

PES UNIVERSITY

EC CAMPUS, BANGALORE

Name – B. Pravena

SRN – PES2UG19CS076

WEEK – 1

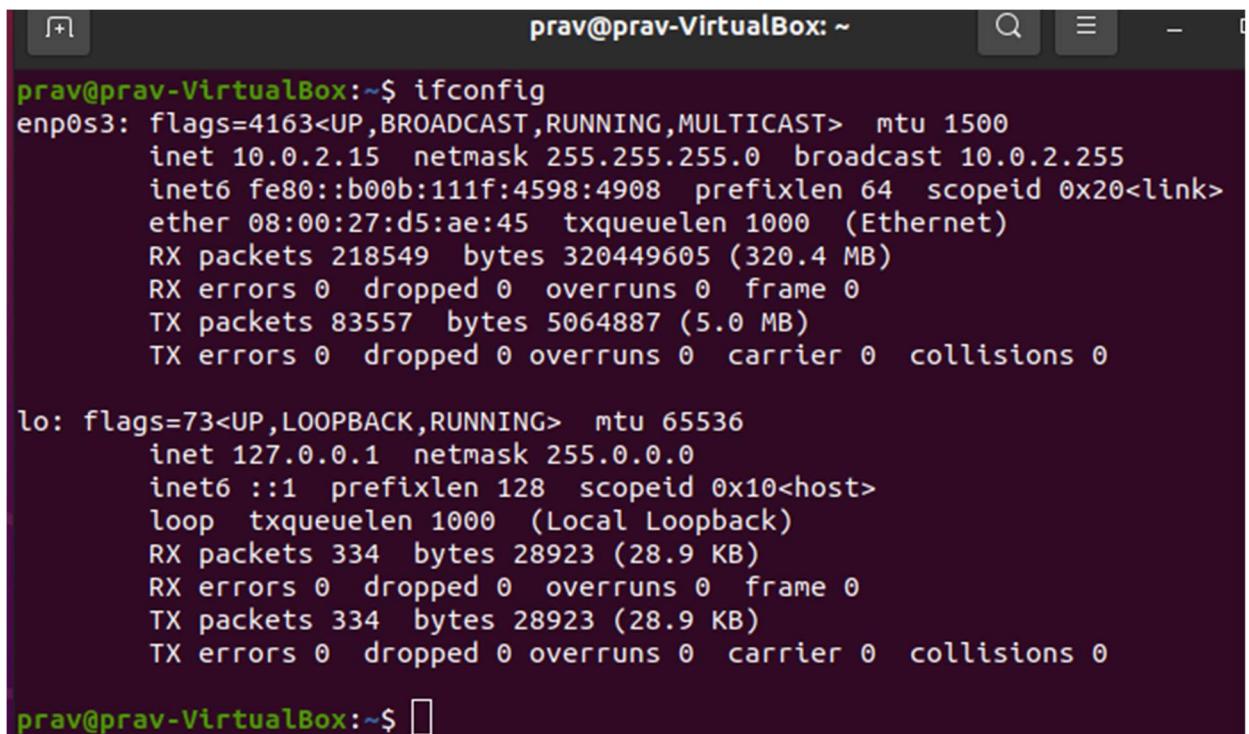
SUBJECT - Computer Networks Laboratory

Objective :- To study and understand the basic networking tools - wireshark, Tcpdump, Ping, Traceroute and Netcat.

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

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

ifconfig (or) ip addr show



```
prav@prav-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::b00b:111f:4598:4908 prefixlen 64 scopeid 0x20<link>
          ether 08:00:27:d5:ae:45 txqueuelen 1000 (Ethernet)
            RX packets 218549 bytes 320449605 (320.4 MB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 83557 bytes 5064887 (5.0 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 334 bytes 28923 (28.9 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 334 bytes 28923 (28.9 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

prav@prav-VirtualBox:~$
```

Analyze and fill the table:-

ip address table:

Interface name	IP address (IPv4 / IPv6)	MAC address
enp0s3	IPV4 - 10.0.2.15 IPV6 – fe80::b00b:111f:4598:4908	ether 08:00:27:d5:ae:45
lo	IPV4 - 127.0.0.1 IPV6 - ::1	00:00:00:00:00:00

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

sudo ifconfig lo 10.0.2.10 netmask 255.255.255.0

```
prav@prav-VirtualBox:~$ sudo ifconfig lo 10.0.2.10 netmask 255.255.255.0
prav@prav-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::b00b:111f:4598:4908 prefixlen 64 scopeid 0x20<link>
              ether 08:00:27:d5:ae:45 txqueuelen 1000 (Ethernet)
              RX packets 218560 bytes 320450767 (320.4 MB)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 83574 bytes 5066410 (5.0 MB)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 10.0.2.10 netmask 255.255.255.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
          RX packets 351 bytes 31153 (31.1 KB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 351 bytes 31153 (31.1 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

prav@prav-VirtualBox:~$ 
```

The IP address of “lo” has changed from previous.

Step 3: Activate/deactivate a network interface, type.

sudo ifconfig lo down

```
prav@prav-VirtualBox:~$ sudo ifconfig lo down
prav@prav-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::b00b:111f:4598:4908 prefixlen 64 scopeid 0x20<link>
              ether 08:00:27:d5:ae:45 txqueuelen 1000 (Ethernet)
              RX packets 218576 bytes 320451847 (320.4 MB)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 83605 bytes 5069084 (5.0 MB)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

prav@prav-VirtualBox:~$ 
```

sudo ifconfig lo up

```
prav@prav-VirtualBox:~$ sudo ifconfig lo up
prav@prav-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::b00b:111f:4908:4908 prefixlen 64 scopeid 0x20<link>
          ether 08:00:27:d5:ae:45 txqueuelen 1000 (Ethernet)
            RX packets 218590 bytes 320453041 (320.4 MB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 83625 bytes 5070811 (5.0 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 367 bytes 33323 (33.3 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 367 bytes 33323 (33.3 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

prav@prav-VirtualBox:~$ █
```

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

ip neigh

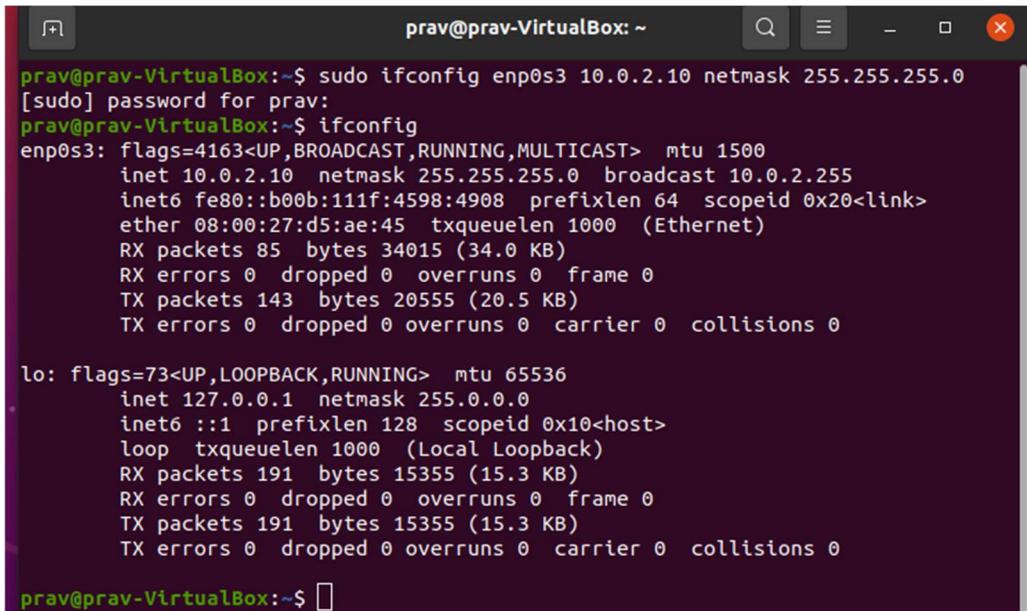
When the wired connection is ON and OFF respectively.

```
prav@prav-VirtualBox:~$ ip neigh
10.0.2.2 dev enp0s3 lladdr 52:54:00:12:35:02 STALE
prav@prav-VirtualBox:~$ ip neigh
prav@prav-VirtualBox:~$ █
```

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

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

sudo ifconfig enp0s3 10.0.2.10 netmask 255.255.255.0

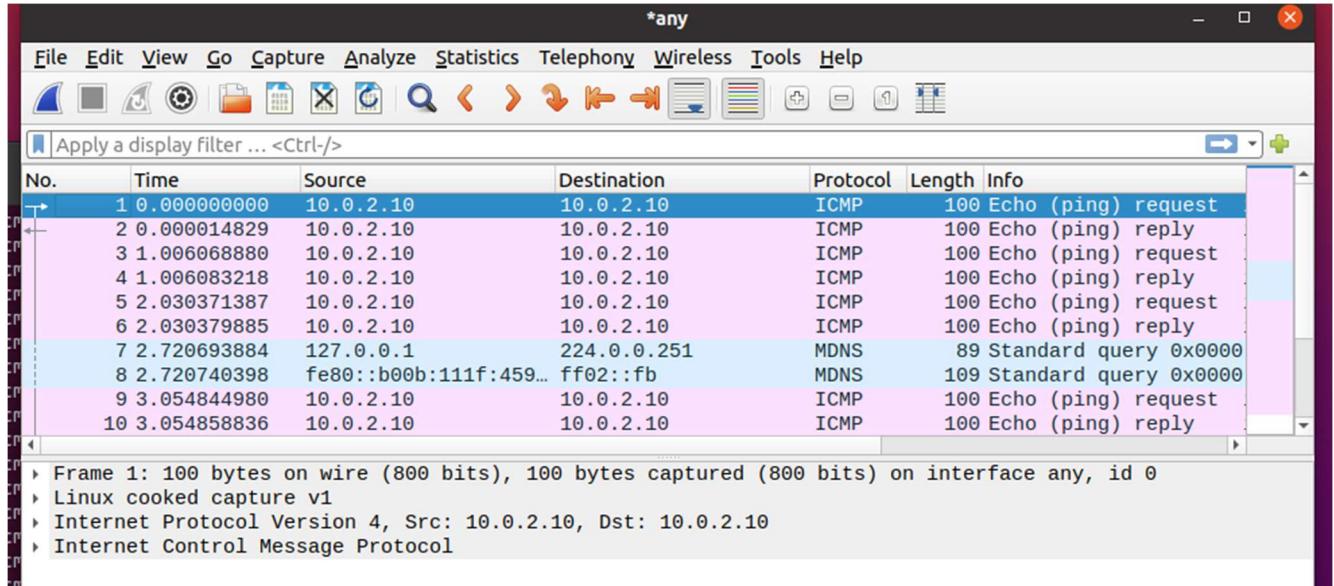


```
prav@prav-VirtualBox:~$ sudo ifconfig enp0s3 10.0.2.10 netmask 255.255.255.0
[sudo] password for prav:
prav@prav-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.10 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::b00b:111f:4908:4908 prefixlen 64 scopeid 0x20<link>
          ether 08:00:27:d5:ae:45 txqueuelen 1000 (Ethernet)
            RX packets 85 bytes 34015 (34.0 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 143 bytes 20555 (20.5 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 191 bytes 15355 (15.3 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 191 bytes 15355 (15.3 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

prav@prav-VirtualBox:~$ █
```

Step 2: Launch Wireshark and select ‘any’ interface



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

The terminal window shows the command "ping 10.0.2.10" being run. The output displays 20 ICMP echo requests (seq 1 to 20) sent to the destination IP 10.0.2.10. Each request shows the source as 10.0.2.10, TTL of 64, and a time between 0.035 ms and 0.068 ms. After 20 packets, the statistics summary shows 20 transmitted, 20 received, 0% packet loss, and an average round-trip time (rtt) of 0.035 ms.

```
prav@prav-VirtualBox:~$ ping 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
64 bytes from 10.0.2.10: icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from 10.0.2.10: icmp_seq=2 ttl=64 time=0.051 ms
64 bytes from 10.0.2.10: icmp_seq=3 ttl=64 time=0.052 ms
64 bytes from 10.0.2.10: icmp_seq=4 ttl=64 time=0.050 ms
64 bytes from 10.0.2.10: icmp_seq=5 ttl=64 time=0.035 ms
64 bytes from 10.0.2.10: icmp_seq=6 ttl=64 time=0.049 ms
64 bytes from 10.0.2.10: icmp_seq=7 ttl=64 time=0.050 ms
64 bytes from 10.0.2.10: icmp_seq=8 ttl=64 time=0.051 ms
64 bytes from 10.0.2.10: icmp_seq=9 ttl=64 time=0.039 ms
64 bytes from 10.0.2.10: icmp_seq=10 ttl=64 time=0.049 ms
64 bytes from 10.0.2.10: icmp_seq=11 ttl=64 time=0.055 ms
64 bytes from 10.0.2.10: icmp_seq=12 ttl=64 time=0.051 ms
64 bytes from 10.0.2.10: icmp_seq=13 ttl=64 time=0.045 ms
64 bytes from 10.0.2.10: icmp_seq=14 ttl=64 time=0.048 ms
64 bytes from 10.0.2.10: icmp_seq=15 ttl=64 time=0.052 ms
64 bytes from 10.0.2.10: icmp_seq=16 ttl=64 time=0.052 ms
64 bytes from 10.0.2.10: icmp_seq=17 ttl=64 time=0.051 ms
64 bytes from 10.0.2.10: icmp_seq=18 ttl=64 time=0.068 ms
64 bytes from 10.0.2.10: icmp_seq=19 ttl=64 time=0.051 ms
64 bytes from 10.0.2.10: icmp_seq=20 ttl=64 time=0.051 ms
^C
--- 10.0.2.10 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19463ms
rtt min/avg/max/mdev = 0.035/0.049/0.068/0.006 ms
prav@prav-VirtualBox:~$
```

Observations to be made

Step 4: Analyze the following in Terminal

- TTL = 64
- Protocol used by ping = ICMP
- Time :-

Min = 0.035ms

Avg = 0.049ms

Max = 0.068ms

Mdev = 0.006ms

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.

The screenshot shows the Wireshark interface with the following details:

- Packet List:** Shows 14 ICMP packets. The first packet (index 1) is selected and highlighted in blue. It is an "Echo (ping) request" from source 10.0.2.10 to destination 10.0.2.10, with a length of 100 bytes.
- Details Pane:** Expanded for the selected packet (index 1).
 - Identification:** 0xe04f (57423)
 - Flags:** 0x40, Don't fragment
 - Fragment Offset:** 0
 - Time to Live:** 64
 - Protocol:** ICMP (1)
 - Header Checksum:** 0x4246 [validation disabled]
 - [Header checksum status: Unverified]**
 - Source Address:** 10.0.2.10
 - Destination Address:** 10.0.2.10
- Hex and ASCII Pans:** Below the details pane, the hex and ASCII panes show the raw data of the selected ICMP request. The hex pane shows the byte sequence: 00 00 03 04 00 06 00 00 00 00 00 00 00 00 00 00 08 00 The ASCII pane shows the corresponding characters: E..T.0@. @ BF...Da`B..%.... !"# \$%%'(*)*+, -./0123 4567.

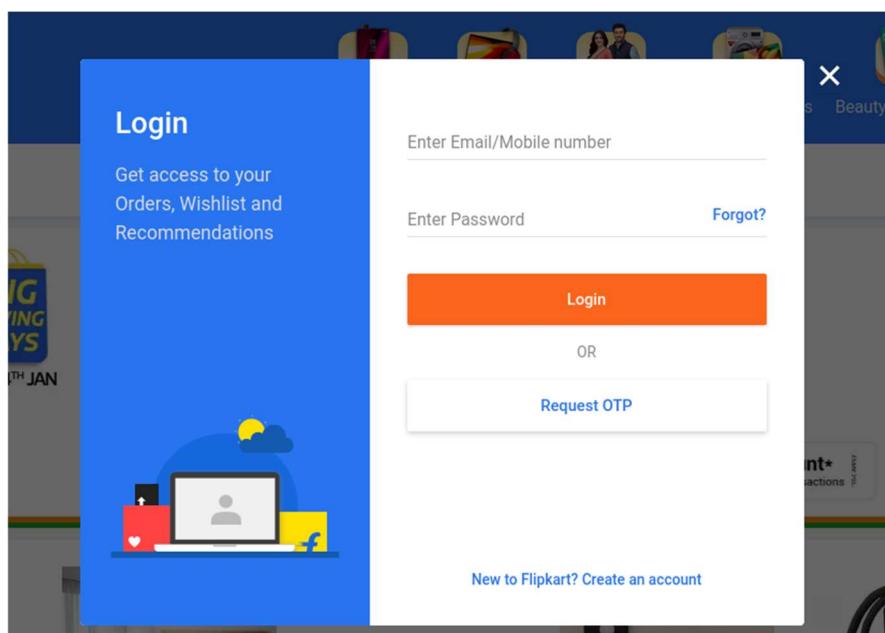
Details	First Echo Request	First Echo Reply
Frame Number	1	2
Source IP address	10.0.2.10	10.0.2.10
Destination IP address	10.0.2.10	10.0.2.10
ICMP Type Value	8(Echo (ping) request)	0(Echo (ping) reply)
ICMP Code Value	0	0
Source Ethernet Address	00:00:00:00	00:00:00:00
Destination Ethernet Address	00:00:00:00	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

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



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:

No.	Time	Source	Destination	Protocol	Length	Info
12	0.161329025	10.0.2.15	34.107.221.82	HTTP	352	GET /success.txt HTTP
14	0.176868983	34.107.221.82	10.0.2.15	HTTP	276	HTTP/1.1 200 OK (tex
43	0.256938371	10.0.2.15	34.107.221.82	HTTP	357	GET /success.txt?ipv4
53	0.271272777	34.107.221.82	10.0.2.15	HTTP	276	HTTP/1.1 200 OK (tex
131	0.945104553	10.0.2.15	117.18.237.29	OCSP	435	Request
133	0.956454243	117.18.237.29	10.0.2.15	OCSP	855	Response
147	1.147129532	10.0.2.15	117.18.237.29	OCSP	435	Request
149	1.157655878	117.18.237.29	10.0.2.15	OCSP	855	Response
186	1.836839196	10.0.2.15	117.18.237.29	OCSP	435	Request
188	1.846783753	117.18.237.29	10.0.2.15	OCSP	854	Response

Frame 12: 352 bytes on wire (2816 bits), 352 bytes captured (2816 bits) on interface any, id 0
Linux cooked capture v1

Packet type: Sent by us (4)
Link-layer address type: Ethernet (1)
Link-layer address length: 6
Source: PcsCompu_d5:ae:45 (08:00:27:d5:ae:45)
Unused: 0000
Protocol: IPv4 (0x0800)

Internet Protocol Version 4, Src: 10.0.2.15, Dst: 34.107.221.82
Transmission Control Protocol, Src Port: 45678, Dst Port: 80, Seq: 1, Ack: 1, Len: 296
Hypertext Transfer Protocol

No.	Time	Source	Destination	Protocol	Length	Info
12	0.161329025	10.0.2.15	34.107.221.82	HTTP	352	GET /success.txt HTTP
14	0.176868983	34.107.221.82	10.0.2.15	HTTP	276	HTTP/1.1 200 OK (tex
43	0.256938371	10.0.2.15	34.107.221.82	HTTP	357	GET /success.txt?ipv4
53	0.271272777	34.107.221.82	10.0.2.15	HTTP	276	HTTP/1.1 200 OK (tex
131	0.945104553	10.0.2.15	117.18.237.29	OCSP	435	Request
133	0.956454243	117.18.237.29	10.0.2.15	OCSP	855	Response
147	1.147129532	10.0.2.15	117.18.237.29	OCSP	435	Request
149	1.157655878	117.18.237.29	10.0.2.15	OCSP	855	Response
186	1.836839196	10.0.2.15	117.18.237.29	OCSP	435	Request
188	1.846783753	117.18.237.29	10.0.2.15	OCSP	854	Response

Frame 14: 276 bytes on wire (2208 bits), 276 bytes captured (2208 bits) on interface any, id 0
Linux cooked capture v1

Packet type: Unicast to us (0)
Link-layer address type: Ethernet (1)
Link-layer address length: 6
Source: RealtekU_12:35:02 (52:54:00:12:35:02)
Unused: 0000
Protocol: IPv4 (0x0800)

Internet Protocol Version 4, Src: 34.107.221.82, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 80, Dst Port: 45678, Seq: 1, Ack: 297, Len: 220
Hypertext Transfer Protocol
Line-based text data: text/plain (1 lines)

NOTE :- Here the system IP address is 10.0.2.15

Details	First Echo Request	First Echo Reply
Frame Number	12	14
Source Port	45678	80
Destination Port	80	45678
Source IP address	10.0.2.15	34.107.221.82
Destination IP address	34.107.221.82	10.0.2.15
Source Ethernet Address	PcsCompu_d5:ae:45 08:00:27:d5:ae:45	RealtekU_12:35:02 52:54:00:12:35:02

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

The Wireshark interface displays two captures. The top capture shows the initial HTTP request from the browser (10.0.2.15) to the server (34.107.221.82). The bottom capture shows the server's response (HTTP/1.1 200 OK) back to the browser.

Request (Frame 12):

```

GET /success.txt HTTP/1.1\r\n
Host: detectportal.firefox.com\r\n
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0\r\n
Accept: */*\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
Cache-Control: no-cache\r\n
Pragma: no-cache\r\n
Connection: keep-alive\r\n
\r\n
[Full request URI: http://detectportal.firefox.com/success.txt]
[HTTP request 1/1]
[Response in frame: 14]
    
```

Response (Frame 14):

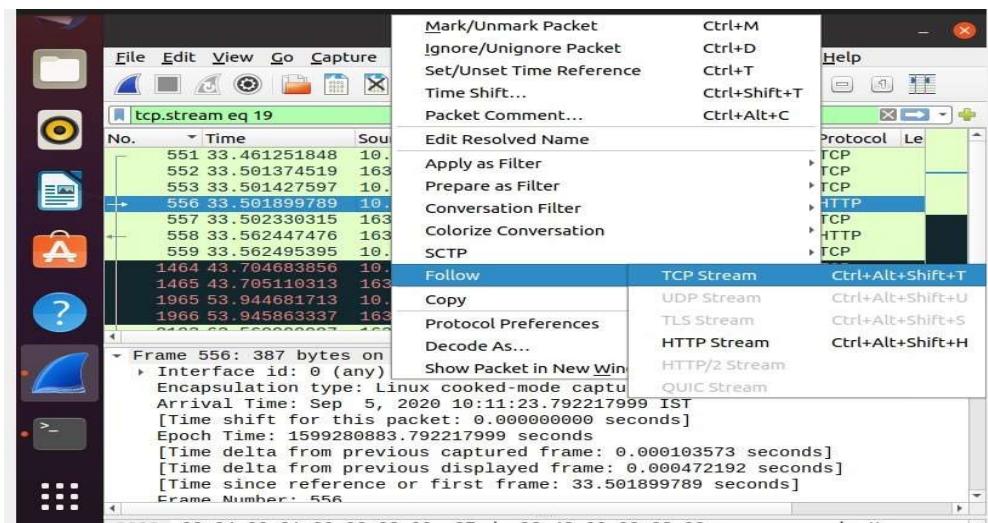
```

HTTP/1.1 200 OK\r\n
Server: nginx\r\n
Date: Mon, 25 Jan 2021 16:52:22 GMT\r\n
Content-Type: text/plain\r\n
Content-Length: 8\r\n
Via: 1.1 google\r\n
Age: 68335\r\n
Cache-Control: public, must-revalidate, max-age=0, s-maxage=86400\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.015539958 seconds]
[Request in frame: 12]
[Request URI: http://detectportal.firefox.com/success.txt]
File Data: 8 bytes
Line-based text data: text/plain (1 lines)
    
```

HTTP Request		HTTP Response	
Get	GET/HTTP/1.1	Server	nginx\r\n
Host	firefox.com\r\n	Content-Type	Text/plain\r\n
User-Agent	Mozilla/5.0(X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0	Date	Mon, 25 Jan 2021 16:52:22 GMT\r\n
Accept-Language	en-US, en;q=0.5\r\n	Location	https://flipkart.com/\r\n
Accept-Encoding	gzip, deflate\r\n	Content-Length	8\r\n
Connection	Keep-alive\r\n	Connection	

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’.



Step 2: Upon following a TCP stream, screenshot the whole window.



Task 4: Capturing packets with tcpdump

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

sudo tcpdump -D

```
prav@prav-VirtualBox:~$ sudo tcpdump -D
[sudo] password for prav:
1.enp0s3 [Up, Running]
2.lo [Up, Running, Loopback]
3.any (Pseudo-device that captures on all interfaces) [Up, Running]
4.bluetooth-monitor (Bluetooth Linux Monitor) [none]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
prav@prav-VirtualBox:~$ 
```

Step 2: Capture all packets in any interface by running this command:

```
sudo tcpdump -i any
```

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

```
prav@prav-VirtualBox:~$ sudo tcpdump -i any
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
21:47:11.199534 IP prav-VirtualBox.56824 > bom05s12-in-f4.1e100.net.https: Flags [P.], seq 3919981754:3919981902, ack 2301838, win 62780, length 148
21:47:11.200490 IP bom05s12-in-f4.1e100.net.https > prav-VirtualBox.56824: Flags [.], ack 148, win 65535, length 0
21:47:11.200695 IP localhost.32924 > localhost.domain: 18696+ [1au] PTR? 164.160.217.172.in-addr.arpa. (57)
21:47:11.200988 IP prav-VirtualBox.36162 > 192.168.1.1.domain: 17357+ [1au] PTR? 164.160.217.172.in-addr.arpa. (57)
21:47:11.201432 IP prav-VirtualBox.56824 > bom05s12-in-f4.1e100.net.https: Flags [P.], seq 148:179, ack 1, win 62780, length 31
21:47:11.201796 IP bom05s12-in-f4.1e100.net.https > prav-VirtualBox.56824: Flags [.], ack 179, win 65535, length 0
21:47:11.207148 IP 192.168.1.1.domain > prav-VirtualBox.36162: 17357 1/0/1 PTR bom05s12-in-f4.1e100.net. (95)
21:47:11.391506 IP localhost.46670 > localhost.domain: 13365+ [1au] PTR? 53.0.0.127.in-addr.arpa. (52)
21:47:12.746322 IP prav-VirtualBox.35350 > 117.18.237.29.http: Flags [.], ack 1538399, win 63920, length 0
21:47:12.746784 IP localhost.46837 > localhost.domain: 64397+ [1au] PTR? 29.237.18.117.in-addr.arpa. (55)
21:47:12.746922 IP 117.18.237.29.http > prav-VirtualBox.35350: Flags [.], ack 1, win 65535, length 0
21:47:12.747409 IP prav-VirtualBox.38117 > 192.168.1.1.domain: 14803+ [1au] PTR? 29.237.18.117.in-addr.arpa. (55)
21:47:12.778670 IP 192.168.1.1.domain > prav-VirtualBox.38117: 14803 NXDomain 0/1/1 (126)
21:47:12.778951 IP prav-VirtualBox.38117 > 192.168.1.1.domain: 14803+ PTR? 29.237.18.117.in-addr.arpa. (44)
21:47:12.783199 IP 192.168.1.1.domain > prav-VirtualBox.38117: 14803 NXDomain 0/1/1 (126)
21:47:13.257961 IP prav-VirtualBox.34002 > maa05s13-in-f3.1e100.net.http: Flags [.], ack 2816703, win 63791, length 0
21:47:13.258499 IP maa05s13-in-f3.1e100.net.http > prav-VirtualBox.34002: Flags [.], ack 1, win 65535, length 0
21:47:13.258766 IP localhost.33600 > localhost.domain: 57157+ [1au] PTR? 67.67.250.142.in-addr.arpa. (55)
21:47:13.259043 IP prav-VirtualBox.36786 > 192.168.1.1.domain: 45513+ [1au] PTR? 67.67.250.142.in-addr.arpa. (55)
21:47:13.363597 IP 192.168.1.1.domain > prav-VirtualBox.36786: 45513 1/0/1 PTR maa05s13-in-f3.1e100.net. (93)
21:47:13.364195 IP localhost.domain > localhost.33600: 57157 1/0/1 PTR maa05s13-in-f3.1e100.net. (93)
21:47:13.769954 IP prav-VirtualBox.59498 > 82.221.107.34.bc.googleusercontent.com.http: Flags [.], ack 1856882, win 64020, length 0
21:47:13.770069 IP prav-VirtualBox.59500 > 82.221.107.34.bc.googleusercontent.com.http: Flags [.], ack 1920442, win 64020, length 0
21:47:13.770415 IP 82.221.107.34.bc.googleusercontent.com.http > prav-VirtualBox.59498: Flags [.], ack 1, win 65535, length 0
21:47:13.770425 IP 82.221.107.34.bc.googleusercontent.com.http > prav-VirtualBox.59500: Flags [.], ack 1, win 65535, length 0
```

Observations

Step 3: Understand the output format.

- **Timestamp:** The first field is the timestamp of received packet as per the local clock.
- **IP:** Network layer protocol.
- **Source IP>Destination IP:** The next two fields represent the Source IP address followed by Destination IP Address.
- **TCP FLAGS:** Describe the status of the connection. These can also include a combination of values.
- **Sequence Number:** Talks about the sequence number of the data contained in the packet
- **Ack Number:** the sequence number of the next byte the receiver expects to receive.
- **win:** Window Size, which represents the number of bytes available in the receiving buffer.
- **Packet Length:** represents the length, in bytes, of the payload data.

Reference – <https://opensource.com/article/18/10/introduction-tcpdump>

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
```

```
prav@prav-VirtualBox:~$ sudo tcpdump -i any -c5 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
22:05:32.776856 IP prav-VirtualBox > bom07s15-in-f14.1e100.net: ICMP echo request, id
6, seq 1, length 64
22:05:32.863730 IP bom07s15-in-f14.1e100.net > prav-VirtualBox: ICMP echo reply, id 6,
seq 1, length 64
22:05:33.778768 IP prav-VirtualBox > bom07s15-in-f14.1e100.net: ICMP echo request, id
6, seq 2, length 64
22:05:33.815361 IP bom07s15-in-f14.1e100.net > prav-VirtualBox: ICMP echo reply, id 6,
seq 2, length 64
22:05:34.780550 IP prav-VirtualBox > bom07s15-in-f14.1e100.net: ICMP echo request, id
6, seq 3, length 64
5 packets captured
5 packets received by filter
0 packets dropped by kernel
prav@prav-VirtualBox:~$
```

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
```

```
prav@prav-VirtualBox:~$ sudo tcpdump -i any -c10 -nn -A port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
22:13:27.454284 IP 10.0.2.15.35512 > 117.18.237.29.80: Flags [S], seq 2579740302, win 64240, options [mss 1460,sackOK,TS val 705535150 ecr 0,nop,wscale 7], length 0
E..<..@.@
.....P.....nm.....
*.....
22:13:27.485708 IP 10.0.2.15.35514 > 117.18.237.29.80: Flags [S], seq 3301434806, win 64240, options [mss 1460,sackOK,TS val 705535181 ecr 0,nop,wscale 7], length 0
E..<(^@.@
.....P.....nm.....
*.....
22:13:27.490242 IP 10.0.2.15.35516 > 117.18.237.29.80: Flags [S], seq 2923110307, win 64240, options [mss 1460,sackOK,TS val 705535186 ecr 0,nop,wscale 7], length 0
E..<..@.F.
.....P;.....nm.....
*.....
22:13:27.495946 IP 117.18.237.29.80 > 10.0.2.15.35512: Flags [S.], seq 207040001, ack 2579740303, win 65535, options [mss 1460], length 0
E.,,...@..U...
.....P...W...: ...$......
22:13:27.655990 IP 10.0.2.15.35512 > 117.18.237.29.80: Flags [.], ack 1, win 64240, length 0
E.(..@. .
.....U....P....W..P...nY..
22:13:27.656470 IP 10.0.2.15.35512 > 117.18.237.29.80: Flags [P.], seq 1:380, ack 1, win 64240, length 379: HTTP: POST / HTTP/1.1
E....@..@.0
.....U....P....W..P...o...POST / HTTP/1.1
Host: ocsp.digicert.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/ocsp-request
Content-Length: 83
Connection: keep-alive
```

```

00000M0K0IO  ..+.....z...'.5...C.....a..1a./(..F8.,.....2....R..I..."U
22:13:27.656901 IP 117.18.237.29.80 > 10.0.2.15.35512: Flags [., ack 380, win 65535, length 0
E..(....@...U...
....P...W....
P...*f.....
22:13:27.684898 IP 117.18.237.29.80 > 10.0.2.15.35516: Flags [S.], seq 207104001, ack 2923110308, win 65535, options [mss 1460], length 0
E..,...@...U...
....P...X(.;... .
22:13:27.684984 IP 10.0.2.15.35516 > 117.18.237.29.80: Flags [., ack 1, win 64240, length 0
E..(@.F.
....u.....P.;...X(.P...nY..
22:13:27.685007 IP 117.18.237.29.80 > 10.0.2.15.35514: Flags [S.], seq 207168001, ack 3301434807, win 65535, options [mss 1460], length 0
E..,...@...U...
....P...Y"....` .
10 packets captured
11 packets received by filter
0 packets dropped by kernel
prav@prav-VirtualBox:~$ 

```

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

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

```

prav@prav-VirtualBox:~$ sudo tcpdump -i any -c10 -nn -w webserver.pcap port 80
[sudo] password for prav:
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
10 packets captured
14 packets received by filter
0 packets dropped by kernel
prav@prav-VirtualBox:~$ 

```

Task 5: Perform Traceroute checks

Step 1: Run the traceroute using the following command.

```
sudo traceroute www.google.com
```

```

prav@prav-VirtualBox:~$ sudo traceroute www.google.com
traceroute to www.google.com (216.58.196.68), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  0.326 ms  0.294 ms  0.284 ms
 2  _gateway (10.0.2.2)  5.821 ms  58.362 ms  58.812 ms

```

Step 2: Analyze destination address of google.com and no. of hops

Destination address = 216.58.196.68

No. of hops = 30

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

```
prav@prav-VirtualBox:~$ sudo traceroute -n www.google.com
traceroute to www.google.com (172.217.160.164), 30 hops max, 60 byte packets
 1  10.0.2.2  0.148 ms  0.120 ms  0.109 ms
 2  10.0.2.2  241.414 ms  241.403 ms  242.208 ms
prav@prav-VirtualBox:~$ 
```

Compared to previous step this has taken lesser time.

Step 4: The **-I** option is necessary so that the traceroute uses ICMP.

sudo traceroute -I www.google.com

```
prav@prav-VirtualBox:~$ sudo traceroute -I www.google.com
traceroute to www.google.com (216.58.196.68), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  0.303 ms  0.241 ms  0.222 ms
 2  192.168.1.1 (192.168.1.1)  57.007 ms * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  182.79.177.69 (182.79.177.69)  17.019 ms  17.919 ms  18.331 ms
 8  72.14.208.234 (72.14.208.234)  23.993 ms  23.139 ms  23.079 ms
 9  108.170.253.97 (108.170.253.97)  23.057 ms  22.974 ms  22.957 ms
10  108.170.253.104 (108.170.253.104)  18.229 ms  18.213 ms  18.007 ms
11  64.233.174.3 (64.233.174.3)  21.510 ms  21.038 ms  22.536 ms
12  209.85.251.242 (209.85.251.242)  41.061 ms  41.034 ms  40.028 ms
13  108.170.248.177 (108.170.248.177)  38.645 ms  39.680 ms  36.777 ms
14  209.85.255.209 (209.85.255.209)  36.231 ms  35.308 ms  38.310 ms
15  bom05s11-in-f4.1e100.net (216.58.196.68)  38.295 ms  38.711 ms  38.634 ms
prav@prav-VirtualBox:~$ 
```

No. of hops =15

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.

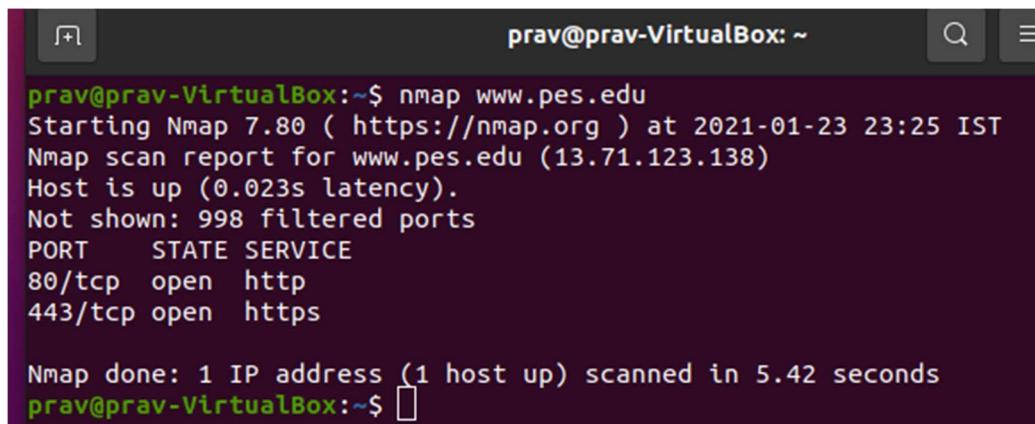
sudo traceroute -T www.google.com

```
prav@prav-VirtualBox:~$ sudo traceroute -T www.google.com
traceroute to www.google.com (172.217.167.132), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  0.240 ms  0.204 ms  0.190 ms
 2  maa03s26-in-f4.1e100.net (172.217.167.132)  67.169 ms  82.962 ms  69.869 ms
prav@prav-VirtualBox:~$ 
```

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

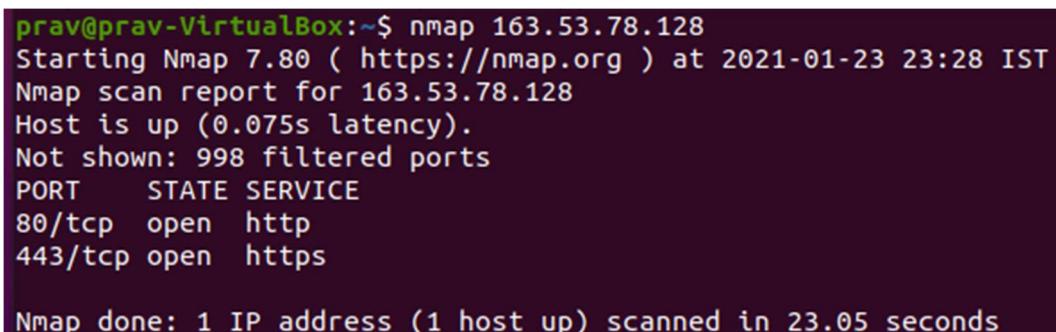


```
prav@prav-VirtualBox:~$ nmap www.pes.edu
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-23 23:25 IST
Nmap scan report for www.pes.edu (13.71.123.138)
Host is up (0.023s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https

Nmap done: 1 IP address (1 host up) scanned in 5.42 seconds
prav@prav-VirtualBox:~$
```

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

nmap 163.53.78.128

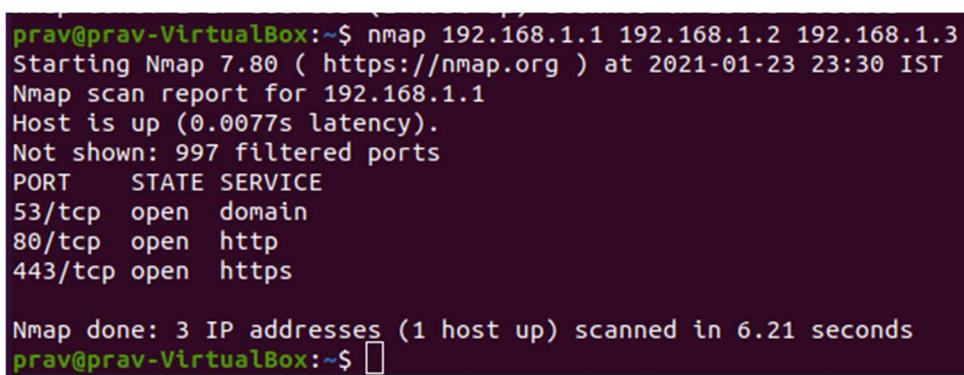


```
prav@prav-VirtualBox:~$ nmap 163.53.78.128
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-23 23:28 IST
Nmap scan report for 163.53.78.128
Host is up (0.075s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https

Nmap done: 1 IP address (1 host up) scanned in 23.05 seconds
```

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

nmap 192.168.1.1 192.168.1.2 192.168.1.3



```
prav@prav-VirtualBox:~$ nmap 192.168.1.1 192.168.1.2 192.168.1.3
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-23 23:30 IST
Nmap scan report for 192.168.1.1
Host is up (0.0077s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https

Nmap done: 3 IP addresses (1 host up) scanned in 6.21 seconds
prav@prav-VirtualBox:~$
```

Task 7 a): Netcat as Chat tool

a) Intra system communication (Using 2 terminals in the same system)

Step 1: Open a terminal (Ctrl+Alt+T). This will act as a Server.

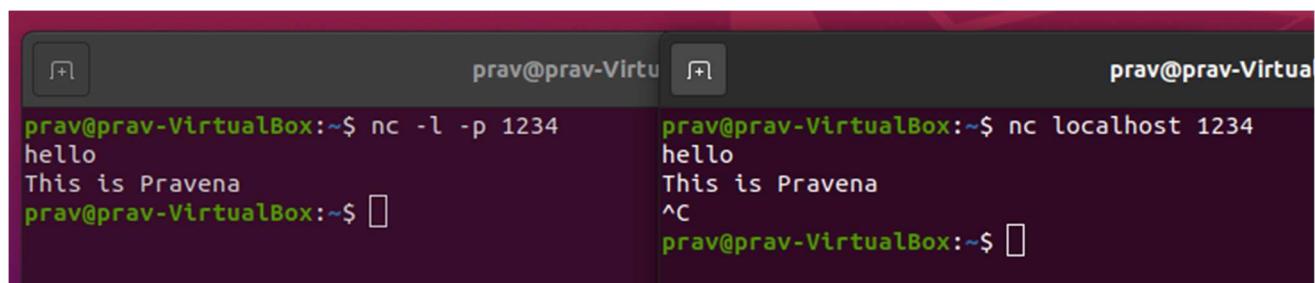
Step 2: Type **nc -l any_portnum** (For eg., nc -l 1234). It will go to listening mode.

Step 3: Open another terminal and this will act as a client.

Step 4: Type nc <your-system-ip-address> portnum

Note: portnum should be same in both the terminals (for eg., nc 10.0.2.8 1234)

Step 5: Type anything in client will appear in server



The screenshot shows two terminal windows side-by-side. Both windows have a dark background and light-colored text. The left window is titled "prav@prav-Virtua" and contains the following text:
prav@prav-VirtualBox:~\$ nc -l -p 1234
hello
This is Pravena
prav@prav-VirtualBox:~\$

The right window is also titled "prav@prav-Virtua" and contains the following text:
prav@prav-VirtualBox:~\$ nc localhost 1234
hello
This is Pravena
^C
prav@prav-VirtualBox:~\$