

COMPUTER NETWORK SECURITY LABORATORY

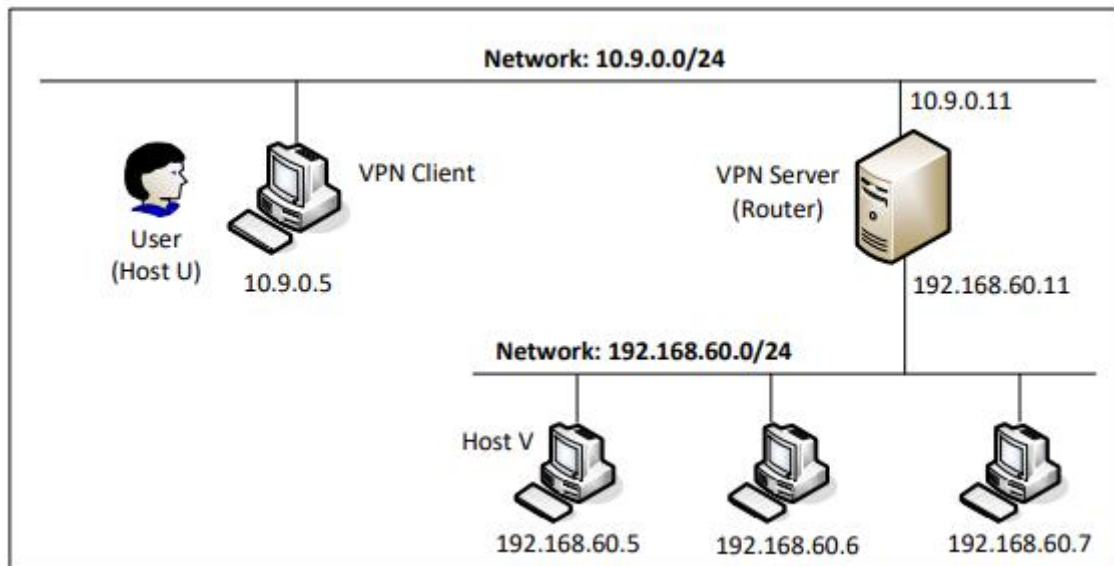
NAME : PREM SAGAR J S

SRN : PES1UG20CS825

SEC : H

VPN Tunneling Lab

Lab Setup



Task 1: Network Setup

Testing :

conducting the following testings to ensure that the lab environment is set up correctly:

- Host U - 10.9.0.5 can communicate with VPN Server (server-router)

On Client-10.9.0.5

Command:

ping server-router

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/
$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=0.250 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.213 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=3 ttl=64 time=0.224 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=4 ttl=64 time=0.236 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=5 ttl=64 time=0.284 ms
^C
--- server-router ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4092ms
rtt min/avg/max/mdev = 0.213/0.241/0.284/0.024 ms
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/
$
```

- VPN Server (server-router) can communicate with Host V (host-192.168.60.5)

On server-router

Command :

ping 192.168.60.5

```
Server-Router:PES1UGCS825:Prem Sagar J S:/
$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=0.277 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.233 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.312 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.226 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=64 time=0.321 ms
^C
--- 192.168.60.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4051ms
rtt min/avg/max/mdev = 0.226/0.273/0.321/0.039 ms
Server-Router:PES1UGCS825:Prem Sagar J S:/
$
```

- Host U (Client - 10.9.0.5) should not be able to communicate with Host V (host 192.168.60.5)

On Client 10.9.0.5

Command:

ping 192.168.60.5

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/
$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 ---
14 packets transmitted, 0 received, 100% packet loss, time 13312ms

Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/
$
```

- Run tcpdump on the router, and sniff the traffic on each of the networks. Show that you can capture packets.

On server-router run -

Command:

tcpdump -i eth0 -n

```
Server-Router:PES1UGCS825:Prem Sagar J S:/
$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
09:25:18.246616 IP6 fe80::42:f8ff:fe0a:1039.5353 > ff02::fb.5353: 0 [9q] PTR (QM)? _nfs._tcp.local
. PTR (QM)? _ipp._tcp.local. PTR (QM)? _ipps._tcp.local. PTR (QM)? _ftp._tcp.local. PTR (QM)? _web
dav._tcp.local. PTR (QM)? _webdavs._tcp.local. PTR (QM)? _sftp-ssh._tcp.local. PTR (QM)? _smb._tcp
.local. PTR (QM)? afpovertcp._tcp.local. (141)
```

```
.local. PTR (QM)?_afpovertcp.tcp.local. (141)
09:25:28.111714 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 1, length 64
09:25:28.111841 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 1, length 64
09:25:29.113925 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 2, length 64
09:25:29.113983 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 2, length 64
09:25:30.145075 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 3, length 64
09:25:30.145359 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 3, length 64
09:25:31.172799 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 4, length 64
09:25:31.173014 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 4, length 64
09:25:32.192903 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 17, seq 5, length 64
09:25:32.193015 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 17, seq 5, length 64
09:25:33.312410 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
09:25:33.312599 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
09:25:33.312614 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
09:25:33.312626 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
^C
15 packets captured
15 packets received by filter
0 packets dropped by kernel
Server-Router:PES1UGCS825:Prem Sagar J S:/
$
```

On Client - 10.9.0.5

Command:

ping server-router

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/
$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=1.04 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.218 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=3 ttl=64 time=0.499 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=4 ttl=64 time=0.227 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=5 ttl=64 time=0.215 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=6 ttl=64 time=0.300 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=7 ttl=64 time=0.228 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=8 ttl=64 time=0.231 ms
64 bytes from server-router.net-10.9.0.0 (10.9.0.11): icmp_seq=9 ttl=64 time=0.297 ms
^C
--- server-router ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8303ms
rtt min/avg/max/mdev = 0.215/0.362/1.044/0.255 ms
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/
$
```

Task 2: Create and Configure TUN Interface

The objective of this task is to get familiar with the TUN technology. We will conduct several experiments to learn the technical details of the TUN interface. We will use the following Python program as the basis for the experiments, and we will modify this base code throughout this lab.

Task 2.a: Name of the Interface

We will run the tun.py program on Host U (Client-10.9.0.5). Make the above tun.py program executable, and run it using the root privilege.

On Client - 10.9.0.5 (change directory to ./volumes)

Command:


```
# chmod a+x tun.py
# ./tun.py &
# ip addr
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$chmod a+x tun.py
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$./tun.py &
[1] 26
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$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
18: eth0@if19: <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
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

Kill the earlier Tunnel Process -

On Client - 10.9.0.5

Command:

```
# kill %1
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$kill %1
```

Now run the following commands again and see the new tunnel interface, it should be “SRNO”

On Client - 10.9.0.5

Command:

```
# chmod a+x tun.py
# ./tun.py &
# ip addr
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$chmod a+x tun.py
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$./tun.py &
[1] 34
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$Interface Name: CS8250
$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
3: CS8250: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
18: eth0@if19: <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
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

Task 2.b: Set up the TUN Interface

point, the TUN interface is not usable, because it has not been configured yet. There are two things that we need to do before the interface can be used. First, we need to assign an IP address to it. Second, we need to bring up the interface, because the interface is still in the down state.

On Client - 10.9.0.5

Command:

```
# ip addr add 192.168.53.99/24 dev CS8250
# ip link set dev CS8250 up
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$ip addr add 192.168.53.99/24 dev CS8250
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$ip link set dev CS8250 up
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

Task 2.c: Read from the TUN Interface

Kill the earlier Tunnel Process -

On Client - 10.9.0.5

Command:

```
# kill %1
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$kill %1
```

Set up the TUN Interface -

On Client - 10.9.0.5

Command:

```
# ip addr add 192.168.53.99/24 dev CS8250
# ip link set dev CS8250 up
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$ip addr add 192.168.53.99/24 dev CS8250
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$ip link set dev CS8250 up
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

Running the revised tun.py program -

On Client - 10.9.0.5

Command:

```
# ./tun.py &
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$./tun.py &
[1] 60
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$Interface Name: CS8250
```

On Host U, ping a host in the 192.168.53.0/24 network. What is printed out by the tun.py program? What has happened? Why?

=>we are read from the TUN interface. Whatever coming out from the TUN interface is an IP packet. We can cast the data received from the interface into a Scapy IP object, so it is printing out each field of the IP packet.

On Client - 10.9.0.5

Command:

```
# ping 192.168.53.5
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$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
^C
--- 192.168.53.5 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3072ms

Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

On Host U, ping a host in the internal network 192.168.60.0/24, Does tun.py print out anything? Why?

=> Im not able to ping any IP in the Internal network 192.168.60.0/24 and Its not printing anything at all.

On Client - 10.9.0.5

Command:

```
# ping 192.168.60.5
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$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 5116ms

Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

Kill the earlier Tunnel Process -

On Client - 10.9.0.5

Command:

```
# kill %1
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$kill %1
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

Task 2.d: Write to the TUN Interface

after getting a packet from the TUN interface, we construct a new packet based on the received packet. We then write the new packet to the TUN interface.

On Client - 10.9.0.5

Command:

```
# chmod a+x tun.py
# ./tun1.py &
# ip addr
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$./tun1.py &
[1] 78
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$Interface Name: CS8250
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
7: CS8250: <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 CS8250
        valid_lft forever preferred_lft forever
18: eth0@if19: <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
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

On Client - 10.9.0.5

Command:

```
# ping 192.168.53.5
```



```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$ping 192.168.53.5
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
CS8250: 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=8.98 ms
CS8250: 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=12.5 ms
CS8250: 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=8.99 ms
^C
--- 192.168.53.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 8.981/10.154/12.497/1.656 ms
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

Kill the earlier Tunnel Process -

On Client - 10.9.0.5

Command:

```
# kill %1
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$kill %1
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

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

we will put the IP packet received from the TUN interface into the UDP payload field of a new IP packet, and send it to another computer. Namely, we place the original packet inside a new packet. This is called IP tunneling.

On server-router run (change directory to ./volumes)

Command:

```
# chmod a+x tun_server.py
```

```
# ./tun_server.py
```

```
Server-Router:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$chmod a+x tun_server.py
Server-Router:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$./tun_server.py
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.60.5
```

Run tun_client.py on Client - 10.9.0.5

Command:

```
# chmod a+x tun_client.py
# ./tun_client.py &
# ip addr
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$chmod a+x tun_client.py
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$./tun_client.py &
[1] 84
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$Interface Name: CS8250
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
6: CS8250: <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 CS8250
        valid_lft forever preferred_lft forever
10: eth0@if11: <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
```

To test whether the tunnel works or not, ping any IP address belonging to the 192.168.53.0/24 network.

On Client - 10.9.0.5

Command:

```
# ping 192.168.53.5
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$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
^C
--- 192.168.53.5 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3095ms
```

Let us ping Host V, and see whether the ICMP packet is sent to VPN Server through the tunnel.

On Client - 10.9.0.5

Command:

```
# ping 192.168.60.5
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$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
^C
--- 192.168.60.5 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3078ms
```

ICMP packets are being sent to VPN Server through the tunnel.

To check the routing run the following command on Client - 10.9.0.5

Command:

```
# ip route
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$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 CS8250 proto kernel scope link src 192.168.53.99
192.168.60.0/24 dev CS8250 scope link
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

Task 4: Set Up the VPN Server

Feeding the packet to the kernel, so the kernel can route the packet towards its final destination. This needs to be done through a TUN interface.

On server-router run

Command:

```
# chmod a+x tun_server1.py
# ./tun_server1.py
```

```
Server-Router:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$chmod a+x tun_server1.py
Server-Router:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$./tun_server1.py
Interface Name: CS8250
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:57447 --> 0.0.0.0:9090
    Inside: 192.168.53.99 --> 192.168.60.5
```

On host-192.168.60.5 run -

Command:

```
# tcpdump -i eth0 -n
```

```
$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
18:11:56.671798 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 99, seq 1, length 64
18:11:56.672092 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 99, seq 1, length 64
18:11:57.698787 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 99, seq 2, length 64
18:11:57.698841 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 99, seq 2, length 64
18:11:58.731809 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 99, seq 3, length 64
18:11:58.732075 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 99, seq 3, length 64
18:11:59.746198 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 99, seq 4, length 64
18:11:59.746243 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 99, seq 4, length 64
18:12:01.792919 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
18:12:01.793025 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
18:12:01.793033 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
18:12:01.793037 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
^C
12 packets captured
12 packets received by filter
0 packets dropped by kernel
```

On Client - 10.9.0.5 run -

Command:

```
# ping 192.168.60.5
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$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
^C
--- 192.168.60.5 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3076ms
```

Kill the earlier Tunnel Process -

On Client - 10.9.0.5

Command:

```
# kill %1
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$jobs
[1]+  Running                  ./tun_client.py &
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$kill %1
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$
```

Task 5: Handling Traffic in Both Directions

Wireshark is Running in the background for this task, capturing the packets on the client interface.

On the server-router run -

Command:

```
# chmod a+x tun_server_select.py
# ./tun_server_select.py
```



```

Server-Router:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$chmod a+x tun_server_select.py
Server-Router:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$./tun_server_select.py
Interface Name: CS8250
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

```

On one terminal of Client - 10.9.0.5 run -

Command:

```

# chmod a+x tun_client_select.py
# ./tun_client_select.py

```

```

Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$chmod a+x tun_client_select.py
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$./tun_client_select.py
Interface Name: CS8250
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

```

On the other terminal of Client - 10.9.0.5 run -

Command:

```

# ping 192.168.60.5

```

```

Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$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=11.2 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=4.73 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=3.89 ms
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 3.890/6.612/11.223/3.277 ms

```

Wireshark Outup :

[SEED Labs] Capturing from br-0d12c45e6915							
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help							
Apply a display filter ... <Ctrl-/>							
No.	Time	Source	Destination	Protocol	Length	Info	
1	2022-11-14 13:2...	10.9.0.5	10.9.0.11	UDP	126	50967 → 9090	Len=84
2	2022-11-14 13:2...	10.9.0.11	10.9.0.5	UDP	126	9090 → 50967	Len=84
3	2022-11-14 13:2...	10.9.0.5	10.9.0.11	UDP	126	50967 → 9090	Len=84
4	2022-11-14 13:2...	10.9.0.11	10.9.0.5	UDP	126	9090 → 50967	Len=84
5	2022-11-14 13:2...	10.9.0.5	10.9.0.11	UDP	126	50967 → 9090	Len=84
6	2022-11-14 13:2...	10.9.0.11	10.9.0.5	UDP	126	9090 → 50967	Len=84
7	2022-11-14 13:2...	02:42:0a:09:00:0b	02:42:0a:09:00:05	ARP	42	Who has 10.9.0.5? Tell 10.9.0.11	
8	2022-11-14 13:2...	02:42:0a:09:00:05	02:42:0a:09:00:0b	ARP	42	Who has 10.9.0.11? Tell 10.9.0.5	
9	2022-11-14 13:2...	02:42:0a:09:00:05	02:42:0a:09:00:0b	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05	
10	2022-11-14 13:2...	02:42:0a:09:00:0b	02:42:0a:09:00:05	ARP	42	10.9.0.11 is at 02:42:0a:09:00:0b	

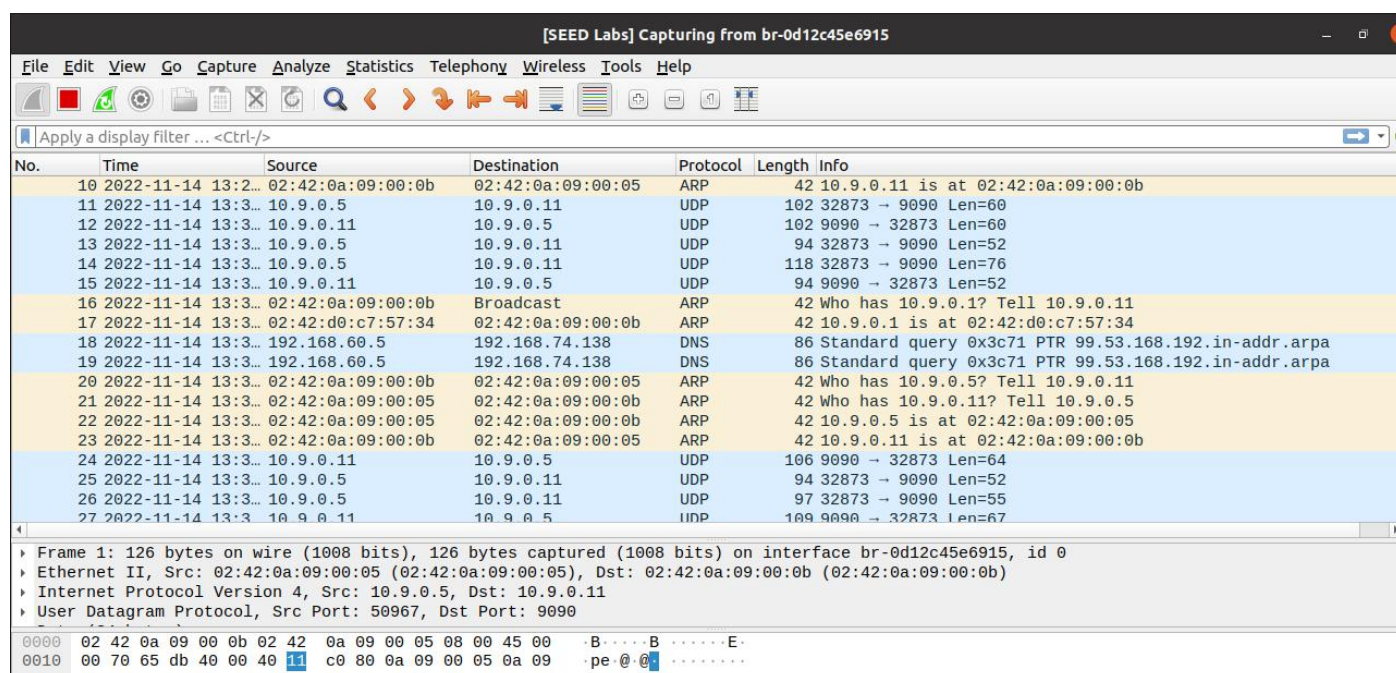
Now we Telnet into 192.168.60.5 - (same terminal as the one who've pinged from)

Command:

```
# telnet 192.168.60.5
```

```
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
44ebc2482b38 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

Wireshark Output :



No.	Time	Source	Destination	Protocol	Length	Info
10	2022-11-14 13:2...	02:42:0a:09:00:0b	02:42:0a:09:00:05	ARP	42	10.9.0.11 is at 02:42:0a:09:00:0b
11	2022-11-14 13:3...	10.9.0.5	10.9.0.11	UDP	102	32873 → 9090 Len=60
12	2022-11-14 13:3...	10.9.0.11	10.9.0.5	UDP	102	9090 → 32873 Len=60
13	2022-11-14 13:3...	10.9.0.5	10.9.0.11	UDP	94	32873 → 9090 Len=52
14	2022-11-14 13:3...	10.9.0.5	10.9.0.11	UDP	118	32873 → 9090 Len=76
15	2022-11-14 13:3...	10.9.0.11	10.9.0.5	UDP	94	9090 → 32873 Len=52
16	2022-11-14 13:3...	02:42:0a:09:00:0b	Broadcast	ARP	42	Who has 10.9.0.1? Tell 10.9.0.11
17	2022-11-14 13:3...	02:42:d0:c7:57:34	02:42:0a:09:00:0b	ARP	42	10.9.0.1 is at 02:42:d0:c7:57:34
18	2022-11-14 13:3...	192.168.60.5	192.168.74.138	DNS	86	Standard query 0x3c71 PTR 99.53.168.192.in-addr.arpa
19	2022-11-14 13:3...	192.168.60.5	192.168.74.138	DNS	86	Standard query 0x3c71 PTR 99.53.168.192.in-addr.arpa
20	2022-11-14 13:3...	02:42:0a:09:00:0b	02:42:0a:09:00:05	ARP	42	Who has 10.9.0.5? Tell 10.9.0.11
21	2022-11-14 13:3...	02:42:0a:09:00:05	02:42:0a:09:00:0b	ARP	42	Who has 10.9.0.11? Tell 10.9.0.5
22	2022-11-14 13:3...	02:42:0a:09:00:05	02:42:0a:09:00:0b	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
23	2022-11-14 13:3...	02:42:0a:09:00:0b	02:42:0a:09:00:05	ARP	42	10.9.0.11 is at 02:42:0a:09:00:0b
24	2022-11-14 13:3...	10.9.0.11	10.9.0.5	UDP	106	9090 → 32873 Len=64
25	2022-11-14 13:3...	10.9.0.5	10.9.0.11	UDP	94	32873 → 9090 Len=52
26	2022-11-14 13:3...	10.9.0.5	10.9.0.11	UDP	97	32873 → 9090 Len=55
27	2022-11-14 13:3...	10.9.0.11	10.9.0.5	UDP	109	9090 → 32873 Len=67

Frame 1: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface br-0d12c45e6915, id 0
Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:0b (02:42:0a:09:00:0b)
Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.11
User Datagram Protocol, Src Port: 50967, Dst Port: 9090

0000 02 42 0a 09 00 0b 02 42 0a 09 00 05 08 00 45 00 .B....B.....E.
0010 00 70 65 db 40 00 40 c0 00 0a 09 00 05 0a 09 .pe.@.@.....

The VPN connections are not Closed (server and client), as we require it for the next task as well.

Task 6: Tunnel-Breaking Experiment

On Host U (10.9.0.5), telnet to Host V (192.168.60.5) .

On Client -10.9.0.5

Command:

```
# telnet 192.168.60.5
```

```
seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x seed@VM: ~/.../La... x
Client-10.9.0.5:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
```

```
44ebc2482b38 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 18:31:25 UTC 2022 on pts/2

seed@44ebc2482b38:~\$ █

while keeping the telnet connection alive, we break the VPN tunnel by stopping the tun_server_select.py program - Ctrl + C in server-router

```
Server-Router:PES1UGCS825:Prem Sagar J S:/volumes/Codes
```

```
$/tun_server_select.py
```

```
Interface Name: CS8250
```

```
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 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 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 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
```

We then type something in the telnet window.

To restore this content, you can run the 'unminimize' command.

Last login: Mon Nov 14 18:31:25 UTC 2022 on pts/2

seed@44ebc2482b38:~\$ █

Do you see what you type? What happens to the TCP connection? Is the connection broken?

=> No I couldn't see what I have typed in the remote terminal and Yes, TCP connection is broken.

Let us now reconnect the VPN tunnel (do not wait for too long).

On the server-router run -

Command:

```
# ./tun_server_select.py
```



```

Server-Router:PES1UGCS825:Prem Sagar J S:/volumes/Codes
$./tun_server_select.py
Interface Name: CS8250
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

```

Once the tunnel is re-established, what is going to happen to the telnet connection?

=> Telnet connection was re-established and Im able to see what I have type while the connection was broken.

To restore this content, you can run the 'unminimize' command.
 Last login: Mon Nov 14 18:31:25 UTC 2022 on pts/2
 seed@44ebc2482b38:~\$ PES

Wireshark :

