Name: Adithya M	SRN: PES1UG20CS621	Section: K
	Date:19/1/2022	Week Number: 2

Computer Networks

Week 1	Study and understand the basic networking tools - Wireshark, Tcpdump, Ping, Traceroute and Netcat.

Learn and understand Network Tools

1. Wireshark
   - Perform and analyze Ping PDU capture
   - Examine HTTP packet capture
   - Analyze HTTP packet capture using filter
2. Netcat
   - Establish communication between client and server
   - Transfer files
3. Tcpdump
   - Capture packets
4. Ping
   - Test the connectivity between two systems
5. Traceroute
   - Perform tranceroute checks
6. Nmap
   - Explore an entire network

Task 1: Linux interface Configuration (ifconfig / IP command)

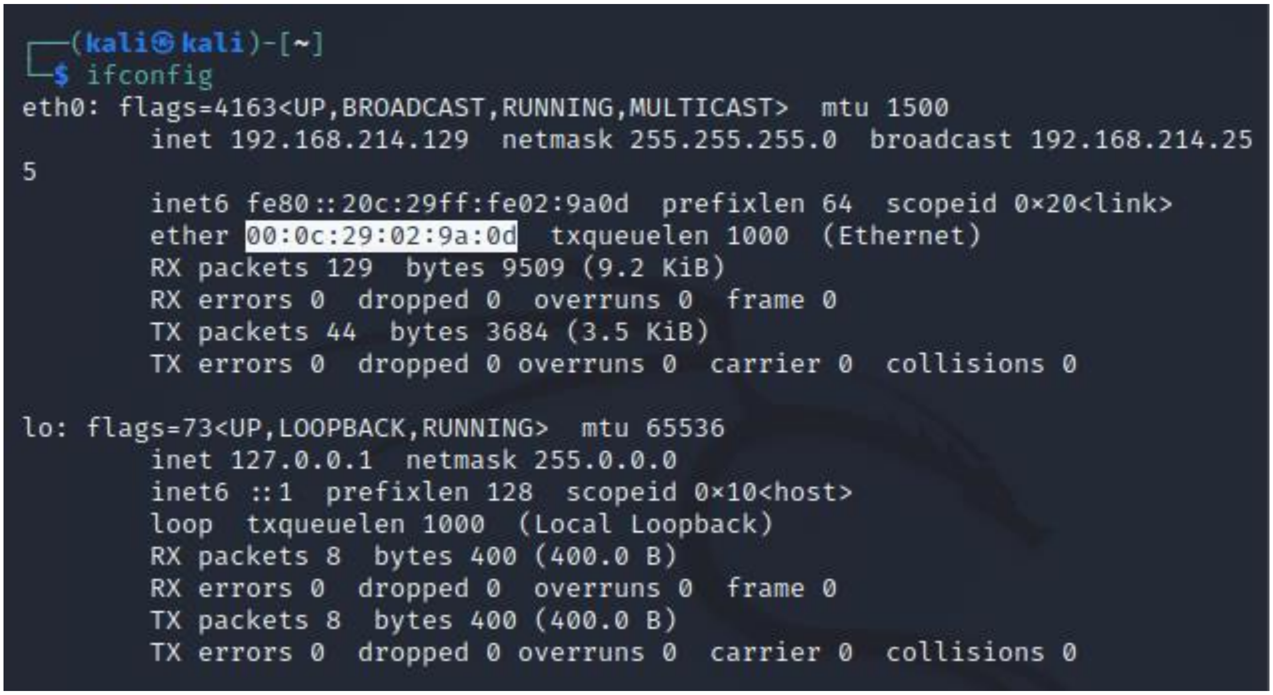
Step 1: To display status of all active network interfaces.

Ifconfig (or) ip addr show

Analyze and fill the following table:

Ip address table:

Interface name	IP address (IPv4 / IPv6)	MAC address
eth0	**192.168.214.129**	**00:0c:29:02:9a:0d**
lo	**127.0.0.1**	**NA**



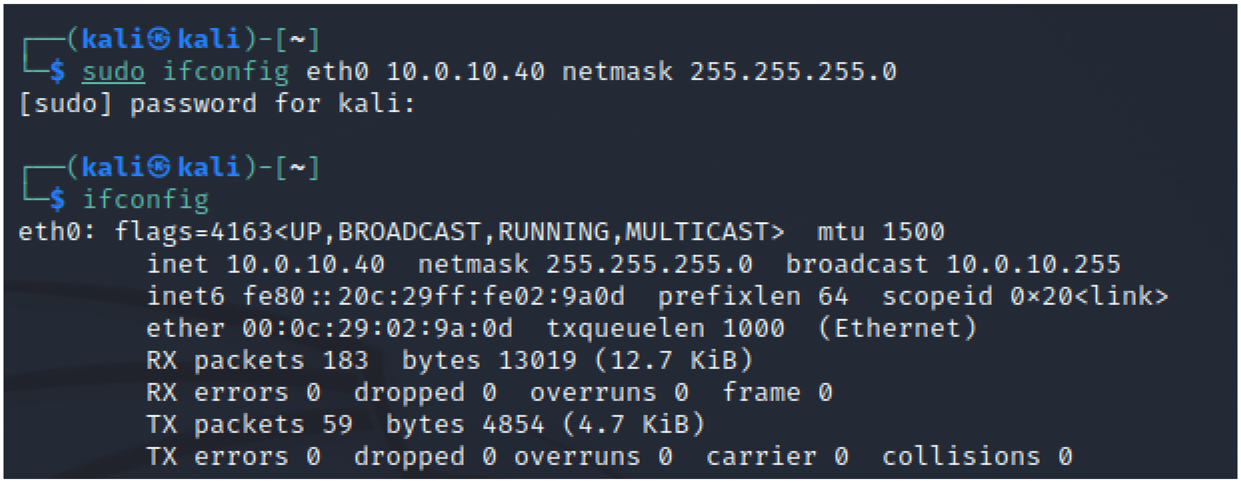
```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.214.129  netmask 255.255.255.0  broadcast 192.168.214.25
5
        inet6 fe80::20c:29ff:fe02:9a0d  prefixlen 64  scopeid 0×20<link>
        ether 00:0c:29:02:9a:0d  txqueuelen 1000  (Ethernet)
        RX packets 129  bytes 9509 (9.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 44  bytes 3684 (3.5 KiB)
        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 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 400 (400.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 400 (400.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The hardware address and the IP address is mentioned, when **ifconfig** is typed in the terminal.

Step 2: To assign an IP address to an interface, use the following command.

sudo ifconfig interface_name 10.0.your_section.your_sno netmask 255.255.255.0 (or)

sudo ip addr add 10.0.your_section.your_sno /24 dev interface_name



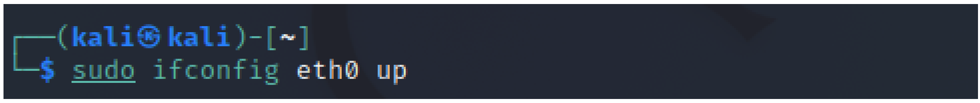
```
┌──(kali㉿kali)-[~]
└─$ sudo ifconfig eth0 10.0.10.40 netmask 255.255.255.0
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.10.40  netmask 255.255.255.0  broadcast 10.0.10.255
        inet6 fe80::20c:29ff:fe02:9a0d  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:02:9a:0d  txqueuelen 1000  (Ethernet)
        RX packets 183  bytes 13019 (12.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 59  bytes 4854 (4.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

10.0.10.40 is assigned as the **IP address** to the interface.

Step 3: To activate / deactivate a network interface, type.

sudo ifconfig interface_name down

sudo ifconfig interface_name up



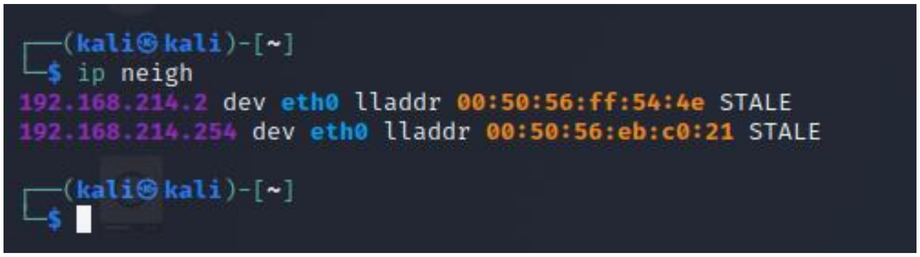
```
┌──(kali㉿kali)-[~]
└─$ sudo ifconfig eth0 up
```

ip

The configured interface is set to up and
running if it isn’t.

Step 4: To show the current neighbour table in kernel, type

ip neigh



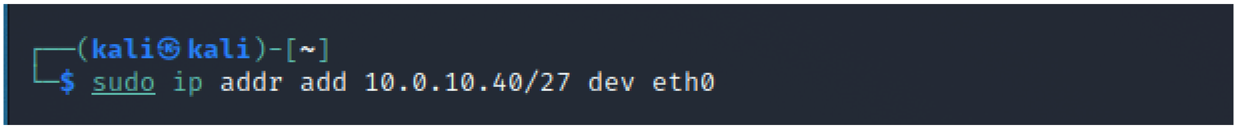
```
┌──(kali㉿kali)-[~]
└─$ ip neigh
192.168.214.2 dev eth0 lladdr 00:50:56:ff:54:4e STALE
192.168.214.254 dev eth0 lladdr 00:50:56:eb:c0:21 STALE

┌──(kali㉿kali)-[~]
└─$ ▮
```

The neighbour table is shown in the output.

Task 2: Ping PDU (Packet Data Units or Packets) Capture

Step 1: Assign an IP address to the system (Host).

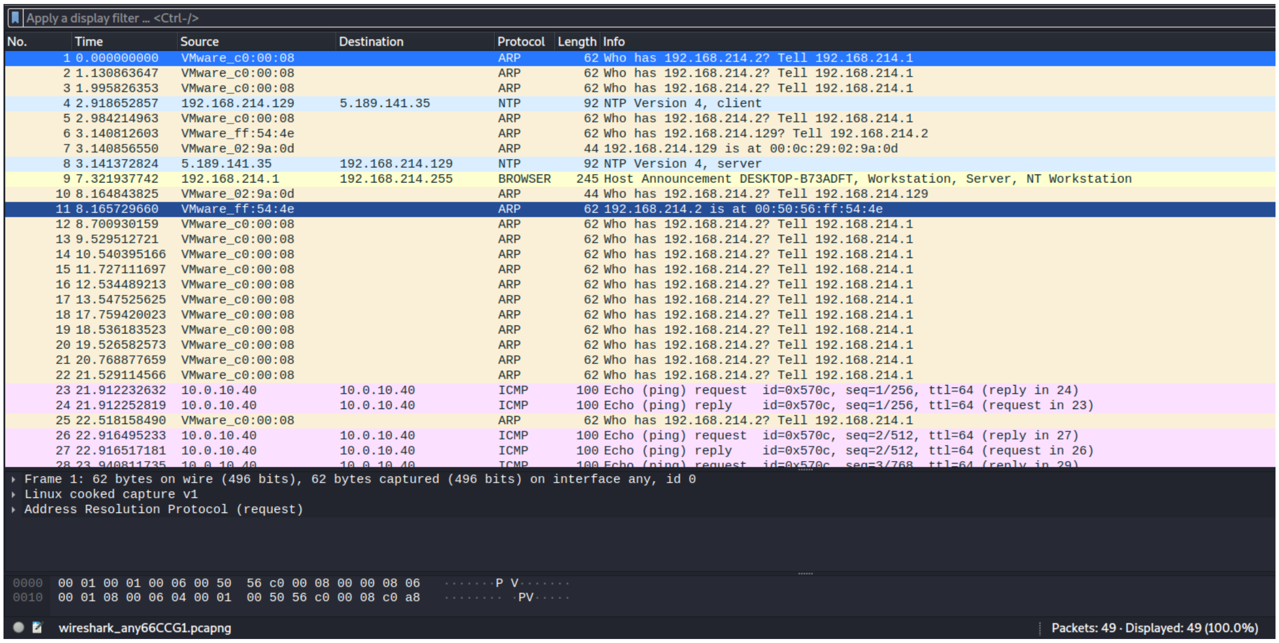
Note: IP address of your system should be **10.0.your_section.your_sno.**



```
┌──(kali㉿kali)-[~]
└─$ sudo ip addr add 10.0.10.40/27 dev eth0
```

The IP address is set to **10.0.10.40.**

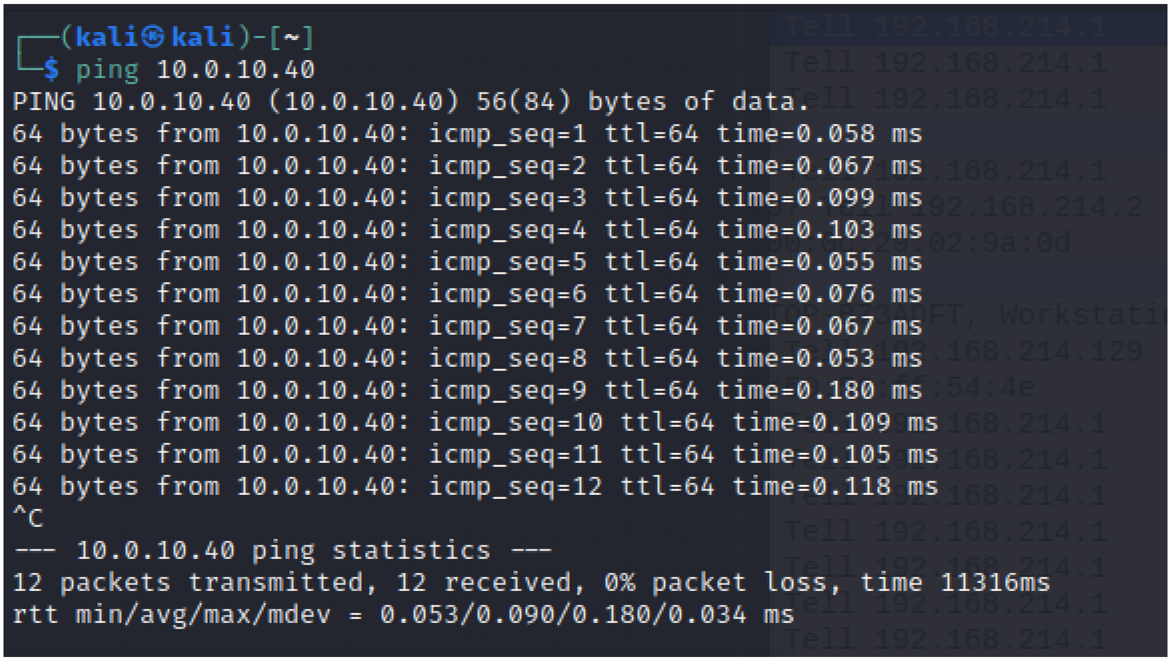
Step 2: Launch Wireshark and select '**any**' interface



```
No.   Time           Source             Destination        Protocol  Length Info
    1 0.000000000    VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
    2 1.130863647    VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
    3 1.995826353    VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
    4 2.918652857    192.168.214.129    5.189.141.35       NTP         92 NTP Version 4, client
    5 2.984214963    VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
    6 3.140812603    VMware_ff:54:4e                       ARP         62 Who has 192.168.214.129? Tell 192.168.214.2
    7 3.140856550    VMware_02:9a:0d                       ARP         44 192.168.214.129 is at 00:0c:29:02:9a:0d
    8 3.141372824    5.189.141.35       192.168.214.129    NTP         92 NTP Version 4, server
    9 7.321937742    192.168.214.1      192.168.214.255    BROWSER    245 Host Announcement DESKTOP-B73ADFT, Workstation, Server, NT Workstation
   10 8.164843825    VMware_02:9a:0d                       ARP         44 Who has 192.168.214.2? Tell 192.168.214.129
   11 8.165729660    VMware_ff:54:4e                       ARP         62 192.168.214.2 is at 00:50:56:ff:54:4e
   12 8.700930159    VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   13 9.529512721    VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   14 10.540395166   VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   15 11.727111697   VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   16 12.534489213   VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   17 13.547525625   VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   18 17.759420023   VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   19 18.536183523   VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   20 19.526582573   VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   21 20.768877659   VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   22 21.529114566   VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   23 21.912232632   10.0.10.40         10.0.10.40         ICMP       100 Echo (ping) request  id=0x570c, seq=1/256, ttl=64 (reply in 24)
   24 21.912252819   10.0.10.40         10.0.10.40         ICMP       100 Echo (ping) reply    id=0x570c, seq=1/256, ttl=64 (request in 23)
   25 22.518158490   VMware_c0:00:08                       ARP         62 Who has 192.168.214.2? Tell 192.168.214.1
   26 22.916495233   10.0.10.40         10.0.10.40         ICMP       100 Echo (ping) request  id=0x570c, seq=2/512, ttl=64 (reply in 27)
   27 22.916517181   10.0.10.40         10.0.10.40         ICMP       100 Echo (ping) reply    id=0x570c, seq=2/512, ttl=64 (request in 26)
   28 23.940811735   10.0.10.40         10.0.10.40         ICMP       100 Echo (ping) request  id=0x570c, seq=3/768, ttl=64 (reply in 29)

› Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface any, id 0
› Linux cooked capture v1
› Address Resolution Protocol (request)

0000  00 01 00 01 00 06 00 50  56 c0 00 08 00 00 08 06   ·······P V·······
0010  00 01 08 00 06 04 00 01  00 50 56 c0 00 08 c0 a8   ········ ·PV·····

wireshark_any66CCG1.pcapng                                Packets: 49 · Displayed: 49 (100.0%)
```

Wireshark on launch and opened into "any".

Step 3: In terminal, type ping **10.0.your_section.your_sno**



```
┌──(kali㉿kali)-[~]
└─$ ping 10.0.10.40
PING 10.0.10.40 (10.0.10.40) 56(84) bytes of data.
64 bytes from 10.0.10.40: icmp_seq=1 ttl=64 time=0.058 ms
64 bytes from 10.0.10.40: icmp_seq=2 ttl=64 time=0.067 ms
64 bytes from 10.0.10.40: icmp_seq=3 ttl=64 time=0.099 ms
64 bytes from 10.0.10.40: icmp_seq=4 ttl=64 time=0.103 ms
64 bytes from 10.0.10.40: icmp_seq=5 ttl=64 time=0.055 ms
64 bytes from 10.0.10.40: icmp_seq=6 ttl=64 time=0.076 ms
64 bytes from 10.0.10.40: icmp_seq=7 ttl=64 time=0.067 ms
64 bytes from 10.0.10.40: icmp_seq=8 ttl=64 time=0.053 ms
64 bytes from 10.0.10.40: icmp_seq=9 ttl=64 time=0.180 ms
64 bytes from 10.0.10.40: icmp_seq=10 ttl=64 time=0.109 ms
64 bytes from 10.0.10.40: icmp_seq=11 ttl=64 time=0.105 ms
64 bytes from 10.0.10.40: icmp_seq=12 ttl=64 time=0.118 ms
^C
--- 10.0.10.40 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11316ms
rtt min/avg/max/mdev = 0.053/0.090/0.180/0.034 ms
```

Observations to be made

Step 4: Analyze the following in Terminal

- TTL
- Protocol used by ping
- Time

The TTL is **64.**

The protocol used by ping is **ICMP.**

The time taken is **0.090 ms** on average.

Step 5: Analyze the following in Wireshark

On Packet List Pane, select the first echo packet on the list. On Packet Details Pane, click on each of the four "+" to expand the information. Analyze the frames with the first echo request and echo reply and complete the table below.

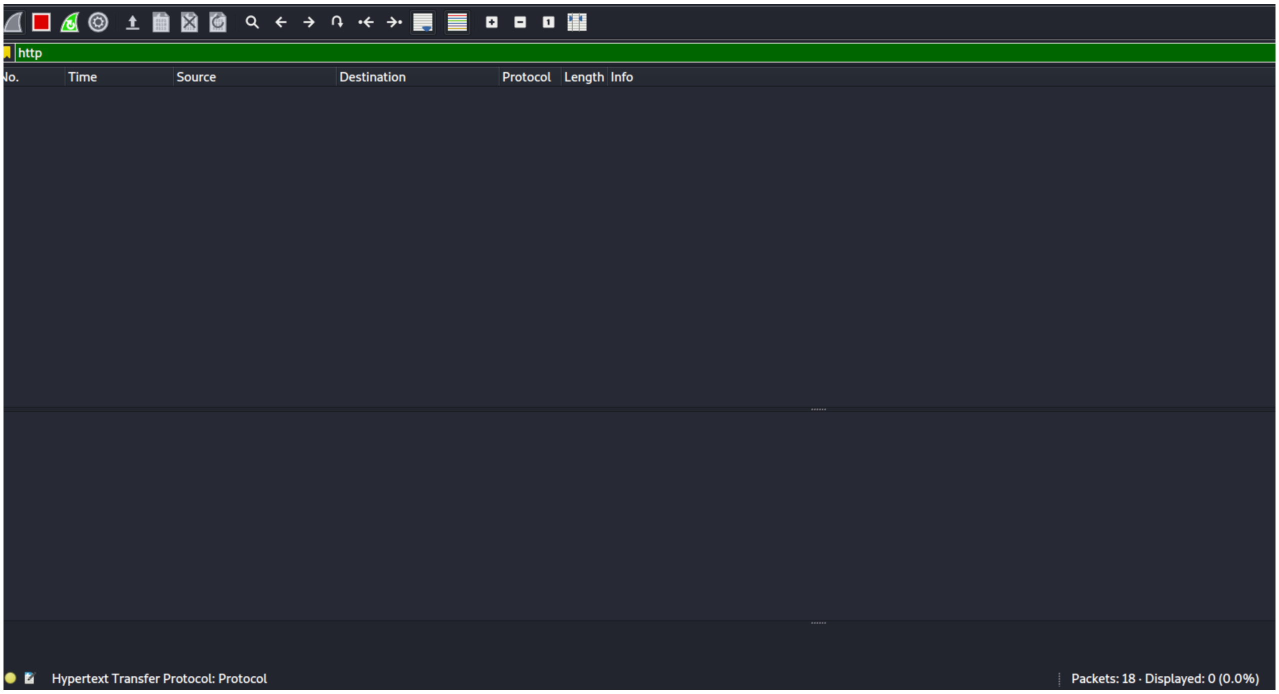
Details	First echo request	First echo reply

Frame Number	23	24
Source IP address	10.0.10.40	10.0.10.40
Destination IP address	10.0.10.49	10.0.10.40
ICMP Type Value	8	0
ICMP Code Value	0	0
Source Ethernet Address	00:00:00:00:00:00	00:00:00:00:00:00
Destination Ethernet Address	00:00:00:00:00:00	00:00:00:00:00:00
Internet Protocol Version	4	4
Time To Live (TTL) Value	64	64

Task 3: HTTP PDU Capture

Using Wireshark's Filter feature

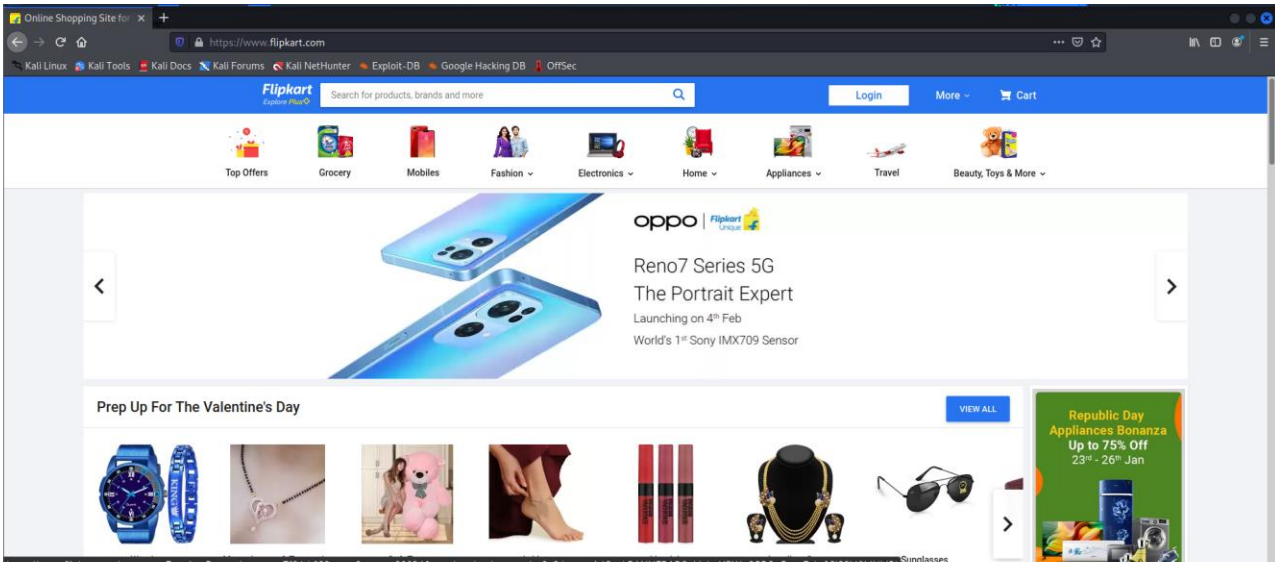
Step 1: Launch Wireshark and select 'any' interface. On the Filter toolbar, type-in 'http' and press enter

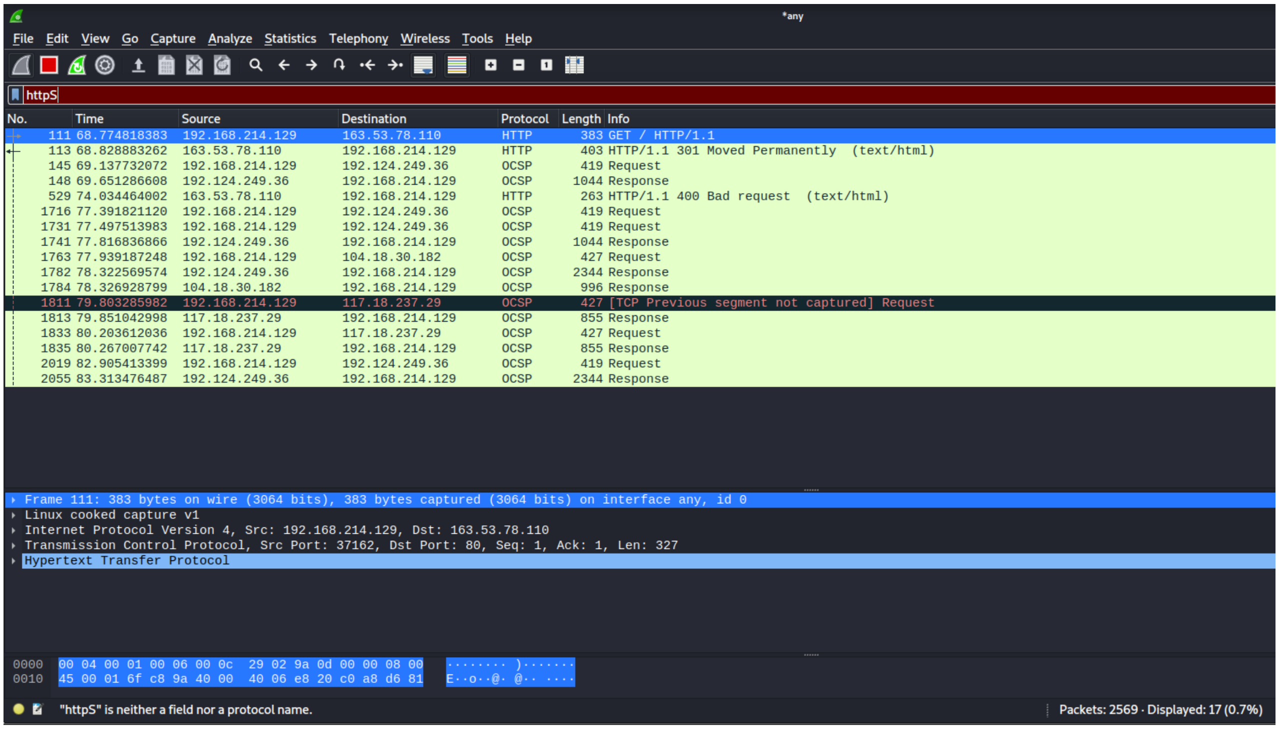


Hypertext Transfer Protocol: Protocol

Packets: 18 · Displayed: 0 (0.0%)

Step 2: Open Firefox browser, and browse www.flipkart.com





```
httpS
```

```
No.     Time          Source           Destination      Protocol Length Info
    111 68.774818383  192.168.214.129  163.53.78.110    HTTP      383 GET / HTTP/1.1
    113 68.828883262  163.53.78.110    192.168.214.129  HTTP      403 HTTP/1.1 301 Moved Permanently  (text/html)
    145 69.137732072  192.168.214.129  192.124.249.36   OCSP      419 Request
    148 69.651286608  192.124.249.36   192.168.214.129  OCSP     1044 Response
    529 74.034464002  163.53.78.110    192.168.214.129  HTTP      263 HTTP/1.1 400 Bad request  (text/html)
   1716 77.391821120  192.168.214.129  192.124.249.36   OCSP      419 Request
   1731 77.497513983  192.168.214.129  192.124.249.36   OCSP      419 Request
   1741 77.816836866  192.124.249.36   192.168.214.129  OCSP     1044 Response
   1763 77.939187248  192.168.214.129  104.18.30.182    OCSP      427 Request
   1782 78.322569574  192.124.249.36   192.168.214.129  OCSP     2344 Response
   1784 78.326928799  104.18.30.182    192.168.214.129  OCSP      996 Response
   1811 79.803285982  192.168.214.129  117.18.237.29    OCSP      427 [TCP Previous segment not captured] Request
   1813 79.851042998  117.18.237.29    192.168.214.129  OCSP      855 Response
   1833 80.203612036  192.168.214.129  117.18.237.29    OCSP      427 Request
   1835 80.267007742  117.18.237.29    192.168.214.129  OCSP      855 Response
   2019 82.905413399  192.168.214.129  192.124.249.36   OCSP      419 Request
   2055 83.313476487  192.124.249.36   192.168.214.129  OCSP     2344 Response
```

```
› Frame 111: 383 bytes on wire (3064 bits), 383 bytes captured (3064 bits) on interface any, id 0
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 192.168.214.129, Dst: 163.53.78.110
› Transmission Control Protocol, Src Port: 37162, Dst Port: 80, Seq: 1, Ack: 1, Len: 327
› Hypertext Transfer Protocol
```

```
0000  00 04 00 01 00 06 00 0c  29 02 9a 0d 00 00 08 00   ········ )·······
0010  45 00 01 6f c8 9a 40 00  40 06 e8 20 c0 a8 d6 81   E··o··@· @·· ····
```

"httpS" is neither a field nor a protocol name.

Packets: 2569 · Displayed: 17 (0.7%)

Observations to be made

Step 3: Analyze the first (interaction of host to the web server) and second frame (response of server to the client). By analyzing the filtered frames, complete the table below:

Details	First echo request	First echo reply

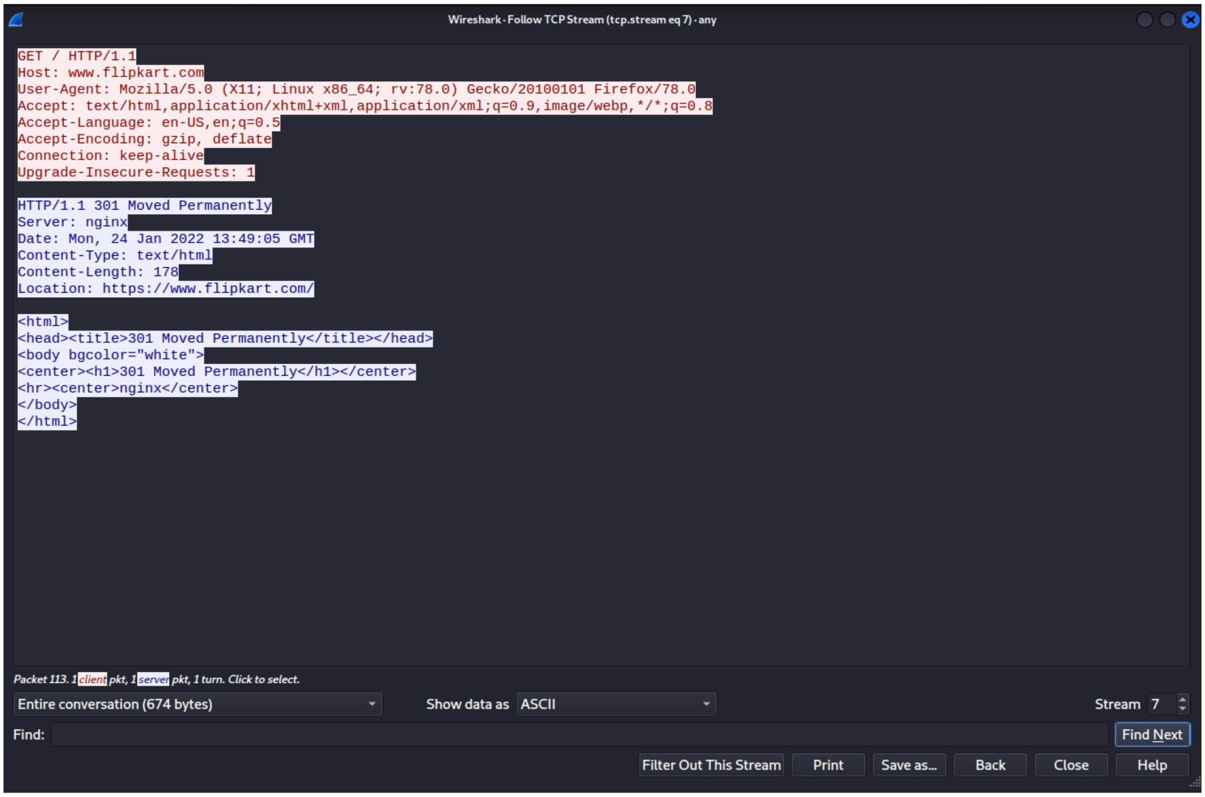
Frame Number	111	113
Source port	37162	80
Destination port	80	37162
Source IP address	192.168.214.129	163.53.78.110
Destination IP address	163.53.78.110	192.168.214.129
Source Ethernet Address	00:00:00:00:00:00	00:00:00:00:00:00
Destination Ethernet Address	00:00:00:00:00:00	00:00:00:00:00:00

Step 4: Analyze the HTTP request and response and complete the table below.

HTTP request		**HTTP response**	
GET	/ HTTP/1.1	**Server**	nginx
Host	www.flipkart.com	**Content-type**	text/html
User-agent	Mozilla/5.0	**Date**	Mon 24,Jan 2022 13:47:06 GMT
Accept Language	en-us,en	**Location**	https://www.flipkart.com
Accept-Encoding	gzip,deflate	**Content-Length**	178
connection	keep-alive	**connection**	keep-alive

Using Wireshark’s Follow TCP Stream

Step 1: Make sure the filter is blank. Right-click any packet inside the Packet List Pane, then select ‘Follow TCP Stream’. For demo purpose, a packet containing the HTTP GET request “GET / HTTP / 1.1” can be selected.
Step 2: Upon following a TCP stream, screenshot the whole window.

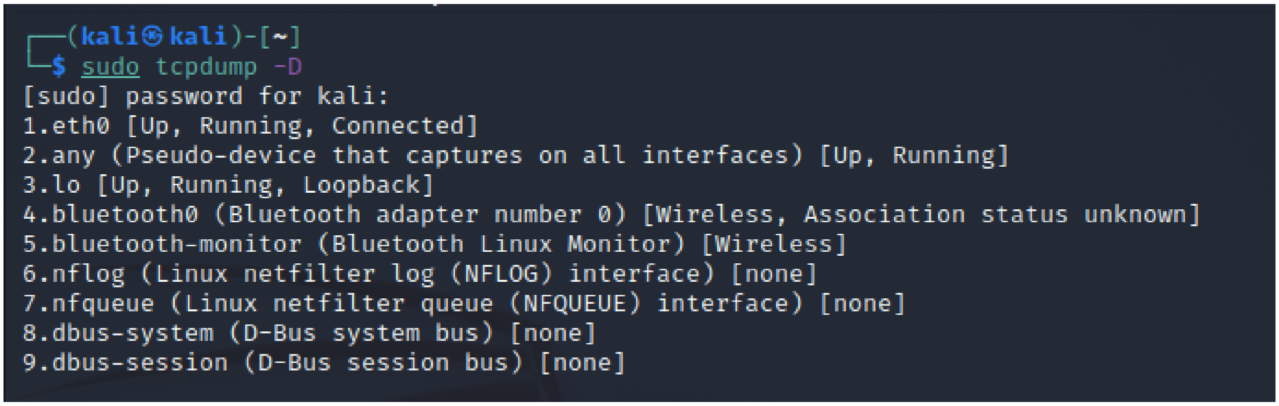


```
GET / HTTP/1.1
Host: www.flipkart.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Mon, 24 Jan 2022 13:49:05 GMT
Content-Type: text/html
Content-Length: 178
Location: https://www.flipkart.com/

<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Packet 113. 1 client pkt, 1 server pkt, 1 turn. Click to select.

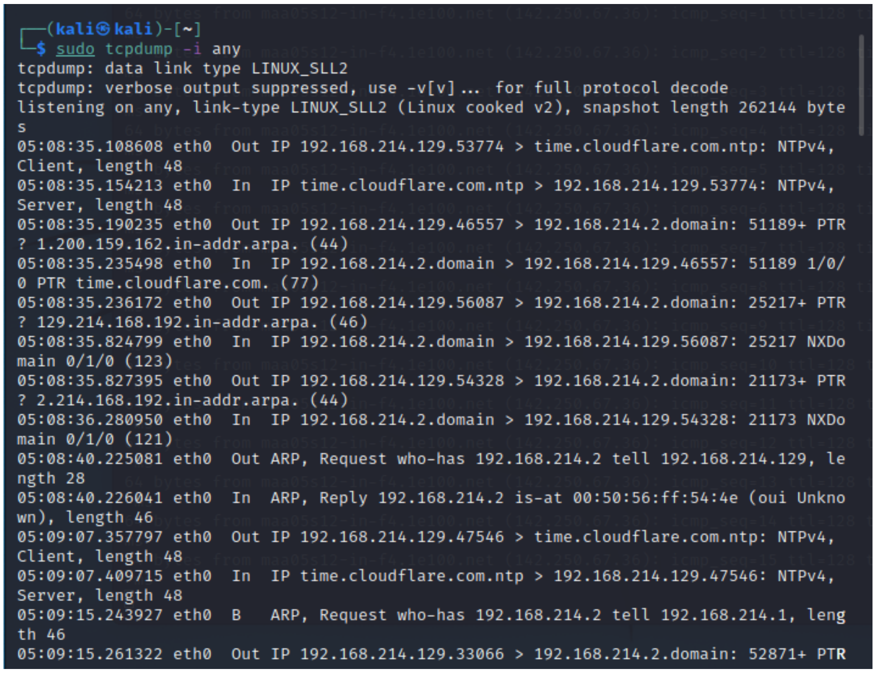
Task 4: Capturing packets with tcpdump

Step 1: Use the command **tcpdump -D** to see which interfaces are available for capture.
sudo tcpdump –D

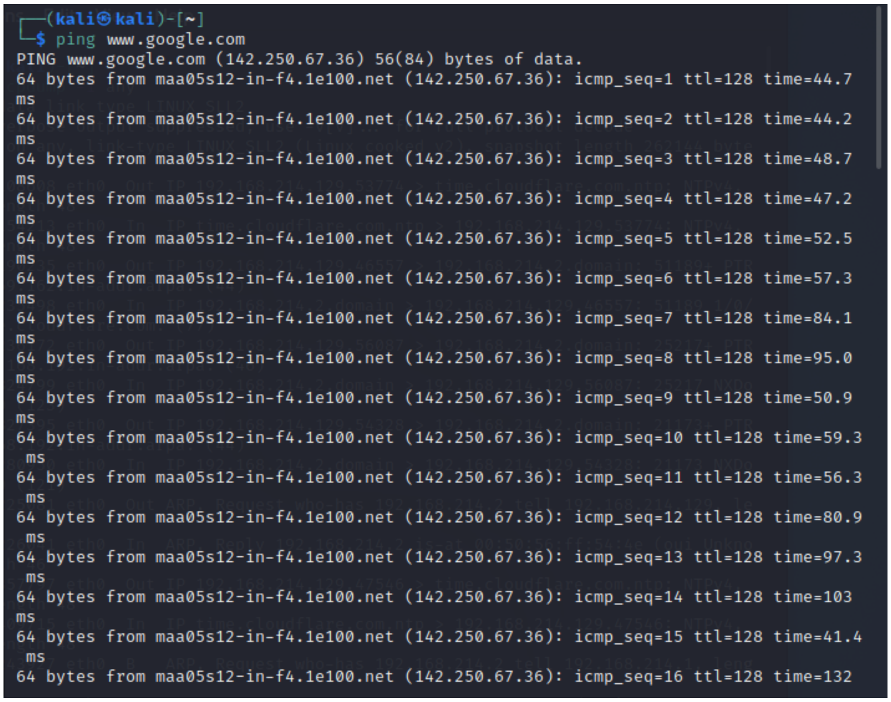


```
┌──(kali㉿kali)-[~]
└─$ sudo tcpdump -D
[sudo] password for kali:
1.eth0 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth0 (Bluetooth adapter number 0) [Wireless, Association status unknown]
5.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
6.nflog (Linux netfilter log (NFLOG) interface) [none]
7.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
8.dbus-system (D-Bus system bus) [none]
9.dbus-session (D-Bus session bus) [none]
```

Step 2: Capture all packets in any interface by running this command:
sudo tcpdump -i any



```
┌──(kali㉿kali)-[~]
└─$ sudo tcpdump -i any
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 byte
s
05:08:35.108608 eth0  Out IP 192.168.214.129.53774 > time.cloudflare.com.ntp: NTPv4,
Client, length 48
05:08:35.154213 eth0  In  IP time.cloudflare.com.ntp > 192.168.214.129.53774: NTPv4,
Server, length 48
05:08:35.190235 eth0  Out IP 192.168.214.129.46557 > 192.168.214.2.domain: 51189+ PTR
? 1.200.159.162.in-addr.arpa. (44)
05:08:35.235498 eth0  In  IP 192.168.214.2.domain > 192.168.214.129.46557: 51189 1/0/
0 PTR time.cloudflare.com. (77)
05:08:35.236172 eth0  Out IP 192.168.214.129.56087 > 192.168.214.2.domain: 25217+ PTR
? 129.214.168.192.in-addr.arpa. (46)
05:08:35.824799 eth0  In  IP 192.168.214.2.domain > 192.168.214.129.56087: 25217 NXDo
main 0/1/0 (123)
05:08:35.827395 eth0  Out IP 192.168.214.129.54328 > 192.168.214.2.domain: 21173+ PTR
? 2.214.168.192.in-addr.arpa. (44)
05:08:36.280950 eth0  In  IP 192.168.214.2.domain > 192.168.214.129.54328: 21173 NXDo
main 0/1/0 (121)
05:08:40.225081 eth0  Out ARP, Request who-has 192.168.214.2 tell 192.168.214.129, le
ngth 28
05:08:40.226041 eth0  In  ARP, Reply 192.168.214.2 is-at 00:50:56:ff:54:4e (oui Unkno
wn), length 46
05:09:07.357797 eth0  Out IP 192.168.214.129.47546 > time.cloudflare.com.ntp: NTPv4,
Client, length 48
05:09:07.409715 eth0  In  IP time.cloudflare.com.ntp > 192.168.214.129.47546: NTPv4,
Server, length 48
05:09:15.243927 eth0  B   ARP, Request who-has 192.168.214.2 tell 192.168.214.1, leng
th 46
05:09:15.261322 eth0  Out IP 192.168.214.129.33066 > 192.168.214.2.domain: 52871+ PTR
```



```
┌──(kali㉿kali)-[~]
└─$ ping www.google.com
PING www.google.com (142.250.67.36) 56(84) bytes of data.
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=1 ttl=128 time=44.7
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=2 ttl=128 time=44.2
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=3 ttl=128 time=48.7
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=4 ttl=128 time=47.2
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=5 ttl=128 time=52.5
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=6 ttl=128 time=57.3
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=7 ttl=128 time=84.1
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=8 ttl=128 time=95.0
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=9 ttl=128 time=50.9
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=10 ttl=128 time=59.3
 ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=11 ttl=128 time=56.3
 ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=12 ttl=128 time=80.9
 ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=13 ttl=128 time=97.3
 ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=14 ttl=128 time=103
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=15 ttl=128 time=41.4
 ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=16 ttl=128 time=132
```

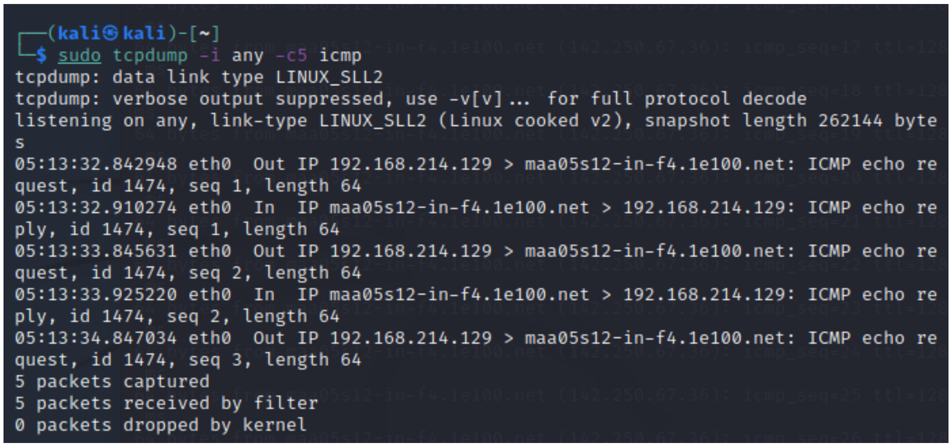
Note: Perform some pinging operation while giving above command. Also type www.google.com in browser.

Observation

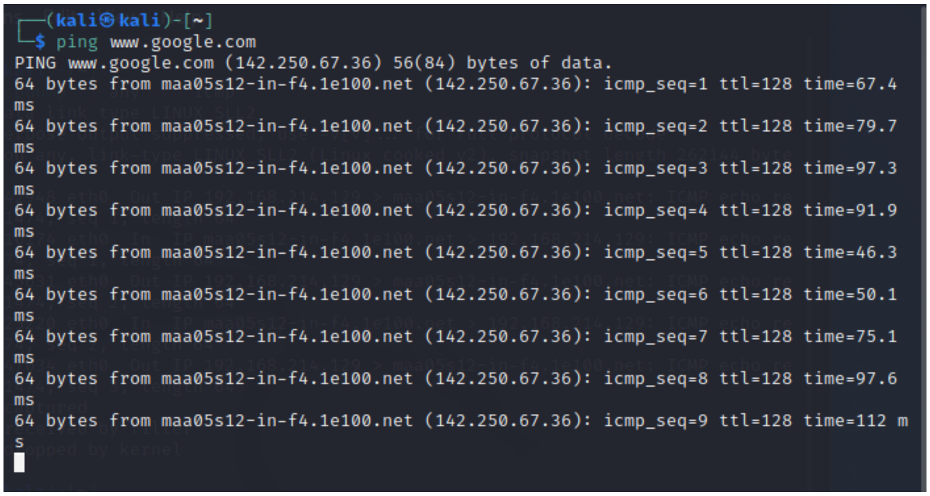
Step 3: Understand the output format.

The above command is used to capture all the packets from all the interfaces.
ICMP, UDP and TCP are the main packets that are visible in the above screenshot.
The timestamp followed by the link level headers, then by ARP/RARP packets if any,
Then by IPv4 packets if any, followed by TCP packets. The sequence numbers and the length finish defining the outputs.

Step 4: To filter packets based on protocol, specifying the protocol in the command line. For example, capture ICMP packets only by using this command:
sudo tcpdump -i any -c5 icmp



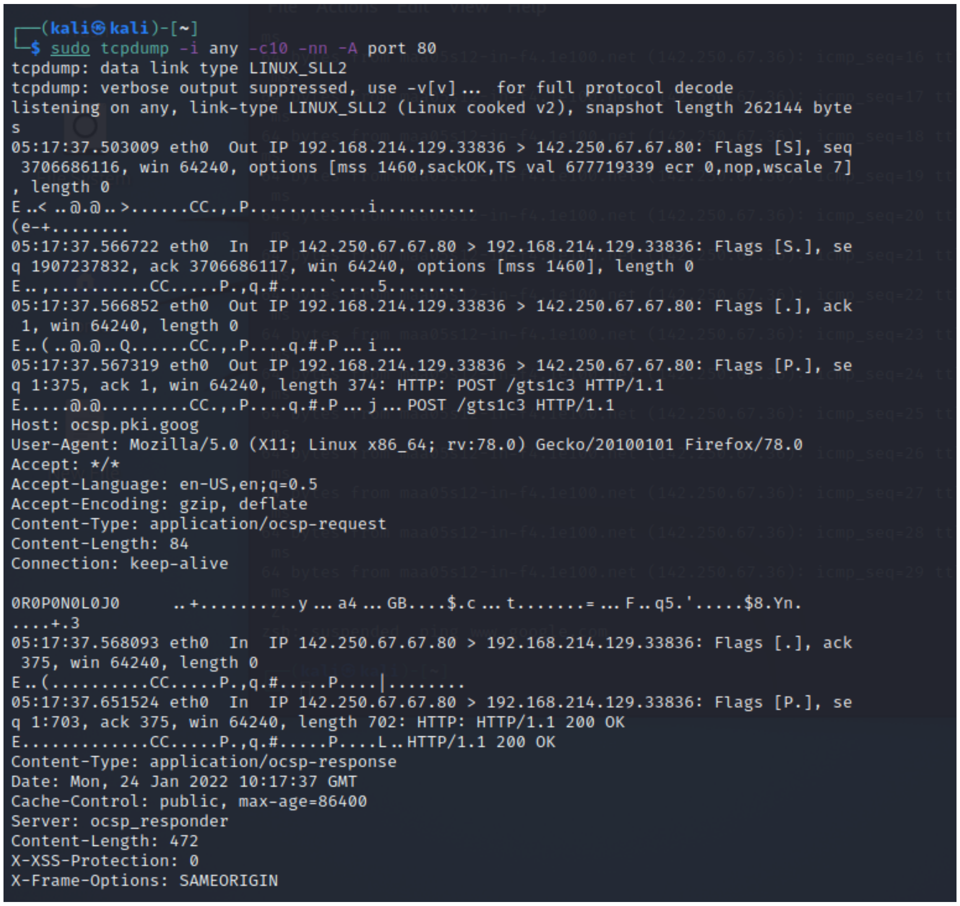
```
┌──(kali㉿kali)-[~]
└─$ sudo tcpdump -i any -c5 icmp
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 byte
s
05:13:32.842948 eth0  Out IP 192.168.214.129 > maa05s12-in-f4.1e100.net: ICMP echo re
quest, id 1474, seq 1, length 64
05:13:32.910274 eth0  In  IP maa05s12-in-f4.1e100.net > 192.168.214.129: ICMP echo re
ply, id 1474, seq 1, length 64
05:13:33.845631 eth0  Out IP 192.168.214.129 > maa05s12-in-f4.1e100.net: ICMP echo re
quest, id 1474, seq 2, length 64
05:13:33.925220 eth0  In  IP maa05s12-in-f4.1e100.net > 192.168.214.129: ICMP echo re
ply, id 1474, seq 2, length 64
05:13:34.847034 eth0  Out IP 192.168.214.129 > maa05s12-in-f4.1e100.net: ICMP echo re
quest, id 1474, seq 3, length 64
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```



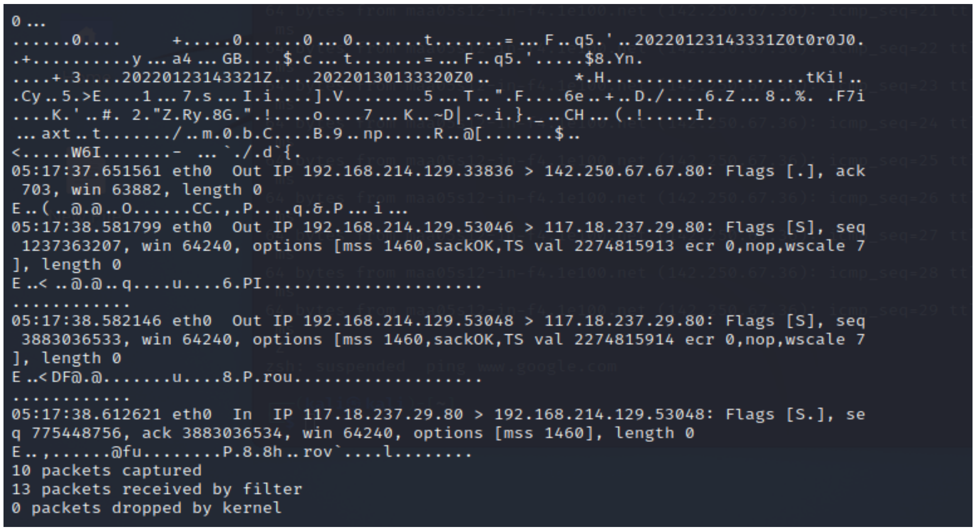
```
┌──(kali㉿kali)-[~]
└─$ ping www.google.com
PING www.google.com (142.250.67.36) 56(84) bytes of data.
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=1 ttl=128 time=67.4
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=2 ttl=128 time=79.7
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=3 ttl=128 time=97.3
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=4 ttl=128 time=91.9
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=5 ttl=128 time=46.3
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=6 ttl=128 time=50.1
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=7 ttl=128 time=75.1
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=8 ttl=128 time=97.6
ms
64 bytes from maa05s12-in-f4.1e100.net (142.250.67.36): icmp_seq=9 ttl=128 time=112 m
s
```

Step 5: Check the packet content. For example, inspect the HTTP content of a web request like this:
sudo tcpdump -i any -c10 -nn -A port 80

On trying to access the Gmail account sign-in website.



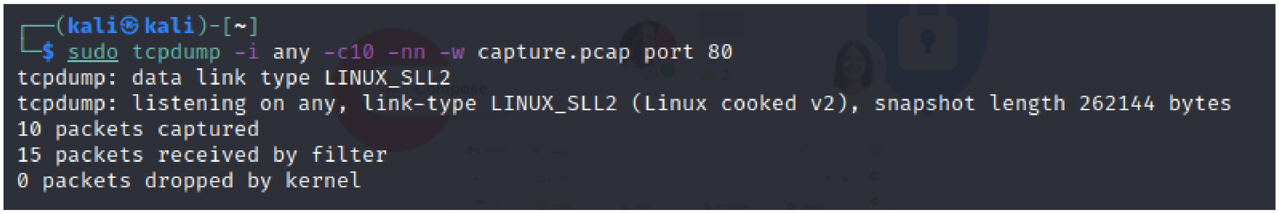
```
┌──(kali㉿kali)-[~]
└─$ sudo tcpdump -i any -c10 -nn -A port 80
tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 byte
s
05:17:37.503009 eth0  Out IP 192.168.214.129.33836 > 142.250.67.67.80: Flags [S], seq
 3706686116, win 64240, options [mss 1460,sackOK,TS val 677719339 ecr 0,nop,wscale 7]
, length 0
E..<..@.@..>......CC.,.P............i..........
(e-+........
05:17:37.566722 eth0  In  IP 142.250.67.67.80 > 192.168.214.129.33836: Flags [S.], se
q 1907237832, ack 3706686117, win 64240, options [mss 1460], length 0
E..,..........CC.....P.,q.#.....`....5........
05:17:37.566852 eth0  Out IP 192.168.214.129.33836 > 142.250.67.67.80: Flags [.], ack
 1, win 64240, length 0
E..(..@.@..Q......CC.,.P....q.#.P...i...
05:17:37.567319 eth0  Out IP 192.168.214.129.33836 > 142.250.67.67.80: Flags [P.], se
q 1:375, ack 1, win 64240, length 374: HTTP: POST /gts1c3 HTTP/1.1
E.....@.@.........CC.,.P....q.#.P...j...POST /gts1c3 HTTP/1.1
Host: ocsp.pki.goog
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/ocsp-request
Content-Length: 84
Connection: keep-alive

0R0P0N0L0J0     ..+..........y...a4...GB....$.c...t.......=...F..q5.'.....$8.Yn.
....+.3
05:17:37.568093 eth0  In  IP 142.250.67.67.80 > 192.168.214.129.33836: Flags [.], ack
 375, win 64240, length 0
E..(..........CC.....P.,q.#.....P....|........
05:17:37.651524 eth0  In  IP 142.250.67.67.80 > 192.168.214.129.33836: Flags [P.], se
q 1:703, ack 375, win 64240, length 702: HTTP: HTTP/1.1 200 OK
E.............CC.....P.,q.#.....P....L..HTTP/1.1 200 OK
Content-Type: application/ocsp-response
Date: Mon, 24 Jan 2022 10:17:37 GMT
Cache-Control: public, max-age=86400
Server: ocsp_responder
Content-Length: 472
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
```



```
0...
......0....     +.....0......0...0.......t.......=...F..q5.'..20220123143331Z0t0r0J0.
.+..........y...a4...GB....$.c...t.......=...F..q5.'.....$8.Yn.
....+.3....20220123143321Z....20220130133320Z0..        *.H...................tKi!..
.Cy..5.>E....1...7.s...I.i....].V........5...T..".F....6e..+..D./....6.Z...8..%. .F7i
....K.'..#. 2."Z.Ry.8G.".!.....o....7...K..~D|.~.i.}._..CH...(.!.....I.
...axt..t......./..m.0.b.C....B.9..np.....R..@[.......$..
<.....W6I.......-  ...`./.d`{.
05:17:37.651561 eth0  Out IP 192.168.214.129.33836 > 142.250.67.67.80: Flags [.], ack
 703, win 63882, length 0
E..(..@.@..O......CC.,.P....q.&.P...i...
05:17:38.581799 eth0  Out IP 192.168.214.129.53046 > 117.18.237.29.80: Flags [S], seq
 1237363207, win 64240, options [mss 1460,sackOK,TS val 2274815913 ecr 0,nop,wscale 7
], length 0
E..<..@.@..q....u....6.PI.....................
............
05:17:38.582146 eth0  Out IP 192.168.214.129.53048 > 117.18.237.29.80: Flags [S], seq
 3883036533, win 64240, options [mss 1460,sackOK,TS val 2274815914 ecr 0,nop,wscale 7
], length 0
E..<DF@.@.......u....8.P.rou..................
............
05:17:38.612621 eth0  In  IP 117.18.237.29.80 > 192.168.214.129.53048: Flags [S.], se
q 775448756, ack 3883036534, win 64240, options [mss 1460], length 0
E..,......@fu........P.8.8h..rov`....l........
10 packets captured
13 packets received by filter
0 packets dropped by kernel
```

Step 6: To save packets to a file instead of displaying them on screen, use the option -w:

sudo tcpdump -i any -c10 -nn -w webserver.pcap port 80

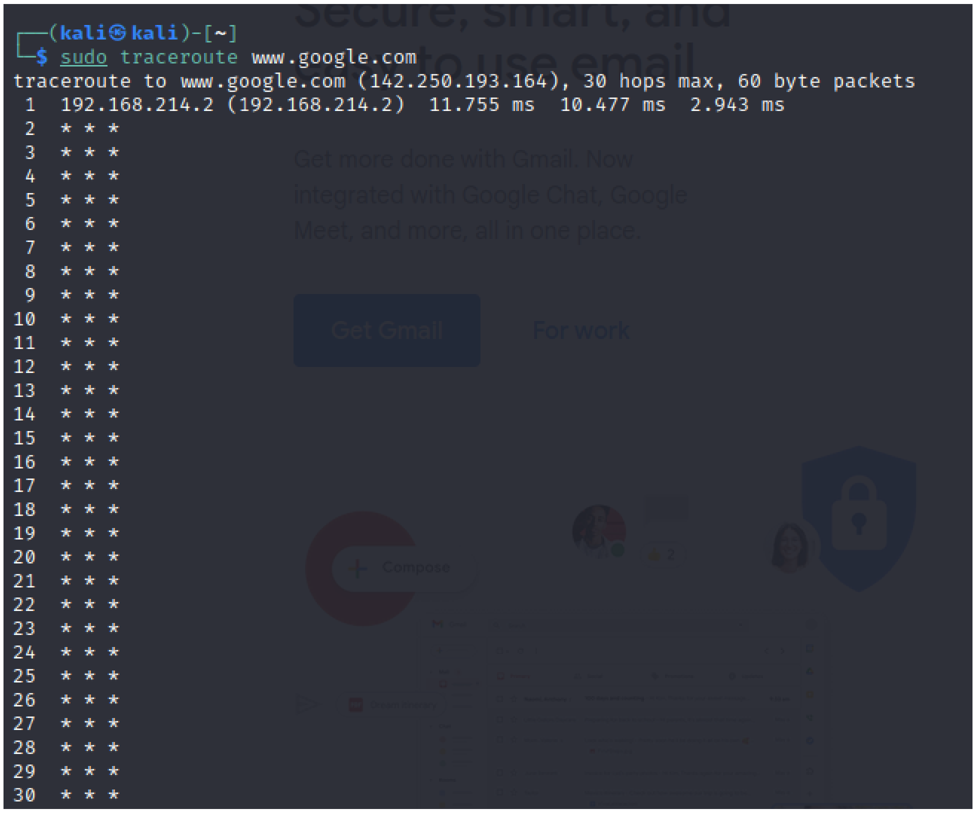


```
┌──(kali㉿kali)-[~]
└─$ sudo tcpdump -i any -c10 -nn -w capture.pcap port 80
tcpdump: data link type LINUX_SLL2
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
10 packets captured
15 packets received by filter
0 packets dropped by kernel
```

Task 5: Perform Traceroute checks

Step 1: Run the traceroute using the following command.

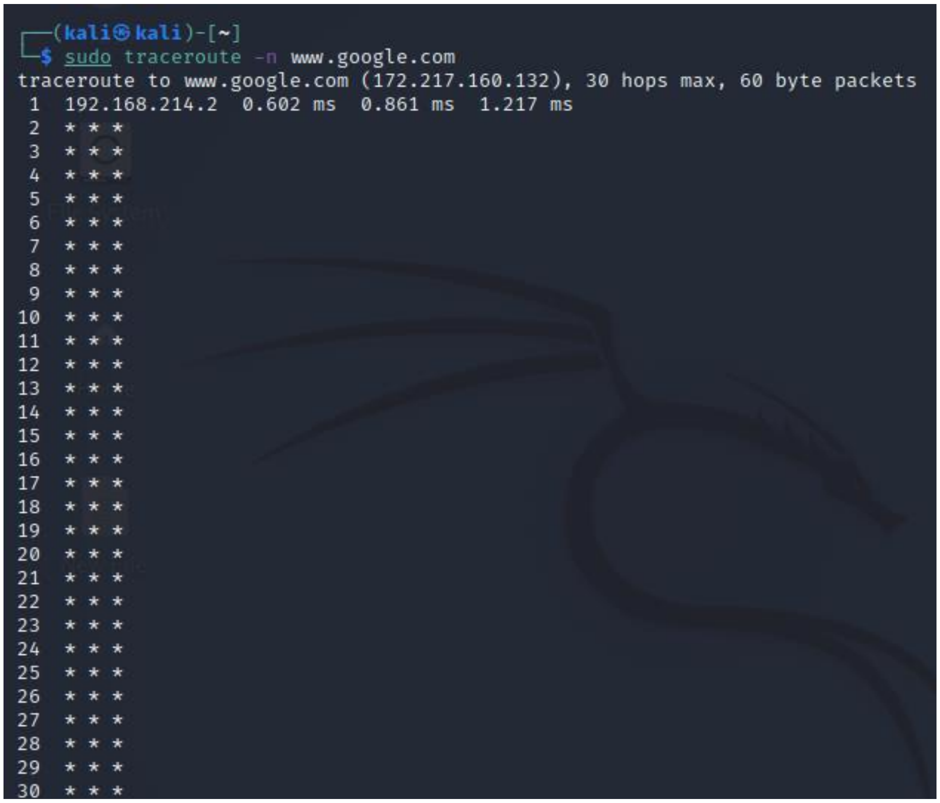
sudo traceroute www.google.com



```
┌──(kali㉿kali)-[~]
└─$ sudo traceroute www.google.com
traceroute to www.google.com (142.250.193.164), 30 hops max, 60 byte packets
 1  192.168.214.2 (192.168.214.2)  11.755 ms  10.477 ms  2.943 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
```

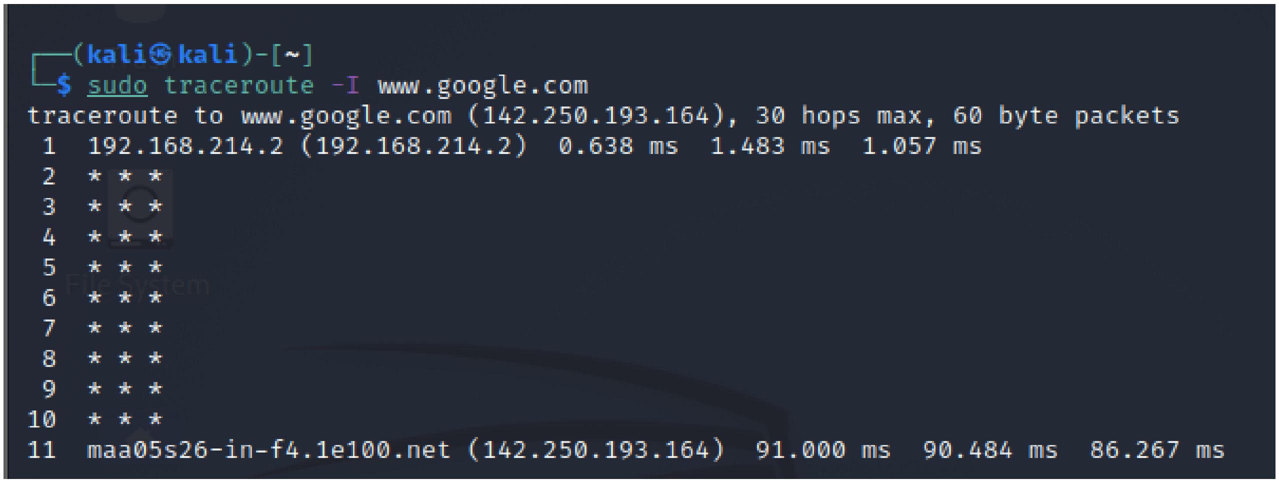
Step 2: Analyze destination address of google.com and no. of hops
The destination address is **142.250.196.164** [FOUND OUT BY PINGING IN WINDOWS]
The total number of hops is **30**, and most of pings have been timed out.

Step 3: To speed up the process, you can disable the mapping of IP addresses with hostnames
by using the -n option
sudo traceroute -n www.google.com



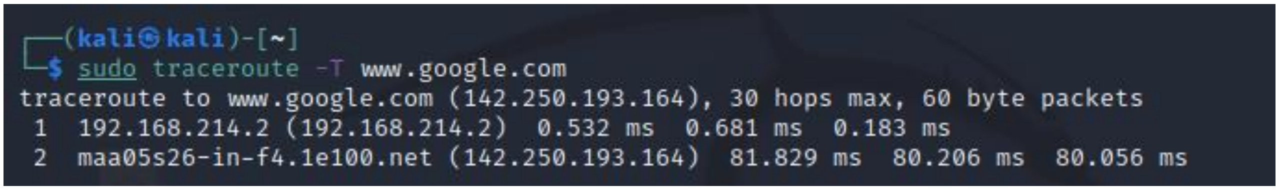
```
┌──(kali㉿kali)-[~]
└─$ sudo traceroute -n www.google.com
traceroute to www.google.com (172.217.160.132), 30 hops max, 60 byte packets
 1  192.168.214.2  0.602 ms  0.861 ms  1.217 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
```

Step 4: The -I option is necessary so that the traceroute uses ICMP.
sudo traceroute -I www.google.com



```
┌──(kali㉿kali)-[~]
└─$ sudo traceroute -I www.google.com
traceroute to www.google.com (142.250.193.164), 30 hops max, 60 byte packets
 1  192.168.214.2 (192.168.214.2)  0.638 ms  1.483 ms  1.057 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  maa05s26-in-f4.1e100.net (142.250.193.164)  91.000 ms  90.484 ms  86.267 ms
```

Step 5: By default, traceroute uses icmp (ping) packets. If you'd rather test a TCP connection to gather data more relevant to web server, you can use the -T flag.
sudo traceroute -T www.google.com

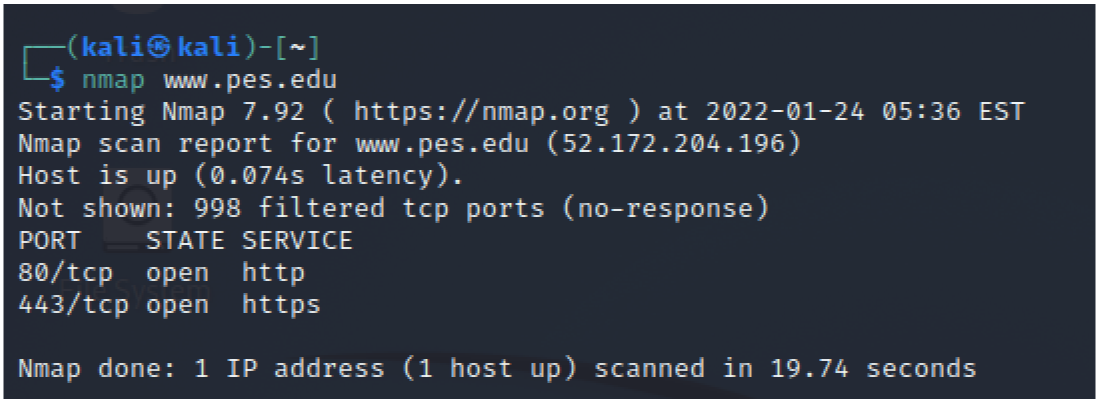


```
┌──(kali㉿kali)-[~]
└─$ sudo traceroute -T www.google.com
traceroute to www.google.com (142.250.193.164), 30 hops max, 60 byte packets
 1  192.168.214.2 (192.168.214.2)  0.532 ms  0.681 ms  0.183 ms
 2  maa05s26-in-f4.1e100.net (142.250.193.164)  81.829 ms  80.206 ms  80.056 ms
```

Task 6: Explore an entire network for information (Nmap)

Step 1: You can scan a host using its host name or IP address, for instance.

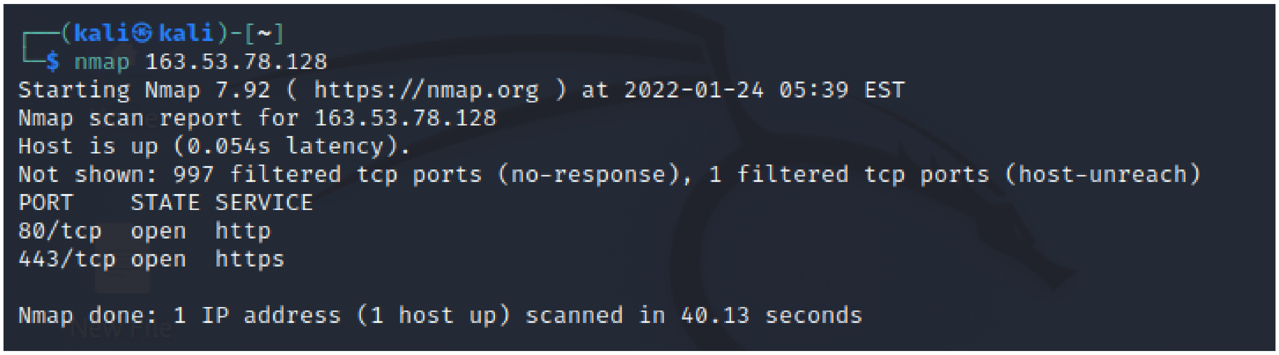
nmap www.pes.edu



```
┌──(kali㉿kali)-[~]
└─$ nmap www.pes.edu
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-24 05:36 EST
Nmap scan report for www.pes.edu (52.172.204.196)
Host is up (0.074s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 19.74 seconds
```

Step 2: Alternatively, use an IP address to scan.

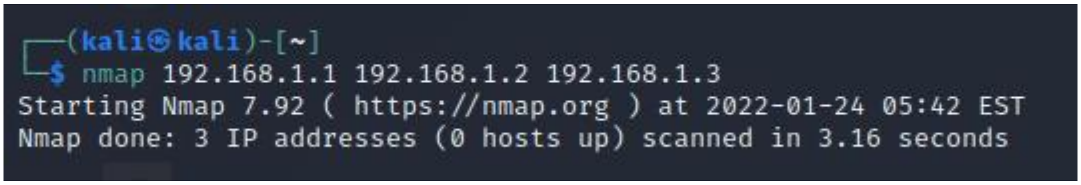
nmap 163.53.78.128



```
┌──(kali㉿kali)-[~]
└─$ nmap 163.53.78.128
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-24 05:39 EST
Nmap scan report for 163.53.78.128
Host is up (0.054s latency).
Not shown: 997 filtered tcp ports (no-response), 1 filtered tcp ports (host-unreach)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 40.13 seconds
```

Step 3: Scan multiple IP address or subnet (IPv4)

nmap 192.168.1.1 192.168.1.2 192.168.1.3



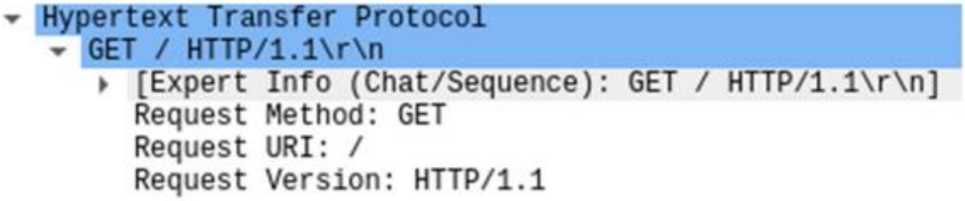
```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.1.1 192.168.1.2 192.168.1.3
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-24 05:42 EST
Nmap done: 3 IP addresses (0 hosts up) scanned in 3.16 seconds
```

Questions on above observations:

1) Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server?

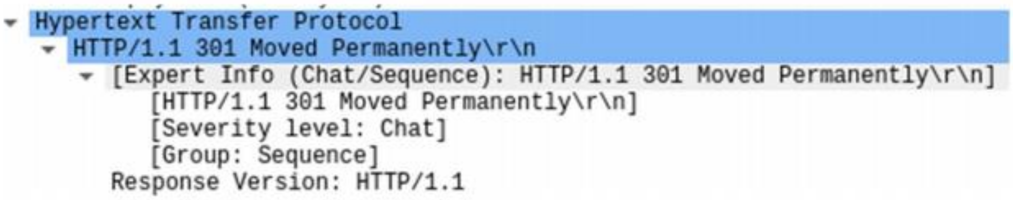
Answer: The Firefox browser used is running HTTP v1.1, and this can be seen in the request header which contains the method (GET) followed by the HTTP version. Similarly, the HTTP version of the web server is v1.1 and can be seen in the header of the HTTP response sent back to the browser.

Request:



```
▾ Hypertext Transfer Protocol
  ▾ GET / HTTP/1.1\r\n
    ▸ [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /
      Request Version: HTTP/1.1
```

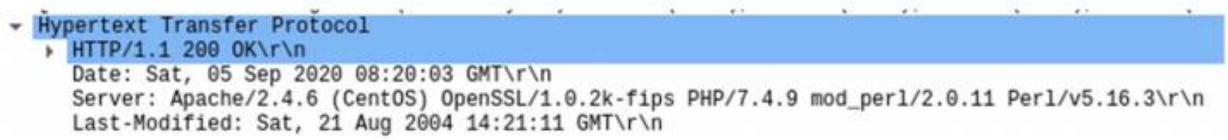
Response:



```
▾ Hypertext Transfer Protocol
  ▾ HTTP/1.1 301 Moved Permanently\r\n
    ▾ [Expert Info (Chat/Sequence): HTTP/1.1 301 Moved Permanently\r\n]
        [HTTP/1.1 301 Moved Permanently\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Response Version: HTTP/1.1
```

2) When was the HTML file that you are retrieving last modified at the server?

Answer: We can find the last modified time of the HTML file at the server by observing the Last-Modified field of the HTTP response object. The Last-Modified field stores a timestamp of the last modification time. Example:



```
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Date: Sat, 05 Sep 2020 08:20:03 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.9 mod_perl/2.0.11 Perl/v5.16.3\r\n
    Last-Modified: Sat, 21 Aug 2004 14:21:11 GMT\r\n
```

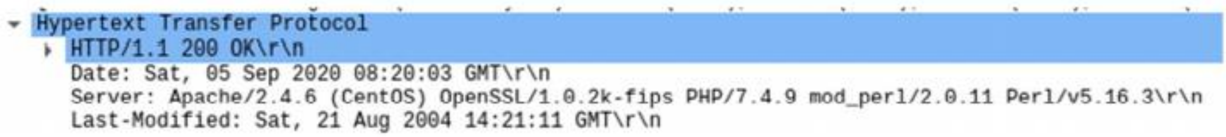
3) How to tell ping to exit after a specified number of ECHO_REQUEST packets?

Answer: Ping continues to send ICMP packages until it receives an interrupt signal. To specify the number of ECHO_REQUEST packages after which ping will exit, we can use the -c option followed by the number of packages.

ping -c 10 www.pes.edu

4) How will you identify remote host apps and OS?

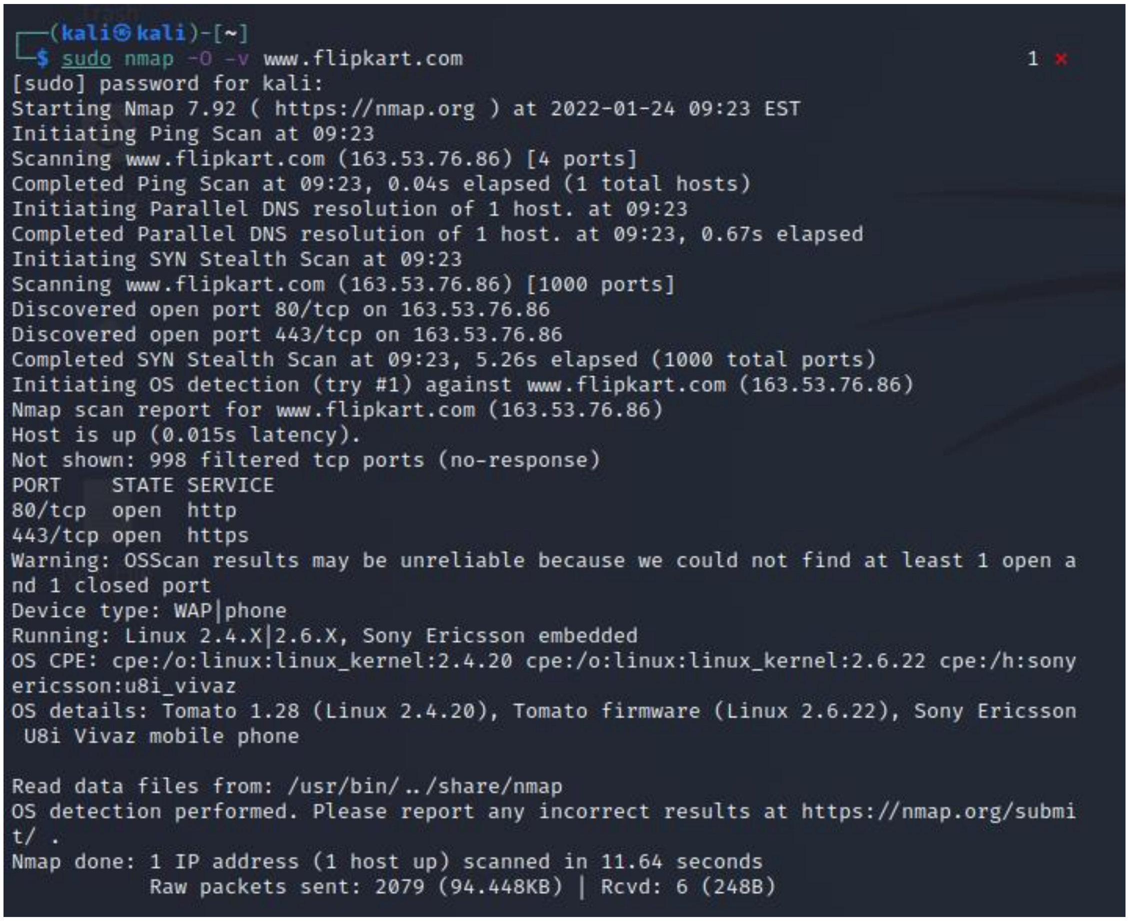
Answer: We can obtain the remote host app and OS of the server by observing the Server files of the HTTP response object. The Server field stores the remote host app or server on which it is hosted and the OS too. Example:



```
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Date: Sat, 05 Sep 2020 08:20:03 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.9 mod_perl/2.0.11 Perl/v5.16.3\r\n
    Last-Modified: Sat, 21 Aug 2004 14:21:11 GMT\r\n
```

We can use nmap to find the OS too. It will scan the network to find information about the remote host apps and OS.

sudo nmap -O -v www.flipkart.co



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -O -v www.flipkart.com                                              1 ✖
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-24 09:23 EST
Initiating Ping Scan at 09:23
Scanning www.flipkart.com (163.53.76.86) [4 ports]
Completed Ping Scan at 09:23, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 09:23
Completed Parallel DNS resolution of 1 host. at 09:23, 0.67s elapsed
Initiating SYN Stealth Scan at 09:23
Scanning www.flipkart.com (163.53.76.86) [1000 ports]
Discovered open port 80/tcp on 163.53.76.86
Discovered open port 443/tcp on 163.53.76.86
Completed SYN Stealth Scan at 09:23, 5.26s elapsed (1000 total ports)
Initiating OS detection (try #1) against www.flipkart.com (163.53.76.86)
Nmap scan report for www.flipkart.com (163.53.76.86)
Host is up (0.015s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
Warning: OSScan results may be unreliable because we could not find at least 1 open a
nd 1 closed port
Device type: WAP|phone
Running: Linux 2.4.X|2.6.X, Sony Ericsson embedded
OS CPE: cpe:/o:linux:linux_kernel:2.4.20 cpe:/o:linux:linux_kernel:2.6.22 cpe:/h:sony
ericsson:u8i_vivaz
OS details: Tomato 1.28 (Linux 2.4.20), Tomato firmware (Linux 2.6.22), Sony Ericsson
 U8i Vivaz mobile phone

Read data files from: /usr/bin/../share/nmap
OS detection performed. Please report any incorrect results at https://nmap.org/submi
t/ .
Nmap done: 1 IP address (1 host up) scanned in 11.64 seconds
           Raw packets sent: 2079 (94.448KB) | Rcvd: 6 (248B)
```