP addresses causing the server to send a reply (SYN-ACK) and leave its ports half-open, awaiting for a reply from a host that doesn't exist.



Traced packets before the Tcp Syn flood attack



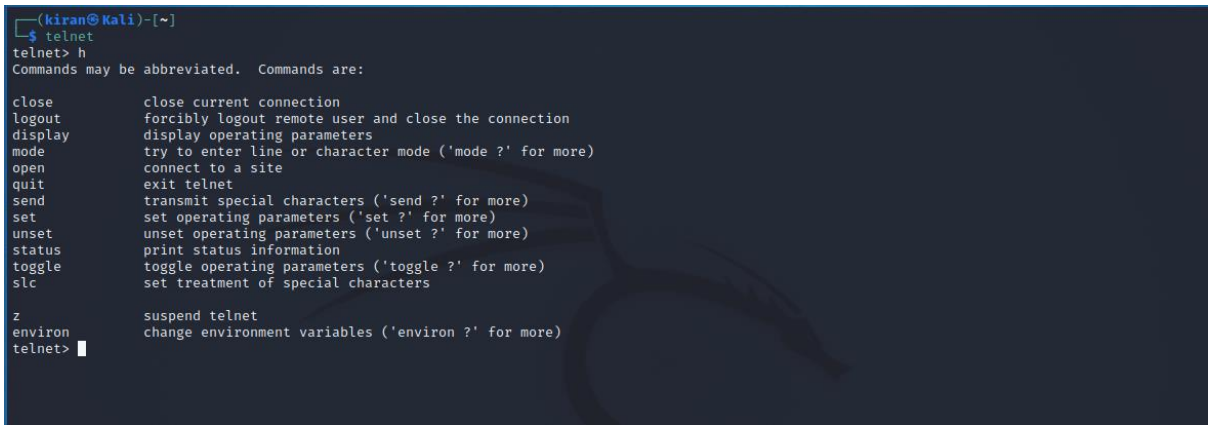


2) Session Hijacking:

Session hijacking is a critical error and gives an chance to the malicious node to behave as a legitimate system. All the communications are authentic only at the beginning of session setup. The attacker may take the advantage of this and commit session hijacking attack. At first, he or she spoofs the IP address of target machine and controls the correct sequence number. After that he performs a DoS attack on the victim. As a result, the target system becomes absent for some time. Thus the attacker imitates the victim node and continues the session. Hijacking a session over UDP is the same as over TCP, except that UDP attackers not to concern about the overhead of dealing sequence numbers and other TCP mechanisms. Since UDP is connectionless, edging into a session without being detected much easier than the TCP session attacks.

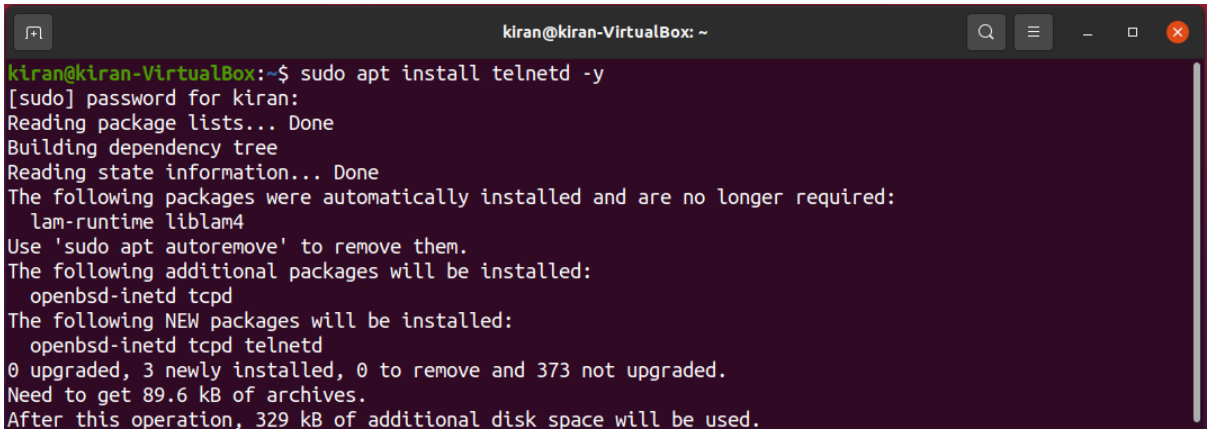
We hijacked the TCP session in the telnet connection by retrieving the source port, sequence number and ack number of the tcp request snooped in the attacker machine using wireshark while making connection requests between client and server. Then used scapy to make a tcp request using the data retrieved to get the required file from the server.

We installed the telnet on the server machine and made changes in the configuration for the host.



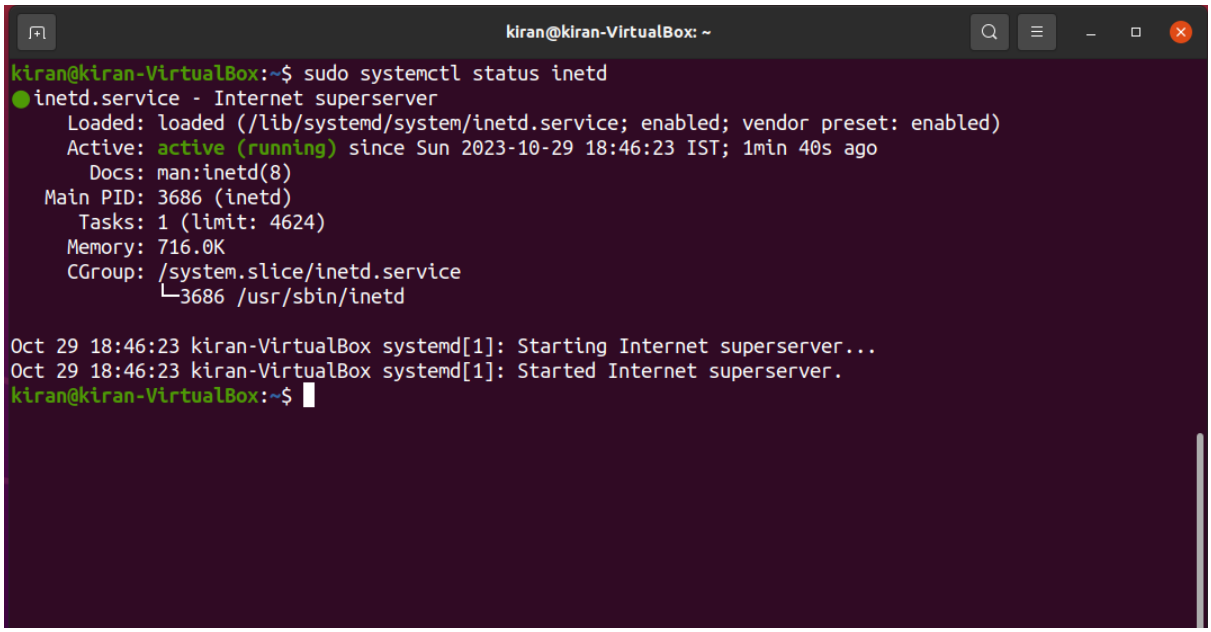
We enabled the telnet on client machine (Ubuntu) by running the following commands in terminal:

sudo apt install telnetd -y



Check if telnet is properly installed by running:

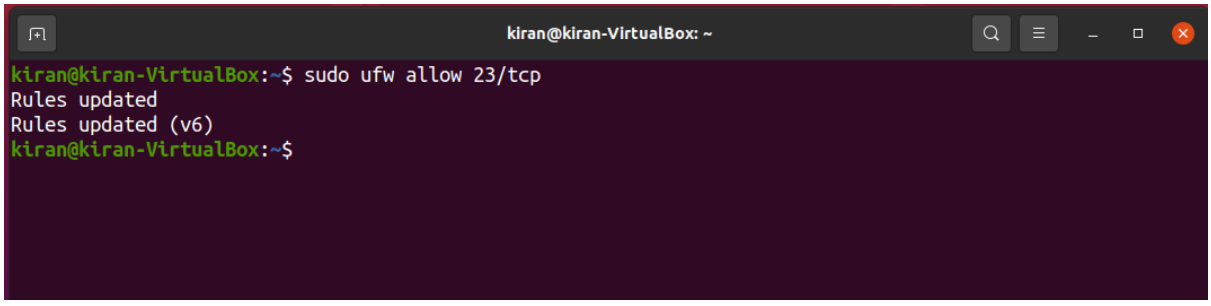
sudo systemctl status inetd



The output shows that the daemon is up and running.

Allow port 23 through the firewall on the remote machine by running:

sudo ufw allow 23/tcp

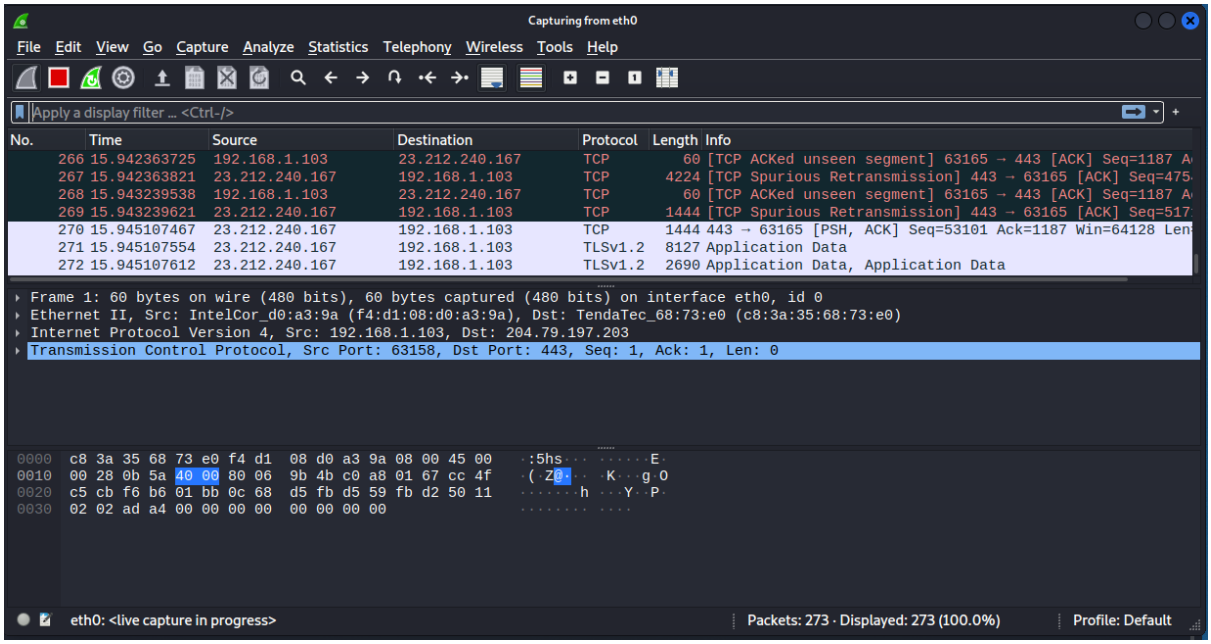


Reload the firewall:

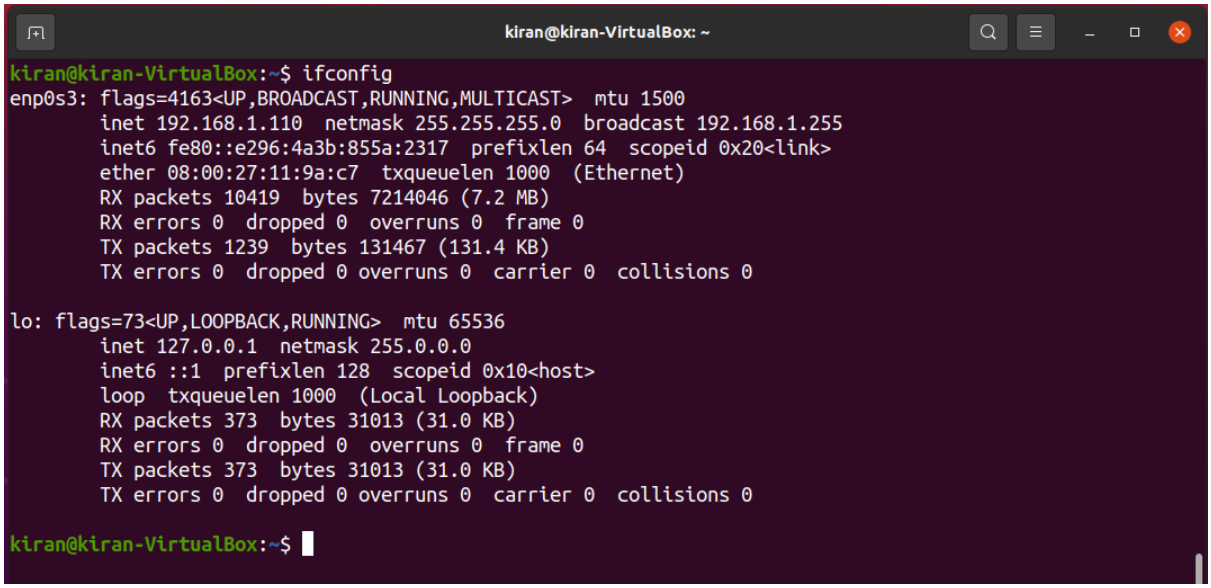
sudo ufw reload

The telnet port is now allowed through the firewall.

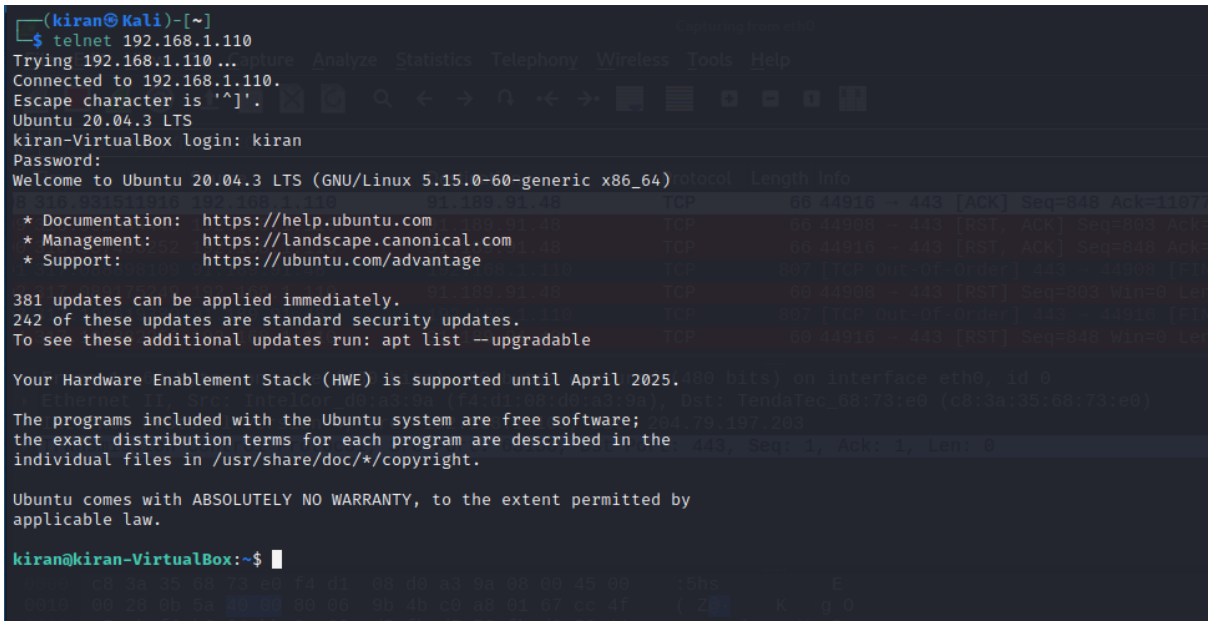
Before making the connection we started capturing packets using wireshark in the attacker machine



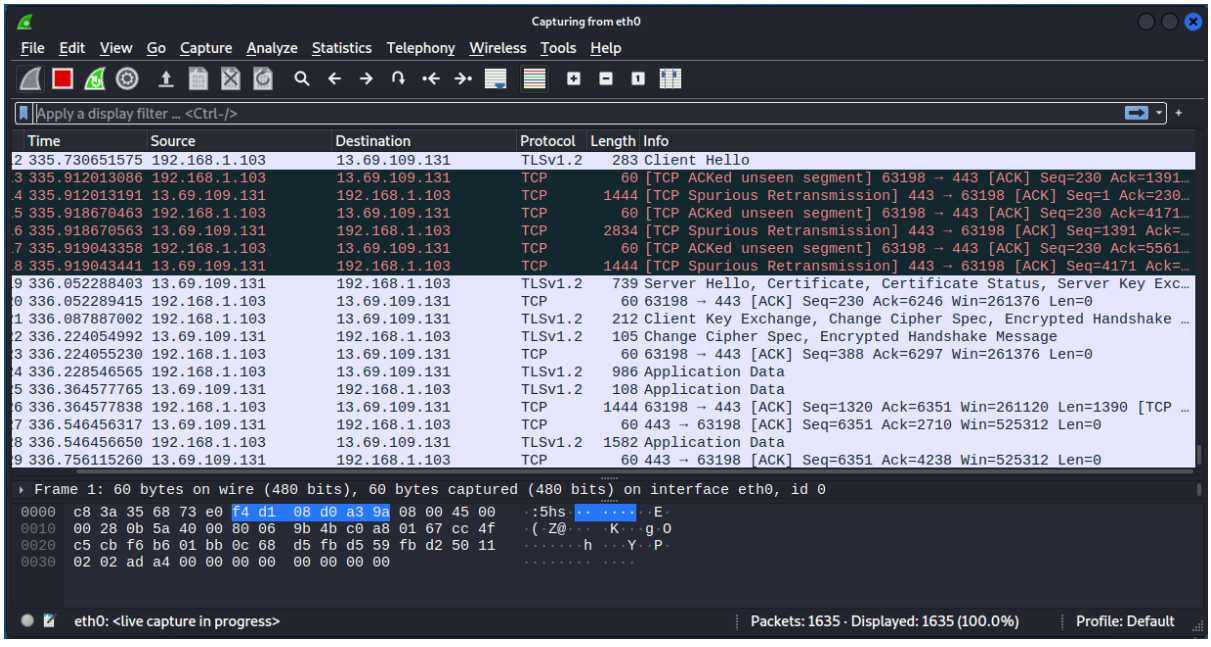
Victim's IP address:



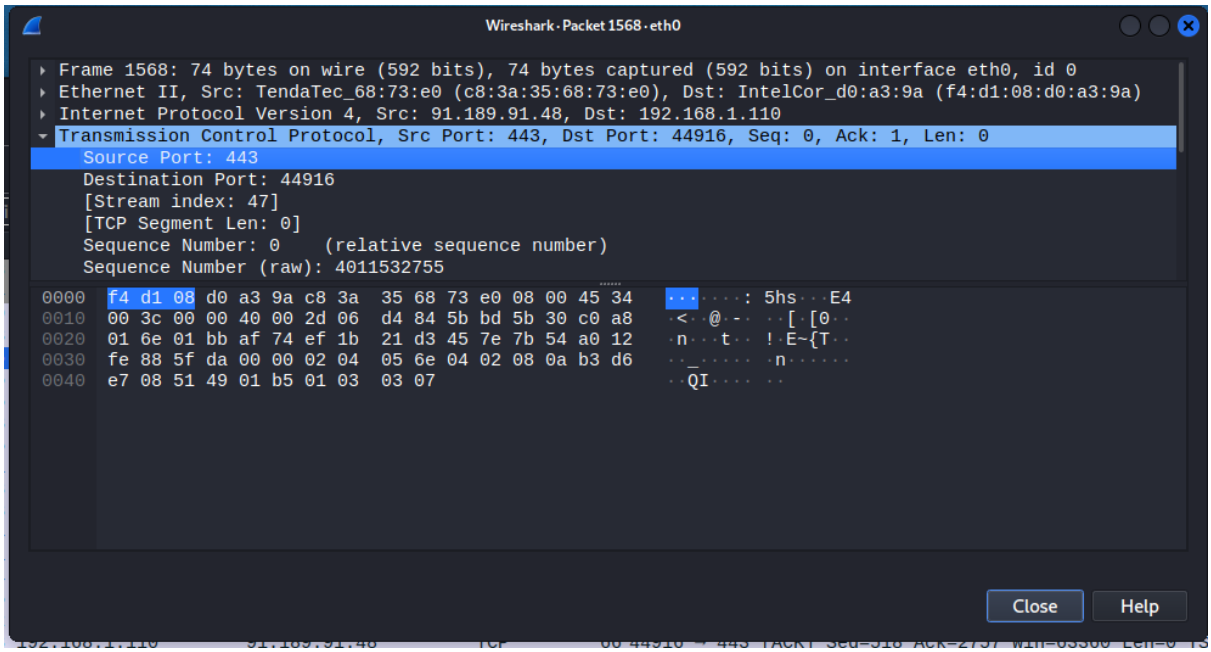
Then we made the telnet connection between client and server by running the command **telnet <host_ip>.** Then entered the login id and password of the host.



We can see the TCP requests between client and server:



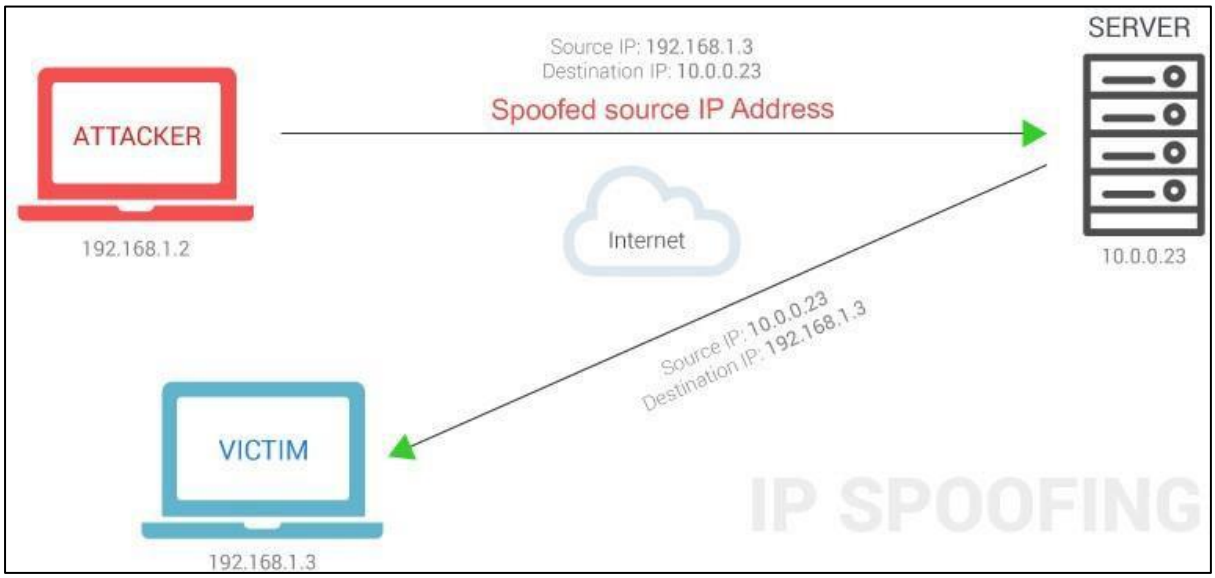
We opened the last TCP packet and noted the source port, sequence number and ack number of the tcp request using this we made the tcp request and using scapy we made the TCP request to retrieve the file from the server. We sent this file output to port no 33267 where we printed the file contents.



3. Internet Layer

1) IP Spoofing:

Spoofing is a specific type of cyber-attack in which someone attempts to use a computer, device, or network to trick other computer networks by masquerading as a legitimate entity. It's one of many tools hackers use to gain access to computers to mine them for sensitive data, turn them into zombies (computers taken over for malicious use), or launch Denial-of-Service (DoS) attacks. Of the several types of spoofing, IP spoofing is the most common.



Install tor browser

Sudo apt-get install tor

```
kiran@kiran-VirtualBox:~$ sudo apt-get install tor
[sudo] password for kiran:
Reading package lists ... Done
Building dependency tree
Reading state information ... Done
The following packages were automatically installed and are no longer required:
  lam-runtime liblam4
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  tor-geoipdb torsocks
Suggested packages:
  mixmaster torbrowser-launcher socat tor-arm apparmor-utils obfs4proxy
The following NEW packages will be installed:
  tor tor-geoipdb torsocks
0 upgraded, 3 newly installed, 0 to remove and 373 not upgraded.
Need to get 2,439 kB of archives.
After this operation, 13.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 tor amd64 0.4.2.7-1 [1,410 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 torsocks amd64 2.3.0-2 [61.5 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 tor-geoipdb all 0.4.2.7-1 [968 kB]
Fetched 2,439 kB in 10s (246 kB/s)
```

Public IP address:

Install curl

Sudo apt install curl

```
kiran@kiran-VirtualBox:~$ sudo apt install curl
Reading package lists ... Done
Building dependency tree
Reading state information ... Done
The following packages were automatically installed and are no longer required:
  lam-runtime liblam4
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  curl
The following packages will be upgraded:
  libcurl4
1 upgraded, 1 newly installed, 0 to remove and 372 not upgraded.
Need to get 396 kB of archives.
After this operation, 423 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 libcurl4 amd64 7.68.0-1ubuntu2.20 [235 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 curl amd64 7.68.0-1ubuntu2.20 [161 kB]
Fetched 396 kB in 1s (305 kB/s)
(Reading database ... 230854 files and directories currently installed.)
Preparing to unpack .../libcurl4_7.68.0-1ubuntu2.20_amd64.deb ...
Unpacking libcurl4:amd64 (7.68.0-1ubuntu2.20) over (7.68.0-1ubuntu2.6) ...
Selecting previously unselected package curl.
Preparing to unpack .../curl_7.68.0-1ubuntu2.20_amd64.deb ...
Unpacking curl (7.68.0-1ubuntu2.20) ...
```

103.42.193.130 is the public address of the network

```
kiran@kiran-VirtualBox:~$ curl icanhazip.com
103.42.193.130
kiran@kiran-VirtualBox:~$
```

Spoof the Ip address with curl tool using following command

torsocks curl icanhazip.com

```
┌──(amey㉿kali)-[~]
└─$ torsocks curl icanhazip.com                                                 127 ×
1697536118 PERROR torsocks[40077]: socks5 libc connect: Connection refused (in socks5_con
nect() at socks5.c:202)
curl: (6) Could not resolve host: icanhazip.com
```

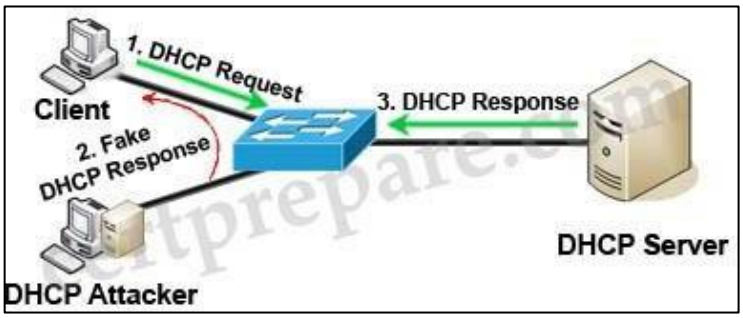
Spoofed IP address:

```
kiran@kiran-VirtualBox:~$ service tor start
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to start 'tor.service'.
Authenticating as: kiran,,, (kiran)
Password:
==== AUTHENTICATION COMPLETE ====
kiran@kiran-VirtualBox:~$ torsocks curl icanhazip.com
23.137.249.185
kiran@kiran-VirtualBox:~$
```

Here the spoofed ip address is 23.137.249.185

2) DHCP Spoofing:

DHCP spoofing occurs when an attacker attempts to respond to DHCP requests and trying to list themselves (spoofs) as the default gateway or DNS server, hence, initiating a man in the middle attack.



With that, it is possible that they can intercept traffic from users before forwarding to the real gateway or perform DoS by flooding the real DHCP server with request to choke ip address resources.

Checking victim IP address:

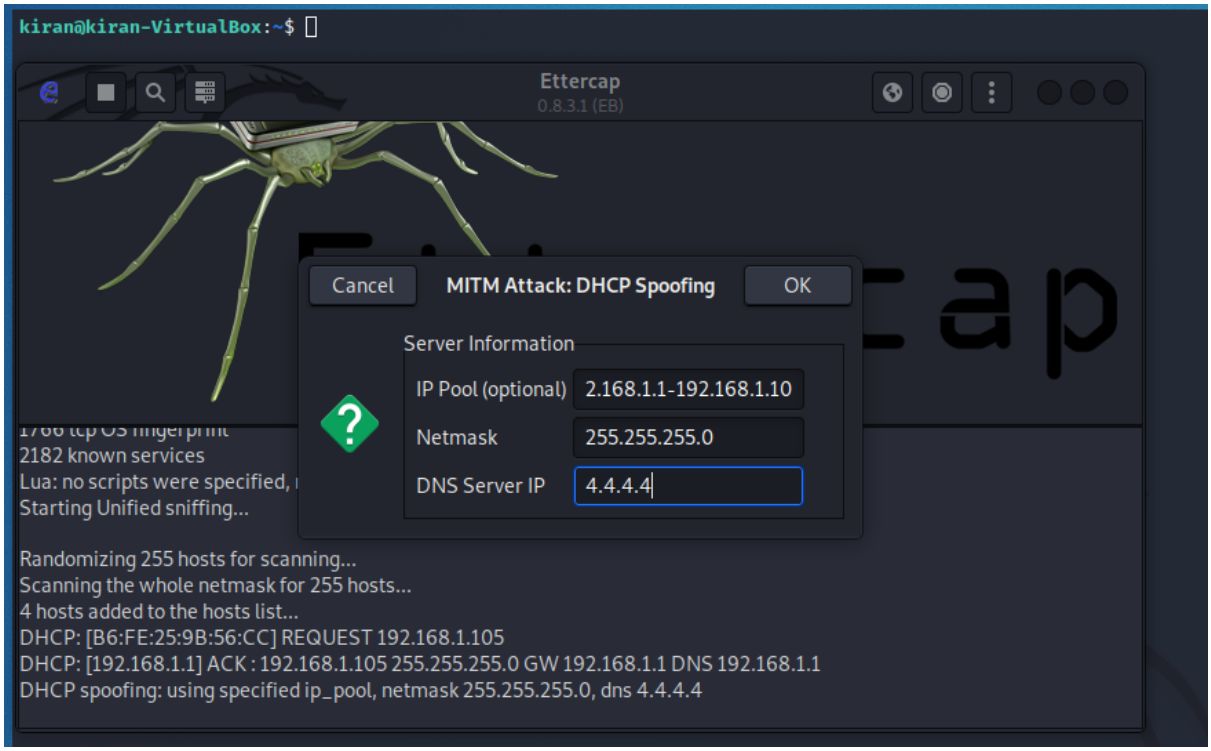
```
kiran@kiran-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.110  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::e296:4a3b:855a:2317  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:11:9a:c7  txqueuelen 1000  (Ethernet)
        RX packets 26292  bytes 19527439 (19.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4878  bytes 1384371 (1.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 459  bytes 38479 (38.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 459  bytes 38479 (38.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

kiran@kiran-VirtualBox:~$
```

To spoof the DHCP server we are using the ettercap tool in the kali linux. In the MITM attacks we chose the DHCP spoofing option.

We provided the custom IP Pool, Netmask and custom DNS Server IP.

IP Pool: 192.168.1.1-192.168.1.10

Netmask: 255.255.255.0 DNS Server IP: 4.4.4.4



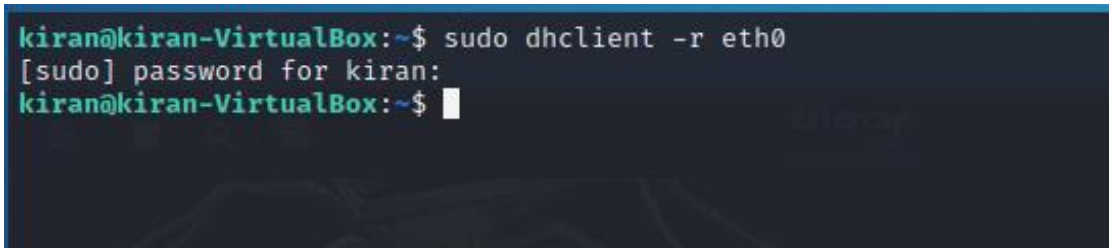
To renew the IP address on the victim machine we ran the following commands:

To reset the IP for eth0 port:

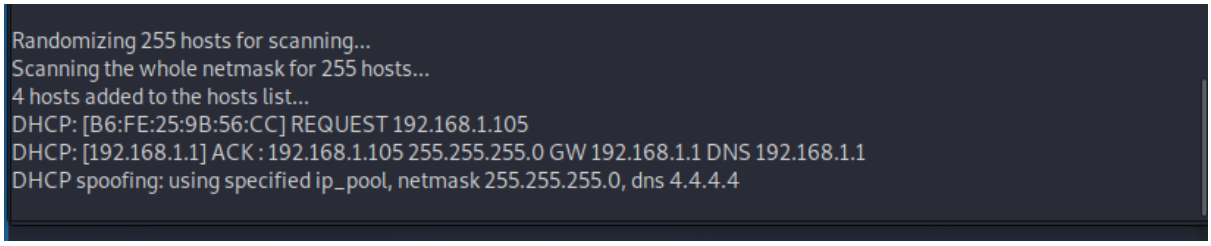
sudo dhclient -r eth0

To assign new IP from DHCP server:

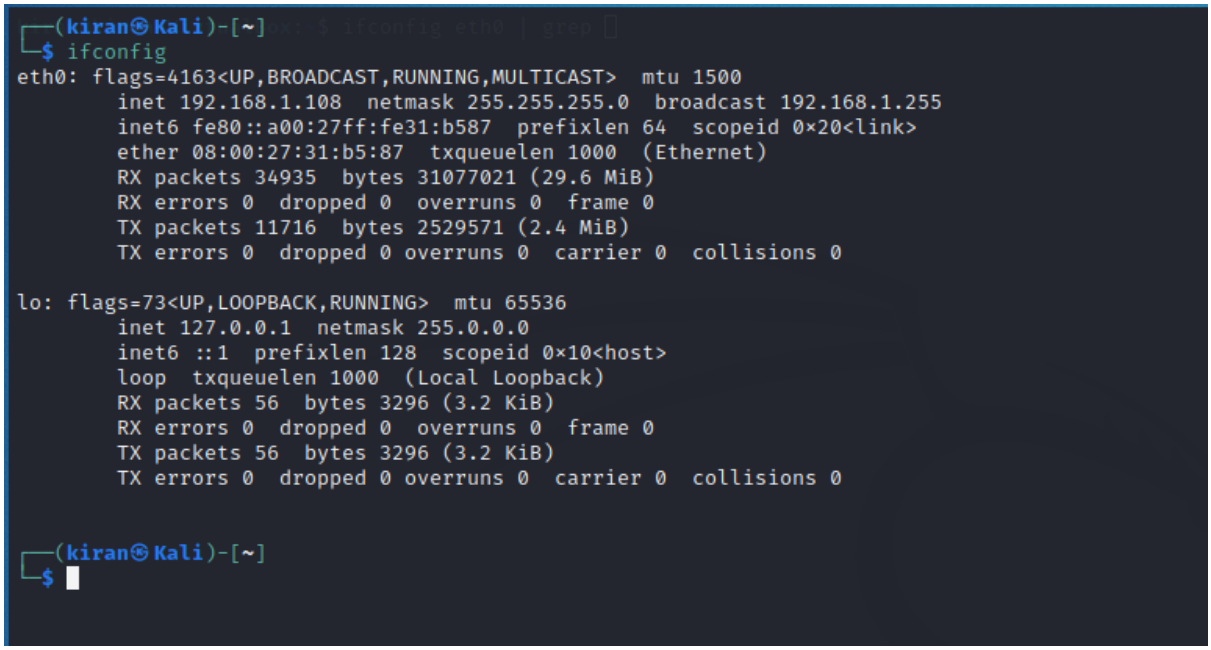
sudo dhclient eth0



In the ettercap window we can see the new IP request from the victim and fake IP being assigned to the victimmachine from the ettercap.



Now when we run the ifconfig command on the victim machine we can see the new IP assigned to the victim.



4. Network Interface Layer

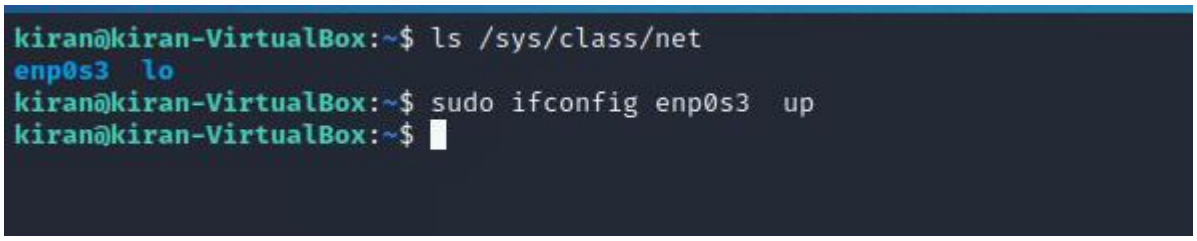
1) MAC Spoofing:

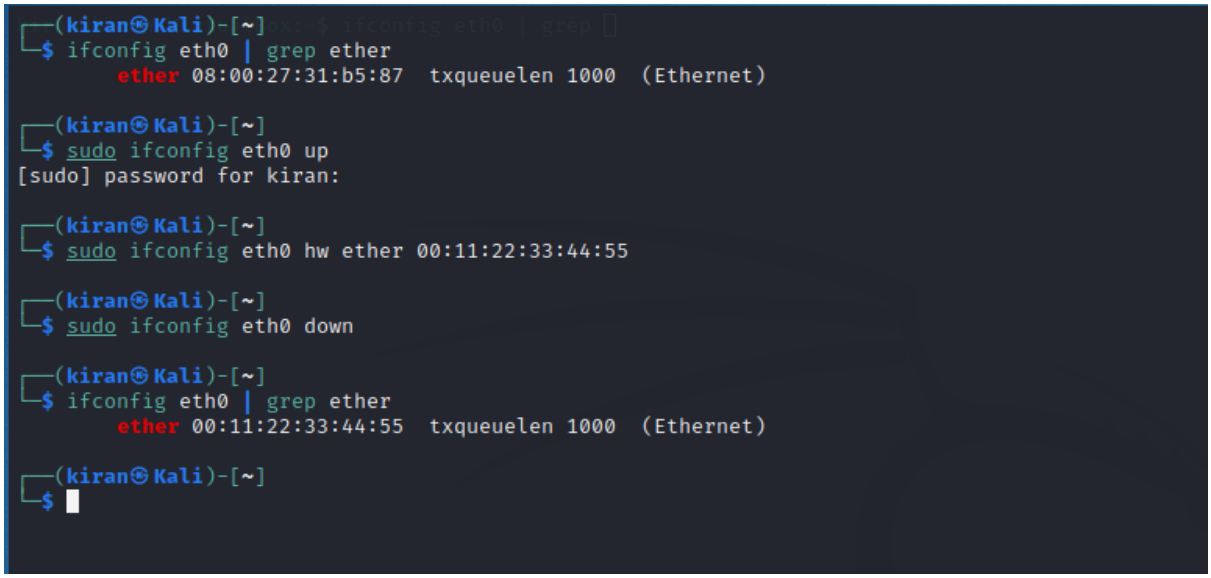
MAC spoofing attack is a common phenomenon currently, thanks to the ever-growing technology. But first, we need to understand what a MAC spoofing attack is in order to prevent ourselves from falling victims to it.

MAC address spoofing attack is where the impostor or hacker hunts the network for valid and original MAC addresses and circumvents access control measures, giving the hacker the advantage to pose as one of the valid MAC addresses.
MAC address spoofing is which type of attack wherein the hacker is also able to bypass authentication checks as he presents this as the default gateway and copies all of the data passed on to the default gateway without being identified, giving him all the important details about applications in use and end-host IP addresses.

To perform MAC spoofing using a manual method, the following commands can be given. Only the root user has the permission to do so.

- ifconfig eth0 | grep ether
- sudo ifconfig eth0 up
- sudo ifconfig eth0 hw ether <MAC address of choice>
- sudo ifconfig eth0 down

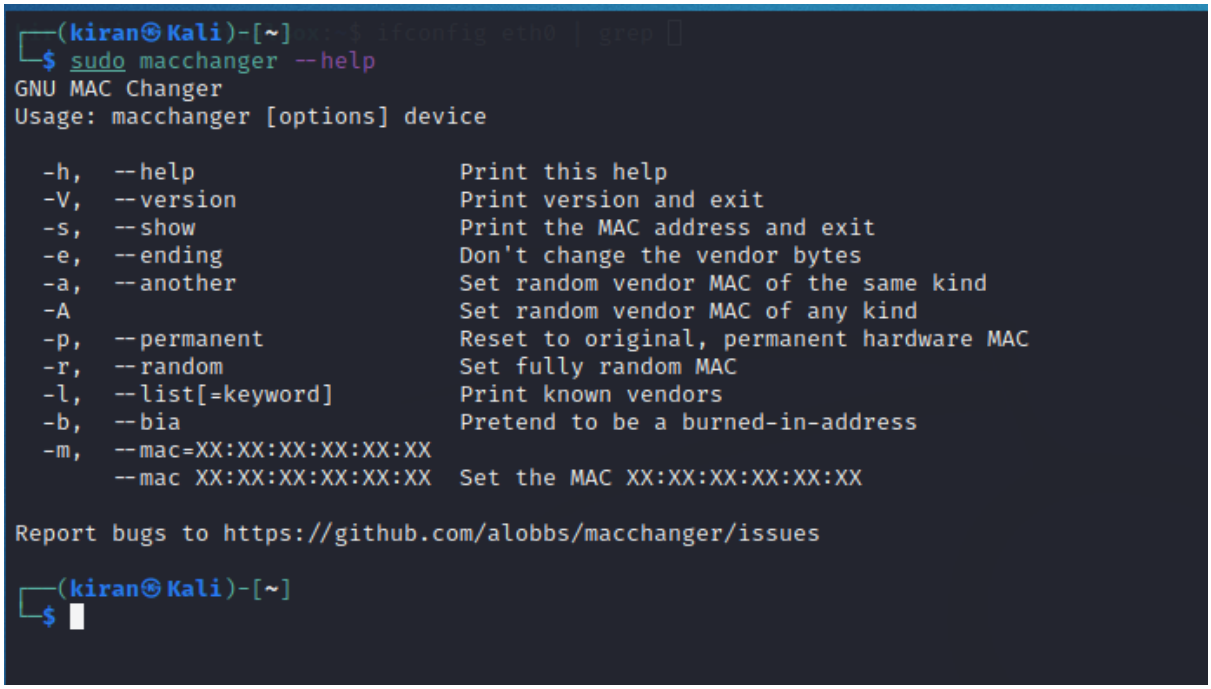


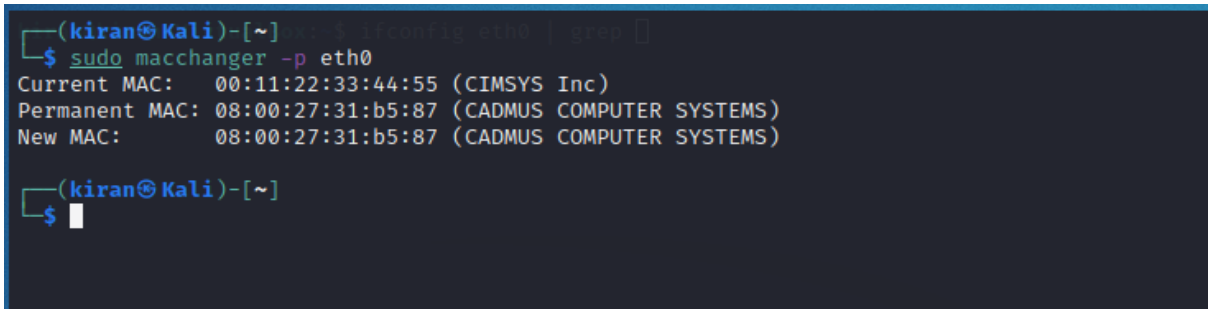


Thus, the MAC address of the choice has been assigned. It is assigned as “00:11:22:33:44:55” here.

Use of macchanger:

- macchanger --help
- To reset original MAC address,
- sudo macchanger-p eth0





The macchanger tool can be used to allocate another MAC address as the current MAC. It has two options to do so:

-A : Set random vendor MAC of any kind

-r : Set fully random MAC

- sudo macchanger -A eth0

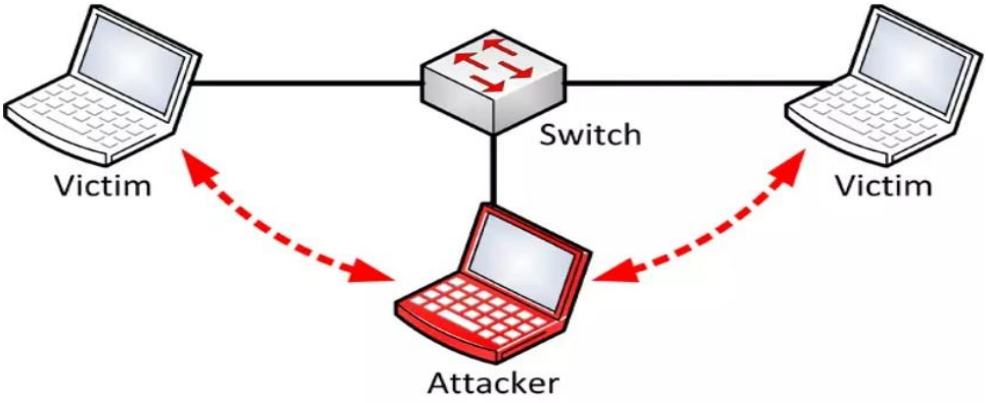
```
┌──(kiran㉿Kali)-[~]
└─$ sudo macchanger -s eth0
Current MAC:   08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)
Permanent MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)

┌──(kiran㉿Kali)-[~]
└─$ sudo macchanger -A eth0
Current MAC:   08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)
Permanent MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)
New MAC:       1c:c1:de:11:b8:19 (Hewlett-Packard Company)

┌──(kiran㉿Kali)-[~]
└─$ sudo macchanger -s eth0
Current MAC:   1c:c1:de:11:b8:19 (Hewlett-Packard Company)
Permanent MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)

┌──(kiran㉿Kali)-[~]
└─$
```

- sudo macchanger -r eth0

```
┌──(kiran㉿Kali)-[~]
└─$ sudo macchanger -s eth0
Current MAC:   1c:c1:de:11:b8:19 (Hewlett-Packard Company)
Permanent MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)

┌──(kiran㉿Kali)-[~]
└─$ sudo macchanger -r eth0
Current MAC:   1c:c1:de:11:b8:19 (Hewlett-Packard Company)
Permanent MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)
New MAC:       0a:64:e1:df:e9:d6 (unknown)

┌──(kiran㉿Kali)-[~]
└─$ sudo macchanger -s eth0
Current MAC:   0a:64:e1:df:e9:d6 (unknown)
Permanent MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)

┌──(kiran㉿Kali)-[~]
└─$
```

2) ARP Spoofing:

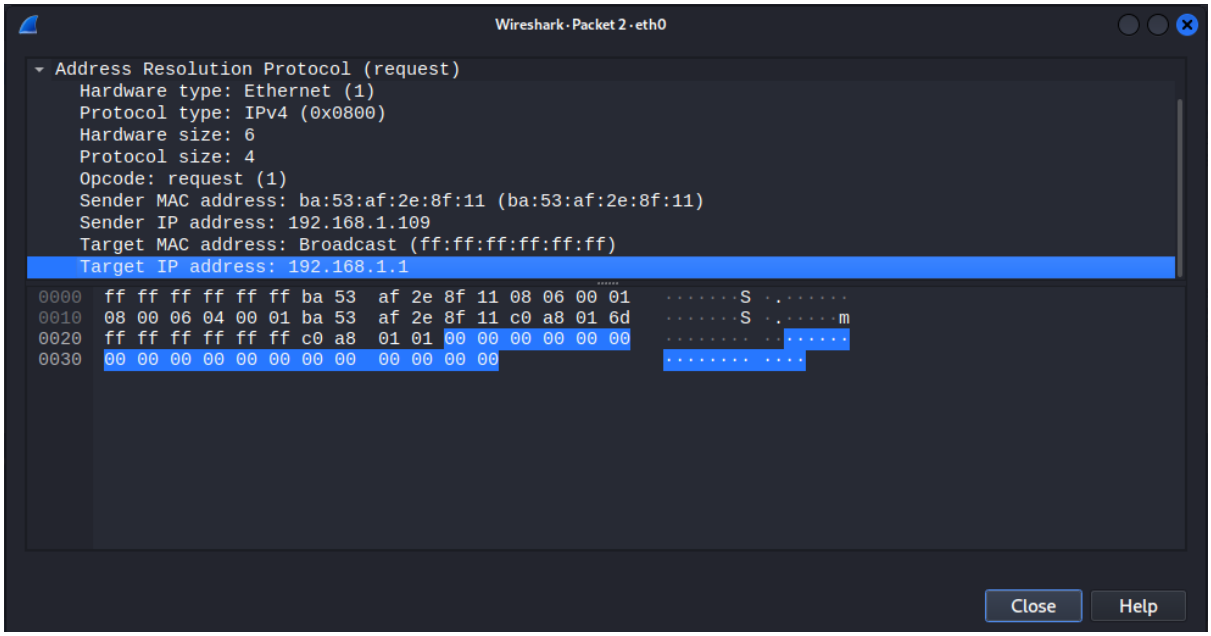
An attack where a hacker impersonates the MAC address of another device on a local network. That results in the linking of an attacker's MAC address with the IP address of a legitimate computer or server on the network.

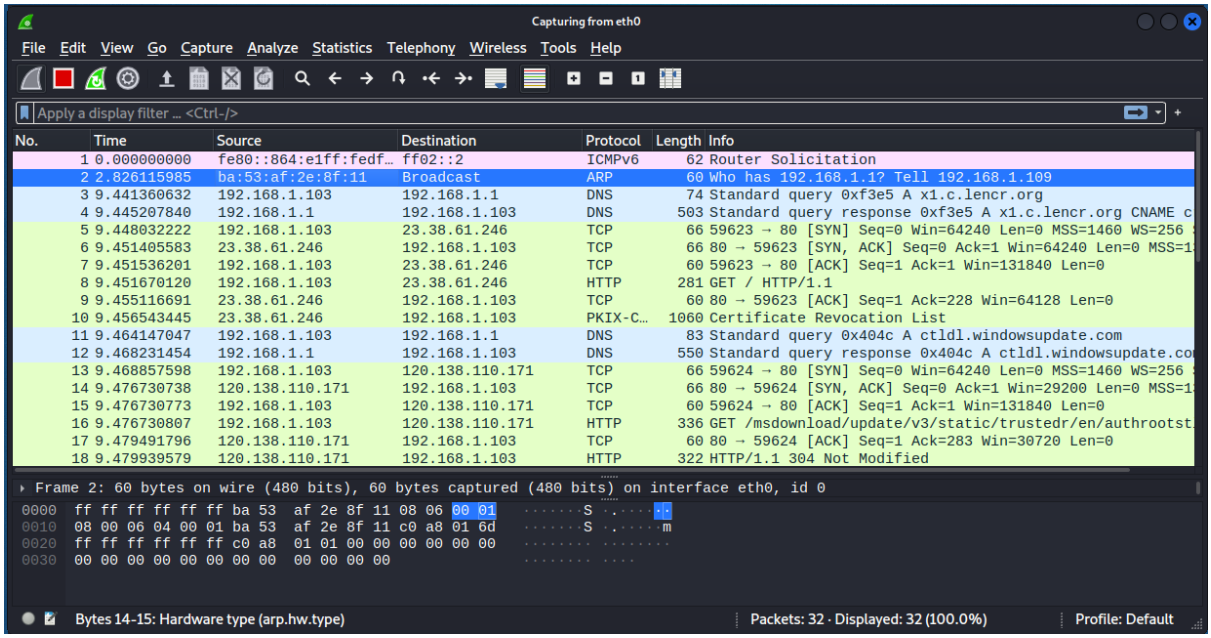



```
┌──(kiran㉿Kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.101  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::864:e1ff:fedf:e9d6  prefixlen 64  scopeid 0x20<link>
        ether 0a:64:e1:df:e9:d6  txqueuelen 1000  (Ethernet)
        RX packets 34993  bytes 31082091 (29.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11791  bytes 2535137 (2.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 77  bytes 4997 (4.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 77  bytes 4997 (4.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(kiran㉿Kali)-[~]
└─$
```

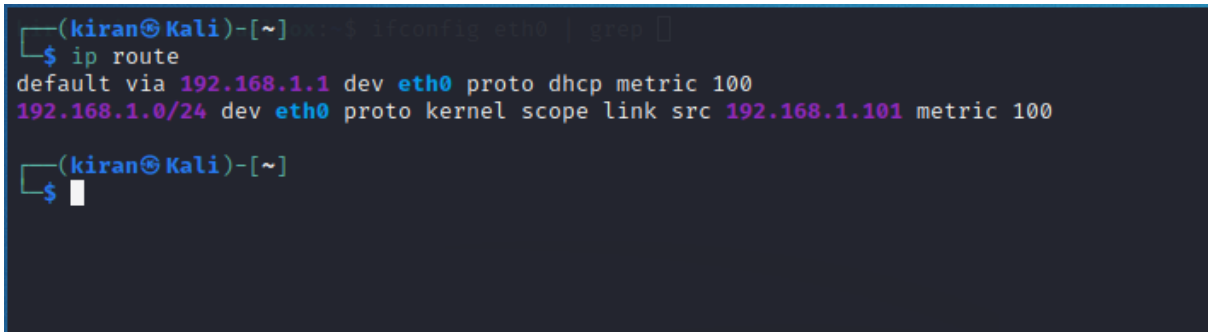
```
┌──(kiran㉿Kali)-[~]
└─$ arp -a
┌──(kiran㉿Kali)-[~]
└─$ sudo arp spoof
```

Arp packet in wireshark

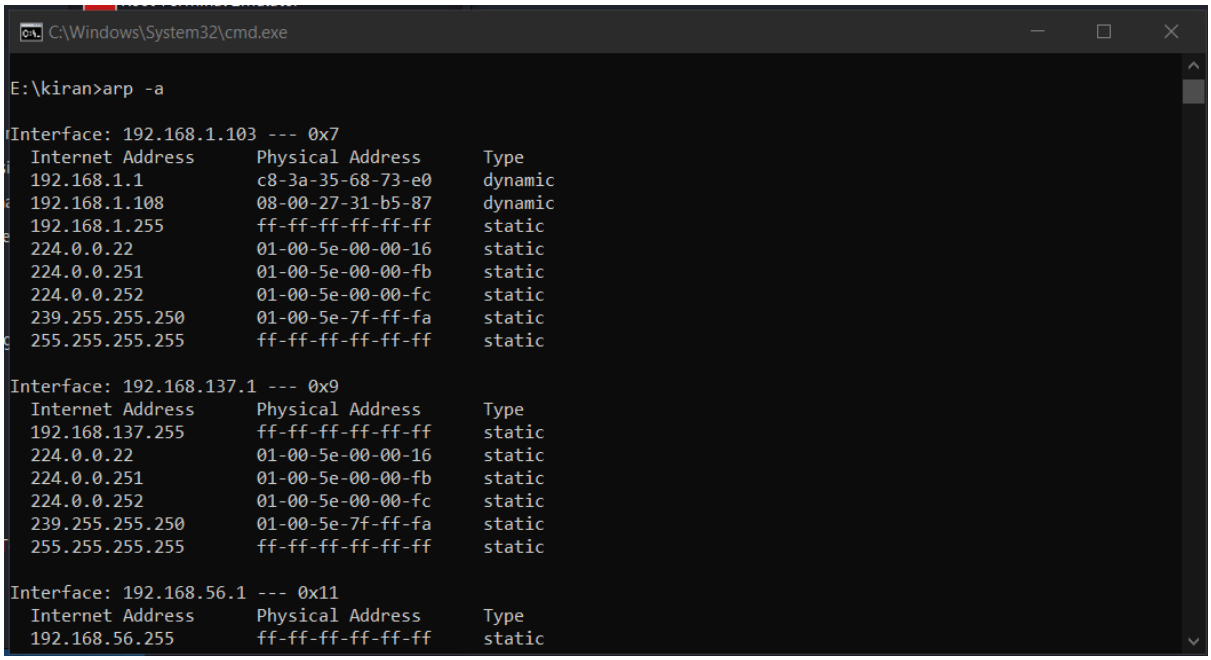




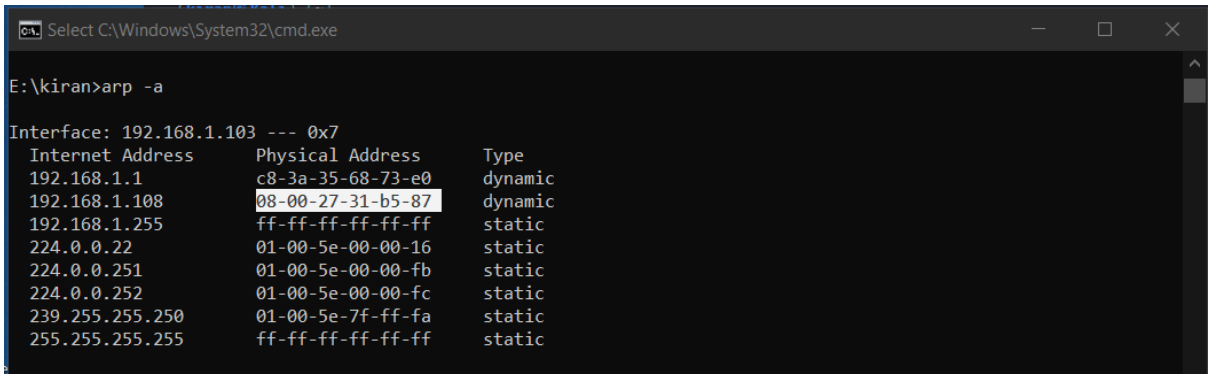
Check the route of the network



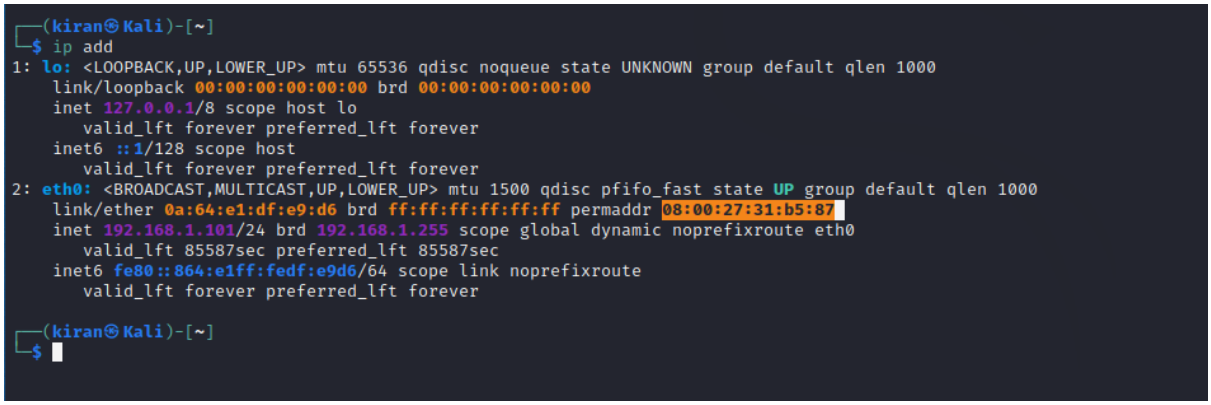
Check the arp -a the network interfaces



Here the the victims ip address is 192.168.1.108 and the corresponding mac address is 08-00-27-31-b5-87

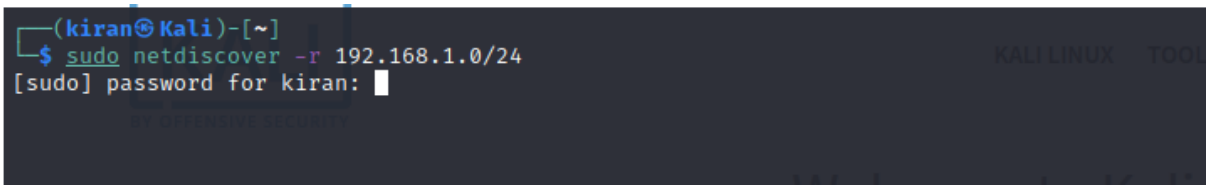


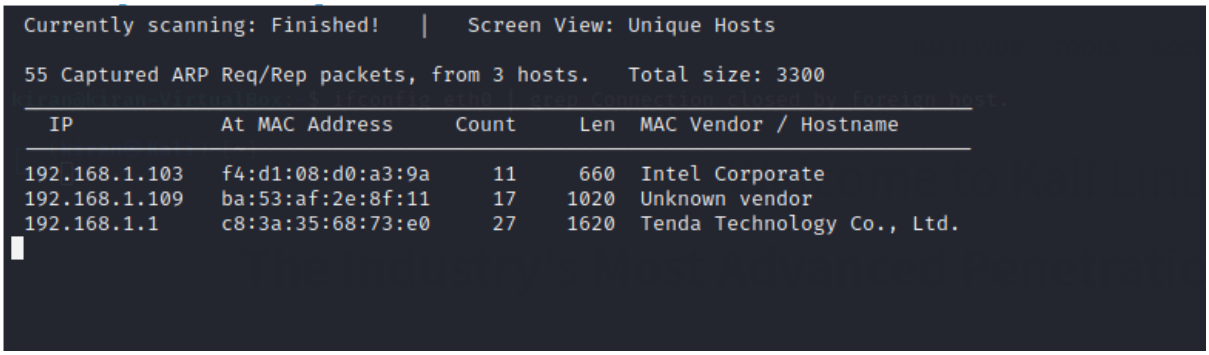
The ip address and MAC address in virtual machine of the victim machine



Check the devices present in the network

sudo netdiscover -r 192.168.1.0/24





Conclusion: Hence, we have studied and performed several attacks from each layer of TCP/IP.