COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

Task 1: Network Setup

Testing

The client is able to ping the server-router (10.9.0.11).

```
CS280_CLIENT_9.5 :/# ping server-router
PING server-router (10.9.0.11) 56(84) bytes of data.
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=1 ttl=64 time=0.097 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.113 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=3 ttl=64 time=0.140 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=5 ttl=64 time=0.557 ms
^Z
[1]+  Stopped                 ping server-router
CS280_CLIENT_9.5 :/# ping server-router
```

The server router is able to ping host in the 192.168.60.0/24 network.

```
CS280_srouter :/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.102 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.088 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.067 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.110 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=64 time=0.071 ms
^Z
[1]+  Stopped                 ping 192.168.60.5
CS280_srouter :/#
```

The client is unable to connect to any host on the private network (192.168.60.0/24). VPN tunnel is needed to establish the tunnel and connect to a host on the private network.

```
CS280_CLIENT_9.5 :/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^Z
[3]+  Stopped                 ping 192.168.60.5
CS280_CLIENT_9.5 :/#
```

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

When the client pings the server, the client (10.9.0.5) sends ICMP packets to the server (10.9.0.11). The corresponding echo request response packets are shown in tcpdump .

```
CS280_CLIENT_9.5 :/# ping server-router
PING server-router (10.9.0.11) 56(84) bytes of data.
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=1 ttl=64 time=0.062 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.145 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=3 ttl=64 time=0.111 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=4 ttl=64 time=0.239 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=5 ttl=64 time=0.103 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=6 ttl=64 time=0.444 ms
^Z
[4]+  Stopped                 ping server-router
```

```
CS280_srouter :/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
12:28:44.549425 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 21, seq 1, length 64
12:28:44.549447 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 21, seq 1, length 64
12:28:45.567693 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 21, seq 2, length 64
12:28:45.567777 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 21, seq 2, length 64
12:28:46.591168 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 21, seq 3, length 64
12:28:46.591204 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 21, seq 3, length 64
12:28:47.615042 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 21, seq 4, length 64
12:28:47.615179 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 21, seq 4, length 64
12:28:48.639502 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 21, seq 5, length 64
12:28:48.639532 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 21, seq 5, length 64
12:28:49.663449 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 21, seq 6, length 64
12:28:49.663731 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 21, seq 6, length 64
12:28:49.732264 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
12:28:49.732485 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
12:28:49.732501 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
12:28:49.732503 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
^Z
[2]+  Stopped                 tcpdump -i eth0 -n
CS280_srouter :/#
```

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

Task 2: Create and Configure TUN Interface

Task 2.a: Name of the Interface

The tun0 interface is to be configured in this task. The corresponding IP address is however not attached to the tun0 interface as seen below-

```
CS280_CLIENT_9.5 :/# cd volumes
CS280_CLIENT_9.5 :/# ls
tun.py
CS280_CLIENT_9.5 :/# chmod a+x tun.py
CS280_CLIENT_9.5 :/# ./tun.py &
[5] 36
CS280_CLIENT_9.5 :/# Interface Name: tun0
ip addr
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
2: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
CS280_CLIENT_9.5 :/#
```

In order to kill the tun.py file, the following commands are executed and the execution of that file is terminated.

```
CS280_CLIENT_9.5 :/# jobs
[1]   Stopped                 ping server-router  (wd: /)
[2]   Stopped                 ping server-router  (wd: /)
[3]-  Stopped                 ping 192.168.60.5  (wd: /)
[4]+  Stopped                 ping server-router  (wd: /)
[5]   Running                 ./tun.py &
CS280_CLIENT_9.5 :/# kill %5
CS280_CLIENT_9.5 :/# jobs
[1]   Stopped                 ping server-router  (wd: /)
[2]   Stopped                 ping server-router  (wd: /)
[3]-  Stopped                 ping 192.168.60.5  (wd: /)
[4]+  Stopped                 ping server-router  (wd: /)
[5]   Terminated              ./tun.py
CS280_CLIENT_9.5 :/#
```

In this task, tun is replaced by CS280 and the same .py file is executed. The resultant interface name is CS2800. There is no IP address bound to this interface yet.

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

```
CS280_CLIENT_9.5 :/# chmod a+x tun.py
CS280_CLIENT_9.5 :/# ./tun.py &
[5] 57
CS280_CLIENT_9.5 :/# Interface Name: CS2800
ip addr
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
5: CS2800: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
CS280_CLIENT_9.5 :/#
```

Task 2.b: Set up the TUN Interface

In this task, we bind an IP address to the tun interface we have created.

```
CS280_CLIENT_9.5 :/# ip addr add 192.168.53.99/24 dev CS2800
CS280_CLIENT_9.5 :/# ip link set dev CS2800 up
CS280_CLIENT_9.5 :/#
```

When the interfaces on the system are checked now, we see that CS2800 interface has an IP address bound to it.

```
ip addr
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
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
14: CS2800: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global CS2800
       valid_lft forever preferred_lft forever
```

Task 2.c: Read from the TUN Interface

When the tun0 interface pings another system on the same subnet (192.168.53.0/24), then ICMP request packets are sent out. Because 192.168.53.5 is not a valid live host, no ICMP response packets are received. The corresponding packet only contains IP header and ICMP header as seen below-

```
CS280_CLIENT_9.5 :/# ping 192.168.53.5
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
^Z
[7]+  Stopped                 ping 192.168.53.5
CS280_CLIENT_9.5 :/#
```

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

When the interface tries to ping a host outside its network, we see that the ping is not successful. The tun interface cannot directly connect to the host system as it not a system/ device inside its subnet.

```
CS280_CLIENT_9.5 :/#ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5093ms

CS280_CLIENT_9.5 :/#
```

Task 2.d: Write to the TUN Interface

In this task, when an ICMP echo request is sent out from the client to 192.168.53.5, a spoofed ICMP response packet is sent back to the client. As seen below, the ICMP request packet is directed from 192.168.53.99 to 53.5 and the response is sent from 53.5 back to 53.99 (spoofed response, 53.5 isn't a valid host and hence, it cannot reply).

```
CS280_CLIENT_9.5 :/# chmod a+x tun1.py
CS280_CLIENT_9.5 :/# ./tun1.py &
[9] 129
CS280_CLIENT_9.5 :/# Interface Name: CS2800
ip addr
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
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
14: CS2800: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global CS2800
       valid_lft forever preferred_lft forever
CS280_CLIENT_9.5 :/#
CS280_CLIENT_9.5 :/#
CS280_CLIENT_9.5 :/# ping 192.168.53.5
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
CS2800: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=1 ttl=99 time=1.61 ms
CS2800: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=2 ttl=99 time=2.15 ms
CS2800: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=3 ttl=99 time=1.99 ms
CS2800: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=4 ttl=99 time=3.02 ms
CS2800: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=5 ttl=99 time=1.51 ms
CS2800: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
64 bytes from 192.168.53.5: icmp_seq=6 ttl=99 time=2.05 ms
^Z
[10]+  Stopped                 ping 192.168.53.5
```

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

Task 3: Send the IP Packet to VPN Server Through a Tunnel

A tunnel is established between the client and server by creating a tun interface at both the client and the server.

```
CS280_CLIENT_9.5 :/# chmod a+x tun_client.py
CS280_CLIENT_9.5 :/# ./tun_client.py &
[11] 185
CS280_CLIENT_9.5 :/# Interface Name: CS2800
ip addr
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
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
19: CS2800: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global CS2800
       valid_lft forever preferred_lft forever
```

When the client pings 192.168.53.5, then the ICMP packets get routed through the tunnel to the server. However, these packets do not get forwarded to any valid host as no such host exists.

```
CS280_CLIENT_9.5 :/# ping 192.168.53.5
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
^Z
[12]+  Stopped                 ping 192.168.53.5
```

However, when 60.5 is pinged, we see that the client is able to send out ICMP request packets to that host. This means that the client is able to use the tunnel to send packets to the server. However, these packets aren't routed to 192.168.60.5 as the server is not configured to route these packets to that host. Therefore, all these packets are dropped.

```
CS280_CLIENT_9.5 :/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
^Z
[13]+  Stopped                 ping 192.168.60.5
```

On the client, packets generated by the client (10.9.0.5) are redirected to 0.0.0.0 at port 9090. This packet gets routed to the tun0 interface of the client. During the tunnelling, the IP address (of the enclosing packet) of the source is 192.168.53.99 and destination IP is 192.168.53.5 and 192.168.60.5 respectively. Those details are shown at server.

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

```
CS280_srouter :/# ./tun_server.py
10.9.0.5:39793 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:39793 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:39793 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:39793 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:39793 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:39793 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
^Z
[3]+  Stopped                 ./tun_server.py
```

Even though the server receives both the ICMP packets, both are dropped as 53.5 does not exist and 60.5 routing is not present at the server end.

The three networks present are displayed when ip route is executed.

```
CS280_CLIENT_9.5 :/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev CS2800 proto kernel scope link src 192.168.53.99
192.168.60.0/24 dev CS2800 scope link
CS280_CLIENT_9.5 :/#
```

The first interface is 10.9.0.0/24 (that of the client and the server). The second interface is that used by the tun0 interfaces. The third interface is where the host (of the private network) is located.

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

Task 4: Set Up the VPN Server

In this task, the server is configured so that the original packet (enclosed within the packet during the tunnelling) can be routed/ directed to 60.5 host in the 192.168.60.0/24 private network. When the ping command is initiated at the client, we see that now the ICMP request packets are routed to 192.168.60.5's system after tunnelling.

```
CS280_CLIENT_9.5 :/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
^Z
[15]+  Stopped                 ping 192.168.60.5
CS280_CLIENT_9.5 :/#
```

On the server router, we observe that the original ICMP packet's source IP is 10.9.0.5 and the destination IP is 0.0.0.0.

Once inside the tun interface, the encompassing packet (that encloses the ICMP packet) has a source IP of 192.168.53.99 (that of the tun interface) and destination Ip of 192.168.60.5 (that of the host to which the packet has to be routed to).

```
CS280_srouter :/# ./tun_server1.py
Interface Name: CS2800
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:40125 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.5
^Z
[3]+  Stopped                 ./tun_server1.py
CS280_srouter :/#
```

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

When tcpdump is executed on the host (60.5), we see the incoming ICMP packets with source and destination IP as shown below. Host 60.5 responds to the ICMP requests with ICMP response packets. However, since the server is not configured to route the packets generated by Host 60.5 back to Client 9.5, the client does not receive any response.

```
CS280_HOST_60.5 :/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
13:32:03.788762 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 233, seq 1, length 64
13:32:03.788798 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 233, seq 1, length 64
13:32:04.801850 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 233, seq 2, length 64
13:32:04.801873 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 233, seq 2, length 64
13:32:05.824275 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 233, seq 3, length 64
13:32:05.824302 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 233, seq 3, length 64
13:32:06.849462 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 233, seq 4, length 64
13:32:06.849487 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 233, seq 4, length 64
13:32:07.879234 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 233, seq 5, length 64
13:32:07.879260 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 233, seq 5, length 64
13:32:08.896416 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 233, seq 6, length 64
13:32:08.896453 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 233, seq 6, length 64
```

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280

5th Semester E section

Task 5: Handling Traffic in Both Directions

In this task, the VPN tunnel created is made bi-directional so that the client can connect to the remote host in the private network and communicate with it.

Now, when the client pings the host 60.5, we see that the client receives an ICMP response packet for every ICMP request packet it sends out. Ping is successful and therefore, the output is as shown below-

```
CS280_CLIENT2_9.5 :/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=7.70 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=3.43 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=2.66 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=1.41 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=1.86 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=2.22 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=2.04 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=2.90 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=2.59 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=2.47 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=2.00 ms
```

When seen on the client, we see that the original ICMP packet passes through the tun interface where it is enclosed within a UDP packet and sent out of the system. ICMP Packets encompassed within the UDP packets are received on this system at the socket and the corresponding IP addresses for the same are shown below –

```
CS280_CLIENT_9.5 :/# ./tun_client_select.py
Interface Name: CS2800
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.53.99 --> 192.168.60.5
```

<u>**COMPUTER NETWORK SECURITY -09**</u>

Name: Pavan R Kashyap SRN: PES1UG20CS280

5th Semester E section

At the server router, the corresponding ICMP packets (encompassed within UDP packets) are obtained via the socket and then redirected to the host to which it is intended. Reply packets obtained from that host pass through the tun interface where they get encompassed within a UDP packet. This is then sent across the tunnel to the client. The corresponding IP addresses are shown below

```
CS280_srouter :/# chmod a+x tun_server_select.py
[3]   Terminated              ./tun_server1.py
CS280_srouter :/# ./tun_server_select.py
Interface Name: CS2800
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
```

UDP packets seen below are the packets that are exchanged across the tunnel. These UDP packets contain the ICMP packets as their payload. VPN client and server exchange these packets accordingly (for every ICMP reply and ICMP response packet).

ICMP request packets originate at the client. They are directed to the tun interface which then puts the encompassed packet into the VPN tunnel. Once the original ICMP packet is retrieved at the server (after passing through the socket first and the tun interface next), it is directed to host 60.5.

The ICMP response packet generated by Host 60.5 reaches the tun interface of the server (from where it is encompassed in a UDP packet and sent back to the client).

14	2022-11-12 08:4…	192.168.53.99	192.168.60.5	ICMP	100 Echo (ping) request	id=0x0129, seq=2/512, ttl=63 (no respons…
15	2022-11-12 08:4…	192.168.53.99	192.168.60.5	ICMP	100 Echo (ping) request	id=0x0129, seq=2/512, ttl=63 (reply in 1…
16	2022-11-12 08:4…	192.168.60.5	192.168.53.99	ICMP	100 Echo (ping) reply	id=0x0129, seq=2/512, ttl=64 (request in…
17	2022-11-12 08:4…	192.168.60.5	192.168.53.99	ICMP	100 Echo (ping) reply	id=0x0129, seq=2/512, ttl=64
18	2022-11-12 08:4…	10.9.0.11	10.9.0.5	UDP	128 9090 → 49656 Len=84	
19	2022-11-12 08:4…	10.9.0.11	10.9.0.5	UDP	128 9090 → 49656 Len=84	
20	2022-11-12 08:4…	10.9.0.5	10.9.0.11	UDP	128 49656 → 9090 Len=84	
21	2022-11-12 08:4…	10.9.0.5	10.9.0.11	UDP	128 49656 → 9090 Len=84	
22	2022-11-12 08:4…	192.168.53.99	192.168.60.5	ICMP	100 Echo (ping) request	id=0x0129, seq=3/768, ttl=63 (no respons…
23	2022-11-12 08:4…	192.168.53.99	192.168.60.5	ICMP	100 Echo (ping) request	id=0x0129, seq=3/768, ttl=63 (reply in 2…
24	2022-11-12 08:4…	192.168.60.5	192.168.53.99	ICMP	100 Echo (ping) reply	id=0x0129, seq=3/768, ttl=64 (request in…
25	2022-11-12 08:4…	192.168.60.5	192.168.53.99	ICMP	100 Echo (ping) reply	id=0x0129, seq=3/768, ttl=64
26	2022-11-12 08:4…	10.9.0.11	10.9.0.5	UDP	128 9090 → 49656 Len=84	
27	2022-11-12 08:4…	10.9.0.11	10.9.0.5	UDP	128 9090 → 49656 Len=84	
28	2022-11-12 08:4…	10.9.0.5	10.9.0.11	UDP	128 49656 → 9090 Len=84	
29	2022-11-12 08:4…	10.9.0.5	10.9.0.11	UDP	128 49656 → 9090 Len=84	

The Wireshark output displays the same thing explained above

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

When the client tries to telnet, the client is able to connect to host 60.5 as shown below -

```
CS280_CLIENT2_9.5 :/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
bea1f290d630 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Nov 12 13:54:34 UTC 2022 on pts/3
seed@bea1f290d630:~$ exit
logout
Connection closed by foreign host.
CS280_CLIENT2_9.5 :#
```

During telnet, TCP packets are sent across the channel. The process through which client connects to the server is pretty much the same as how it happens with ICMP. The result is as shown below

10	2022-11-12 08:4…	10.9.0.5	10.9.0.11	UDP	96 49656 → 9090 Len=52
11	2022-11-12 08:4…	10.9.0.5	10.9.0.11	UDP	96 49656 → 9090 Len=52
12	2022-11-12 08:4…	192.168.53.99	192.168.60.5	TCP	68 43622 → 23 [ACK] Seq=2295711712 Ack=235546318 Win=64256 Len=0…
13	2022-11-12 08:4…	192.168.53.99	192.168.60.5	TCP	68 [TCP Dup ACK 12#1] 43622 → 23 [ACK] Seq=2295711712 Ack=235546…
14	2022-11-12 08:4…	10.9.0.5	10.9.0.11	UDP	120 49656 → 9090 Len=76
15	2022-11-12 08:4…	10.9.0.5	10.9.0.11	UDP	120 49656 → 9090 Len=76
16	2022-11-12 08:4…	192.168.53.99	192.168.60.5	TELNET	92 Telnet Data ...
17	2022-11-12 08:4…	192.168.53.99	192.168.60.5	TCP	92 [TCP Retransmission] 43622 → 23 [PSH, ACK] Seq=2295711712 Ack…
18	2022-11-12 08:4…	192.168.60.5	192.168.53.99	TCP	68 23 → 43622 [ACK] Seq=235546318 Ack=2295711736 Win=65152 Len=0…
19	2022-11-12 08:4…	192.168.60.5	192.168.53.99	TCP	68 [TCP Dup ACK 18#1] 23 → 43622 [ACK] Seq=235546318 Ack=2295711…
20	2022-11-12 08:4…	10.9.0.11	10.9.0.5	UDP	96 9090 → 49656 Len=52
21	2022-11-12 08:4…	10.9.0.11	10.9.0.5	UDP	96 9090 → 49656 Len=52

After successfully establishing telnet:

254	2022-11-12 08:5…	192.168.60.5	192.168.53.99	TELNET	89 Telnet Data ...
255	2022-11-12 08:5…	192.168.60.5	192.168.53.99	TCP	89 [TCP Retransmission] 23 → 43626 [PSH, ACK] Seq=3678722259 Ack…
256	2022-11-12 08:5…	10.9.0.11	10.9.0.5	UDP	117 9090 → 41646 Len=73
257	2022-11-12 08:5…	10.9.0.11	10.9.0.5	UDP	117 9090 → 41646 Len=73
258	2022-11-12 08:5…	10.9.0.5	10.9.0.11	UDP	96 41646 → 9090 Len=52
259	2022-11-12 08:5…	10.9.0.5	10.9.0.11	UDP	96 41646 → 9090 Len=52
260	2022-11-12 08:5…	192.168.53.99	192.168.60.5	TCP	68 43626 → 23 [ACK] Seq=3252627638 Ack=3678722280 Win=64128 Len=…
261	2022-11-12 08:5…	192.168.53.99	192.168.60.5	TCP	68 [TCP Dup ACK 260#1] 43626 → 23 [ACK] Seq=3252627638 Ack=36787…
262	2022-11-12 08:5…	10.0.2.7	192.168.29.1	DNS	88 Standard query 0x8480 PTR 99.53.168.192.in-addr.arpa
263	2022-11-12 08:5…	02:42:0a:09:00:0b		ARP	44 Who has 10.9.0.5? Tell 10.9.0.11
264	2022-11-12 08:5…	02:42:0a:09:00:0b		ARP	44 Who has 10.9.0.5? Tell 10.9.0.11
265	2022-11-12 08:5…	02:42:0a:09:00:05		ARP	44 10.9.0.5 is at 02:42:0a:09:00:05
266	2022-11-12 08:5…	02:42:0a:09:00:05		ARP	44 10.9.0.5 is at 02:42:0a:09:00:05

```
▸ Frame 254: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 192.168.60.5, Dst: 192.168.53.99
▾ Transmission Control Protocol, Src Port: 23, Dst Port: 43626, Seq: 3678722259, Ack: 3252627638, Len: 21
    Source Port: 23
    Destination Port: 43626
    [Stream index: 0]
    [TCP Segment Len: 21]
    Sequence number: 3678722259
    [Next sequence number: 3678722280]
    Acknowledgment number: 3252627638
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
```

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

The client sends data via a TCP packet. The VPN client encompasses the TCP packet inside the UDP packet and sends that across the VPN tunnel. The VPN server, obtains the same TCP packet and redirects it to the intended host.

The Telnet packet sent by the host reaches the client the same way. The IP addresses obtained are the same as what was obtained in ICMP.

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

Task 6: Tunnel-Breaking Experiment

In this task, we attempt to break a VPN tunnel and see the behaviour of telnet between the client and the host system.

The tunnel is established b/w the client and the server. Client telnets to Host 60.5 and the resultant packets involved in this process are accordingly shown at the client and server. At the client, the packets first pass through the tun (to reach the server). Packets received pass through the socket (to reach the client).

```
CS280_CLIENT_9.5 :/# ./tun_client_select.py
Interface Name: CS2800
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
```

```
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
^CTraceback (most recent call last):
  File "./tun_server_select.py", line 38, in <module>
    ready, _, _ = select.select(fds, [], [])
KeyboardInterrupt

CS280_srouter :/#
```

At the server, incoming packets at the VPN tunnel pass through the socket to reach the server. Incoming packets from the host pass through the tun interface to be routed/sent across the VPN channel.

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

When the server connection is broken, the VPN tunnel is broken. Any packets sent from the client do not reach the server; any packets sent from the host do not enter the VPN channel.

TCP packets can no more be exchanged between the two hosts; therefore, when we try to type anything, the cursor is stationary and does not move. The TCP connection between the two systems does not close though. The result can be seen below-

```
CS280_CLIENT_9.5 :/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
bea1f290d630 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Mon Nov 14 05:08:20 UTC 2022 from 192.168.53.99 on pts/3
seed@bea1f290d630:~$
```

When the VPN tunnel is brought back up, TCP packets being generated at the client can route to the host. This causes the data packets from the client to reach the server and back to client (where they get displayed on screen). There is an initial lag when the user types characters. This is because the terminal displays those characters only after it has received a Telnet packet from the host.

After a certain amount of time, all the entered characters are displayed on the screen. The server is running and the packet movement through its socket and tun interface are displayed like before.

```
^CTraceback (most recent call last):
  File "./tun_server_select.py", line 38, in <module>
    ready, _, _ = select.select(fds, [], [])
KeyboardInterrupt

CS280_srouter :/# ./tun_server_select.py
Interface Name: CS2800
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From tun    ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
```

COMPUTER NETWORK SECURITY -09

Name: Pavan R Kashyap SRN: PES1UG20CS280
5th Semester E section

```
CS280_CLIENT_9.5 :/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
bea1f290d630 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Mon Nov 14 05:08:20 UTC 2022 from 192.168.53.99 on pts/3
seed@bea1f290d630:~$ seljgboewboggiewSaeuw0ienwienviewivc
```

The characters entered are shown above.