

BÁO CÁO THỰC HÀNH

Môn học: NT140.P12.ANTT – An Toàn Mạng

Tên chủ đề: Lab 2 - VPN

GVHD: Tô Trọng Nghĩa

Nhóm: 6

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT140.P12.ANTT.2

STT	Họ và tên	MSSV	Email
1	Lại Quan Thiên	22521385	22521385@gm.uit.edu.vn
2	Mai Nguyễn Nam Phương	22521164	22521164@gm.uit.edu.vn
3	Hồ Diệp Huy	22520541	22520541@gm.uit.edu.vn
4	Đặng Đức Tài	22521270	22521270@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:

STT	Nội dung	Tình trạng	Trang
1	Task 1	100%	2 - 6
2	Task 2	100%	7 – 10
3	Task 3	100%	11 - 17
4	Task 4	100%	18 - 14
Điểm tự đánh giá			10/10

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

BÁO CÁO CHI TIẾT

1. TASK 1: Hãy thực hiện cấu hình route trên các máy ảo Gateway, Host U và Host V cho phép traffic VPN được điều hướng thông qua VPN tunnel.

- Trên cả 3 máy host U, host V và Gateway đều cần cho phép traffic giữa các mạng, ta dùng lệnh sau:

```
+ sudo iptables -A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT
+ sudo iptables -A FORWARD -i enp0s3 -o enp0s8 -j ACCEPT
```

- Trong đó: “enp0s8” và “enp0s3” là tên interface của hai vùng mạng mà ta cần chúng có thể traffic với nhau

```
[10/20/24] seed@server:~/.../vpn$ sudo iptables -A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT; sudo iptables -A FORWARD -i enp0s3 -o enp0s8 -j ACCEPT
[10/20/24] seed@server:~/.../vpn$
```

```
[10/18/24] seed@host-u:~$ sudo iptables -A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT
; sudo iptables -A FORWARD -i enp0s3 -o enp0s8 -j ACCEPT
[10/18/24] seed@host-u:~$
```

```
[10/18/24] seed@host-v:~$ sudo iptables -A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT
; sudo iptables -A FORWARD -i enp0s3 -o enp0s8 -j ACCEPT
[10/18/24] seed@host-v:~$
```

** Setting trên Server

- Set cấu hình địa chỉ IP 192.168.53.5 qua tun0 và kích hoạt nó: `sudo ifconfig tun0 192.168.53.5/24 up`

- Ngoài ra ta phải thêm một tuyến đường mới vào bảng định tuyến, chỉ ra rằng mọi lưu lượng hướng đến mạng 192.168.53.0/24 sẽ đi qua giao diện tun0: `sudo route add -net 192.168.60.0/24 tun0`

- Thực hiện kiểm tra cấu hình mạng của Server bằng lệnh: `ip route show` và `ifconfig -a`

```
[10/20/24] seed@server:~/.../vpn$ ifconfig -a
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        ether 02:42:51:38:5e:22 txqueuelen 0 (Ethernet)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.60.1 netmask 255.255.255.0 broadcast 192.168.60.255
        inet6 fe80::9a51:3346:30df:511d prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:8a:34:98 txqueuelen 1000 (Ethernet)
            RX packets 136 bytes 16256 (16.2 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 226 bytes 26759 (26.7 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.60.1 netmask 255.255.255.0 broadcast 192.168.60.255
        inet6 fe80::9a51:3346:30df:511d prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:ae:7b:71 txqueuelen 1000 (Ethernet)
            RX packets 57 bytes 5392 (5.3 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 148 bytes 19818 (19.8 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 426 bytes 37272 (37.2 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 426 bytes 37272 (37.2 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
tun0: flags=4385<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 192.168.53.1 netmask 255.255.255.0 destination 192.168.53.1
        inet6 fe80::c01d:b760:bff3:2354 prefixlen 64 scopeid 0x20<link>
            unspec 00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
            RX packets 57 bytes 4421 (4.4 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 41 bytes 3156 (3.1 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
[10/20/24] seed@server:~/.../vpn$ ip route show
default via 10.0.2.1 dev enp0s3 proto static metric 20100
default via 192.168.60.1 dev enp0s8 proto static metric 20101
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.5 metric 100
169.254.0.0/16 dev enp0s3 scope link metric 1000
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.53.0/24 dev tun0 proto kernel scope link src 192.168.53.1
192.168.60.0/24 dev enp0s8 proto kernel scope link src 192.168.60.1 metric 101
[10/20/24] seed@server:~/.../vpn$ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0          10.0.2.1       0.0.0.0        UG    20100   0      0 enp0s3
0.0.0.0          192.168.60.1   0.0.0.0        UG    20101   0      0 enp0s8
10.0.2.0          0.0.0.0        255.255.255.0  U     100    0      0 enp0s3
169.254.0.0       0.0.0.0        255.255.0.0    U     1000   0      0 enp0s3
172.17.0.0         0.0.0.0        255.255.0.0    U     0      0      0 docker0
192.168.53.0      0.0.0.0        255.255.255.0  U     0      0      0 tun0
192.168.60.0      0.0.0.0        255.255.255.0  U     101    0      0 enp0s8
[10/20/24] seed@server:~/.../vpn$
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	10.0.2.1	0.0.0.0	UG	20100	0	0	enp0s3
0.0.0.0	192.168.60.1	0.0.0.0	UG	20101	0	0	enp0s8
10.0.2.0	0.0.0.0	255.255.255.0	U	100	0	0	enp0s3
169.254.0.0	0.0.0.0	255.255.0.0	U	1000	0	0	enp0s3
172.17.0.0	0.0.0.0	255.255.0.0	U	0	0	0	docker0
192.168.53.0	0.0.0.0	255.255.255.0	U	0	0	0	tun0
192.168.60.0	0.0.0.0	255.255.255.0	U	101	0	0	enp0s8

** Setting trên Host U

- Đầu tiên ta phải thiết lập địa chỉ IP 192.168.53.5 với subnet mask /24 cho giao diện mạng ảo tun0 và kích hoạt nó với câu lệnh: `sudo ifconfig tun0 192.168.53.5/24 up`

- Ngoài ra ta phải thêm định tuyến mới cho mạng 192.168.60.0/24, tất cả lưu lượng đi đến mạng này sẽ được gửi qua giao diện tun0: `sudo route add -net 192.168.60.0/24 tun0`

- Thực hiện kiểm tra cấu hình mạng của Host U bằng lệnh `ip route show` và `ifconfig -a`

```
[10/20/24]seed@host-u:~/.../vpn$ ifconfig -a
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.255.0 broadcast 172.17.255.255
        ether 02:42:b2:10:f2:44 txqueuelen 0 (Ethernet)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.4 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::b14e:8fb2:20c6:9195 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:c9:60 txqueuelen 1000 (Ethernet)
            RX packets 270 bytes 40643 (40.6 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 196 bytes 20887 (20.8 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 18453 (18.4 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 191 bytes 18453 (18.4 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 192.168.53.5 netmask 255.255.255.0 destination 192.168.53.5
        inet6 fe80::945c:4000:efc8:73f7 prefixlen 64 scopeid 0x20<link>
            unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
            RX packets 36 bytes 2916 (2.9 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 56 bytes 4416 (4.4 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[10/20/24]seed@host-u:~/.../vpn$
```

```
[10/20/24]seed@host-u:~/.../vpn$ ip route show
default via 10.0.2.1 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.4 metric 100
169.254.0.0/16 dev enp0s3 scope link metric 1000
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.53.0/24 dev tun0 proto kernel scope link src 192.168.53.5
192.168.60.0/24 dev tun0 scope link
[10/20/24]seed@host-u:~/.../vpn$ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref  Use Iface
0.0.0.0          10.0.2.1       0.0.0.0        UG    100    0      0 enp0s3
10.0.2.0         0.0.0.0        255.255.255.0   U     100    0      0 enp0s3
169.254.0.0      0.0.0.0        255.255.0.0     U     1000   0      0 enp0s3
172.17.0.0        0.0.0.0        255.255.0.0     U     0      0      0 docker0
192.168.53.0     0.0.0.0        255.255.255.0   U     0      0      0 tun0
192.168.60.0     0.0.0.0        255.255.255.0   U     0      0      0 tun0
[10/20/24]seed@host-u:~/.../vpn$
```

** Setting trên Host V

- Đầu tiên ta thêm một tuyến mới vào bảng định tuyến, trong đó cho biết mọi lưu lượng mạng hướng đến mạng 10.0.2.0/24 sẽ đi qua cổng 192.168.60.1: `sudo ip route add 10.0.2.0/24 via 192.168.60.1`

- Tiếp tục ta sẽ thêm tuyến đường đi đến mạng 192.168.53.0/24 qua tun0 vào bảng định tuyến: `sudo ip route add -net 192.168.53.0/24 dev tun0`

- Thực hiện kiểm tra cấu hình mạng của Host V bằng lệnh `ip route show` và `ifconfig -a`

```
[10/20/24] seed@host-v:~/.../vpn$ ifconfig -a
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        ether 02:42:e0:8c:f7:7d txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.60.101 netmask 255.255.255.0 broadcast 192.168.60.255
        inet6 fe80::4ab4:5da2:6e37d0 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:cb:ac:05 txqueuelen 1000 (Ethernet)
            RX packets 647 bytes 53040 (53.0 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 159 bytes 15646 (15.6 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 5157 bytes 313633 (313.6 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 5157 bytes 313633 (313.6 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet6 fe80::1a8c:1f56:69bc:3f38 prefixlen 64 scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 6 bytes 288 (288.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[10/20/24] seed@host-v:~/.../vpn$ ip route show
default via 192.168.60.1 dev enp0s8 proto static metric 20100
10.0.2.0/24 via 192.168.60.1 dev enp0s8
169.254.0.0/16 dev enp0s8 scope link metric 1000
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.53.0/24 dev tun0 scope link
192.168.60.0/24 dev enp0s8 proto kernel scope link src 192.168.60.101 metric 100
[10/20/24] seed@host-v:~/.../vpn$
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	192.168.60.1	0.0.0.0	UG	20100	0	0	enp0s8
10.0.2.0	192.168.60.1	255.255.255.0	UG	0	0	0	enp0s8
169.254.0.0	0.0.0.0	255.255.0.0	U	1000	0	0	enp0s8
172.17.0.0	0.0.0.0	255.255.0.0	U	0	0	0	docker0
192.168.53.0	0.0.0.0	255.255.255.0	U	0	0	0	tun0
192.168.60.0	0.0.0.0	255.255.255.0	U	100	0	0	enp0s8

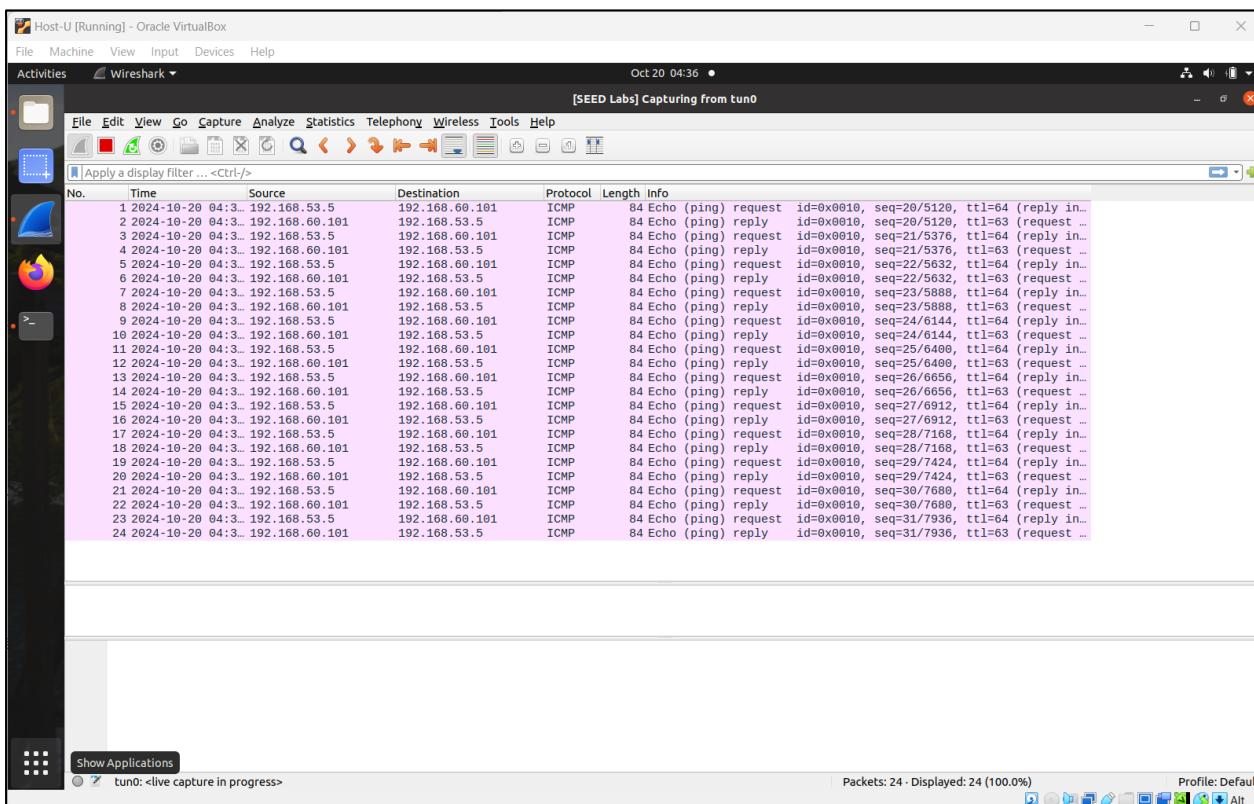
```
[10/20/24] seed@host-v:~/.../vpn$ ip route show
default via 192.168.60.1 dev enp0s8 proto static metric 20100
10.0.2.0/24 via 192.168.60.1 dev enp0s8
169.254.0.0/16 dev enp0s8 scope link metric 1000
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.53.0/24 dev tun0 scope link
192.168.60.0/24 dev enp0s8 proto kernel scope link src 192.168.60.101 metric 100
[10/20/24] seed@host-v:~/.../vpn$
```

2. TASK 2: Hãy thực hiện các lệnh ping, telnet và sử dụng Wireshark để bắt gói tin và chứng minh rằng hệ thống VPN đã hoạt động chính xác.

- Ping từ U sang V:

```
[10/20/24]seed@host-u:~/.../vpn$ ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
64 bytes from 192.168.60.101: icmp_seq=1 ttl=63 time=2.08 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 time=2.62 ms
64 bytes from 192.168.60.101: icmp_seq=3 ttl=63 time=2.51 ms
64 bytes from 192.168.60.101: icmp_seq=4 ttl=63 time=2.64 ms
64 bytes from 192.168.60.101: icmp_seq=5 ttl=63 time=1.72 ms
64 bytes from 192.168.60.101: icmp_seq=6 ttl=63 time=1.91 ms
64 bytes from 192.168.60.101: icmp_seq=7 ttl=63 time=2.38 ms
64 bytes from 192.168.60.101: icmp_seq=8 ttl=63 time=3.10 ms
64 bytes from 192.168.60.101: icmp_seq=9 ttl=63 time=2.03 ms
64 bytes from 192.168.60.101: icmp_seq=10 ttl=63 time=2.54 ms
64 bytes from 192.168.60.101: icmp_seq=11 ttl=63 time=2.15 ms
64 bytes from 192.168.60.101: icmp_seq=12 ttl=63 time=1.79 ms
64 bytes from 192.168.60.101: icmp_seq=13 ttl=63 time=2.79 ms
64 bytes from 192.168.60.101: icmp_seq=14 ttl=63 time=2.63 ms
64 bytes from 192.168.60.101: icmp_seq=15 ttl=63 time=1.90 ms
64 bytes from 192.168.60.101: icmp_seq=16 ttl=63 time=1.97 ms
64 bytes from 192.168.60.101: icmp_seq=17 ttl=63 time=1.96 ms
64 bytes from 192.168.60.101: icmp_seq=18 ttl=63 time=3.18 ms
64 bytes from 192.168.60.101: icmp_seq=19 ttl=63 time=2.12 ms
64 bytes from 192.168.60.101: icmp_seq=20 ttl=63 time=2.43 ms
```

- Wireshark bắt gói tin Ping từ U sang V:



- Telnet từ U sang V

[Host-U [Running] - Oracle VirtualBox

File Machine View Input Devices Help

Activities Terminal Oct 20 04:43

seed@host-u: ~/.../vpn\$ telnet 192.168.60.101

Trying 192.168.60.101...
Connected to 192.168.60.101.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
host-v login: sd
Password: Connection closed by foreign host.

[10/20/24]seed@host-u:~/.../vpn\$ telnet 192.168.60.101

Trying 192.168.60.101...
Connected to 192.168.60.101.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
host-v login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

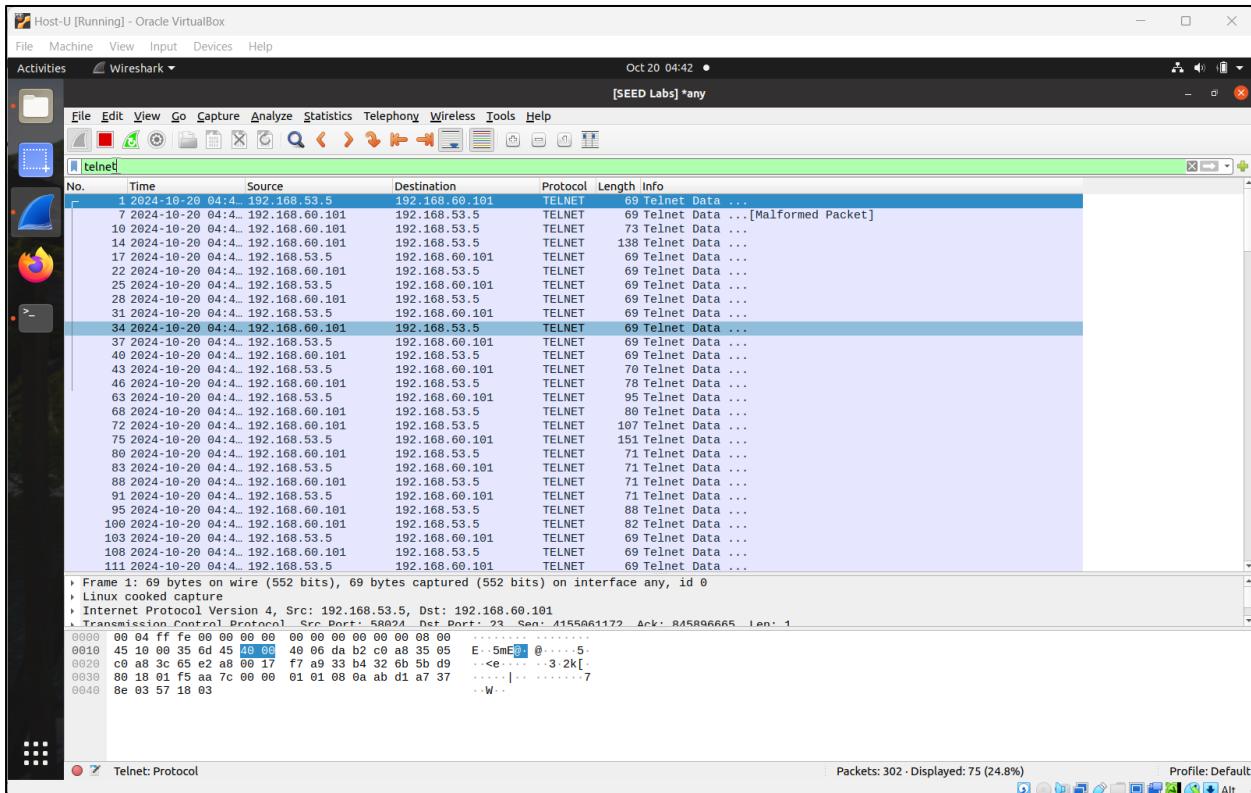
* Documentation: <https://help.ubuntu.com>
* Management: <https://landscape.canonical.com>
* Support: <https://ubuntu.com/advantage>

0 updates can be installed immediately.
0 of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Oct 20 04:42:19 EDT 2024 on pts/4

[10/20/24]seed@host-v:~\$

- Wireshark bắt gói tin Telnet từ U sang V





- Ping từ V sang U

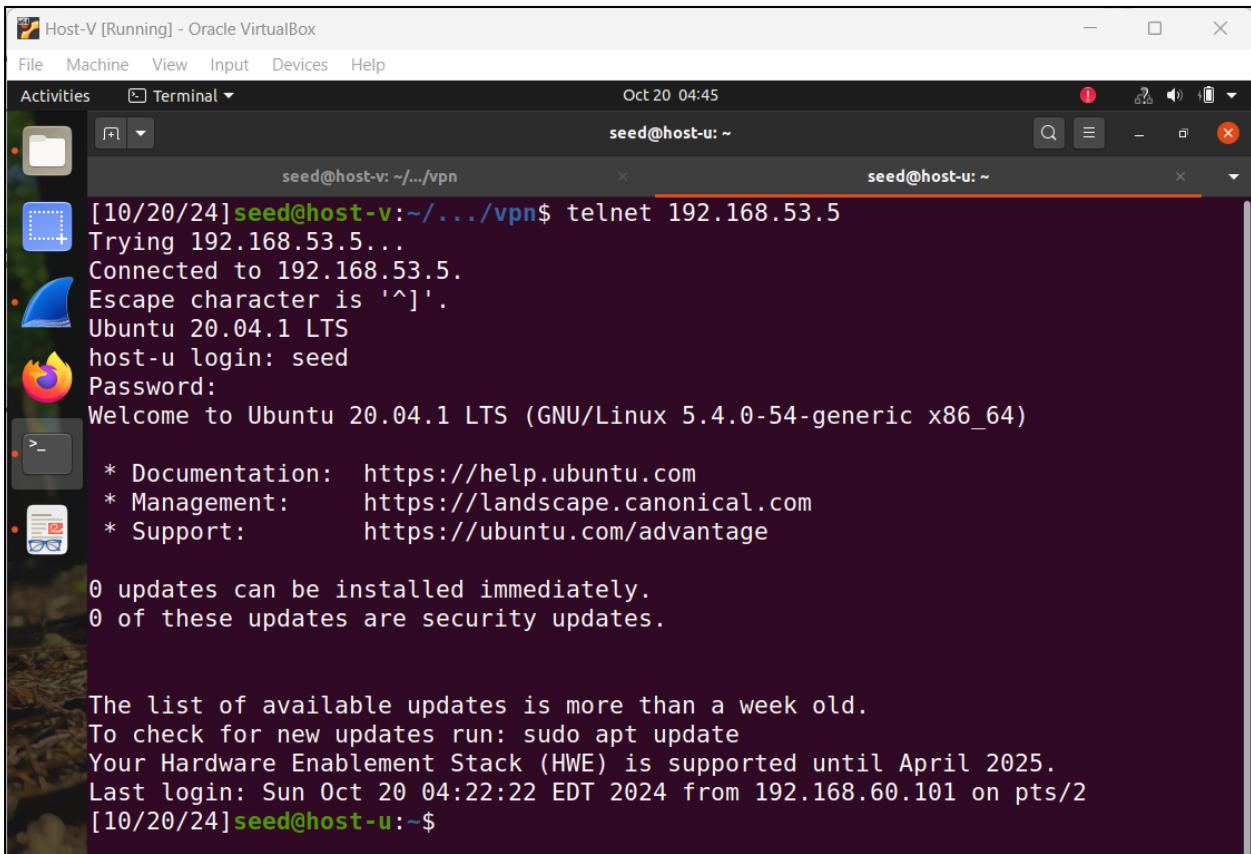
```
[10/20/24] seed@host-v:~/.../vpn$ ping 192.168.53.5
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
64 bytes from 192.168.53.5: icmp_seq=1 ttl=63 time=2.89 ms
64 bytes from 192.168.53.5: icmp_seq=2 ttl=63 time=1.95 ms
64 bytes from 192.168.53.5: icmp_seq=3 ttl=63 time=2.13 ms
64 bytes from 192.168.53.5: icmp_seq=4 ttl=63 time=2.51 ms
64 bytes from 192.168.53.5: icmp_seq=5 ttl=63 time=1.96 ms
64 bytes from 192.168.53.5: icmp_seq=6 ttl=63 time=2.31 ms
64 bytes from 192.168.53.5: icmp_seq=7 ttl=63 time=3.22 ms
64 bytes from 192.168.53.5: icmp_seq=8 ttl=63 time=3.10 ms
64 bytes from 192.168.53.5: icmp_seq=9 ttl=63 time=1.79 ms
64 bytes from 192.168.53.5: icmp_seq=10 ttl=63 time=3.10 ms
64 bytes from 192.168.53.5: icmp_seq=11 ttl=63 time=2.23 ms
64 bytes from 192.168.53.5: icmp_seq=12 ttl=63 time=2.46 ms
64 bytes from 192.168.53.5: icmp_seq=13 ttl=63 time=2.73 ms
64 bytes from 192.168.53.5: icmp_seq=14 ttl=63 time=2.12 ms
64 bytes from 192.168.53.5: icmp_seq=15 ttl=63 time=2.50 ms
64 bytes from 192.168.53.5: icmp_seq=16 ttl=63 time=2.33 ms
64 bytes from 192.168.53.5: icmp_seq=17 ttl=63 time=1.79 ms
64 bytes from 192.168.53.5: icmp_seq=18 ttl=63 time=3.87 ms
64 bytes from 192.168.53.5: icmp_seq=19 ttl=63 time=1.99 ms
```

- Wireshark bắt gói tin Ping từ V sang U

The screenshot shows the Wireshark interface capturing ICMP traffic. The packet list pane displays 46 captured packets, all of which are ICMP Echo (ping) requests and replies between two hosts. The details pane shows the structure of one of the ICMP frames, and the bytes pane shows the raw hex and ASCII data. The packet list includes entries like:

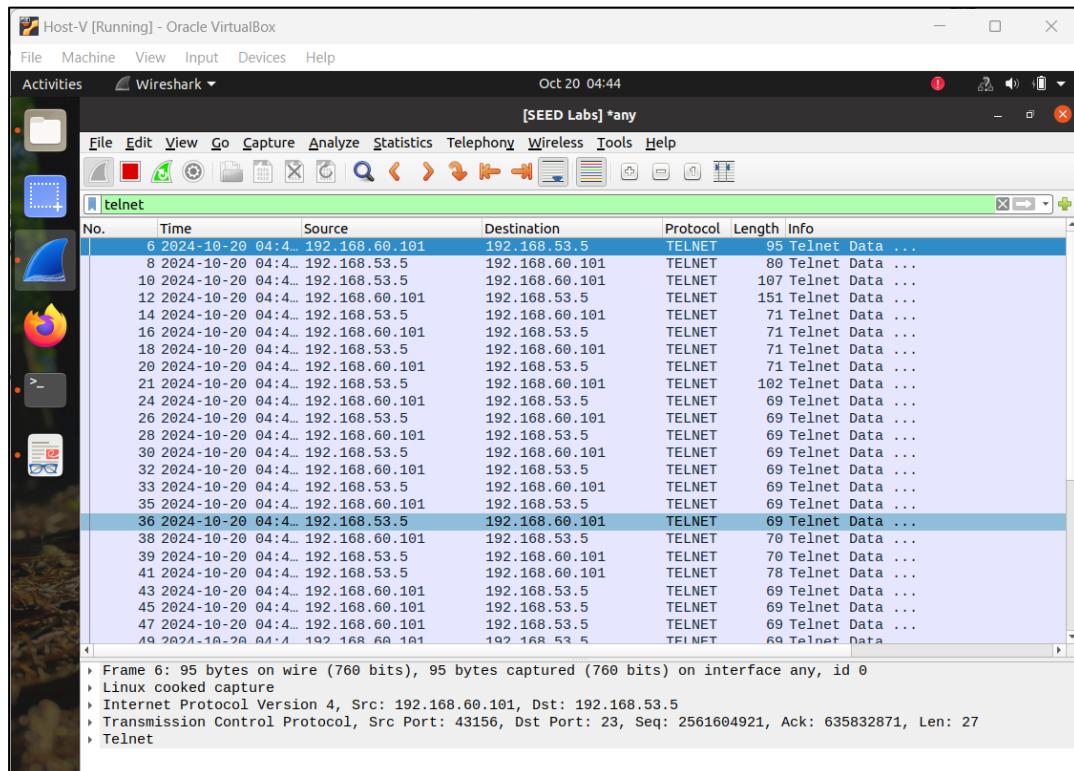
- Frame 98: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
- Linux cooked capture
- Internet Protocol Version 4, Src: 192.168.60.101, Dst: 192.168.53.5
- Internet Control Message Protocol

- Telnet từ V sang U



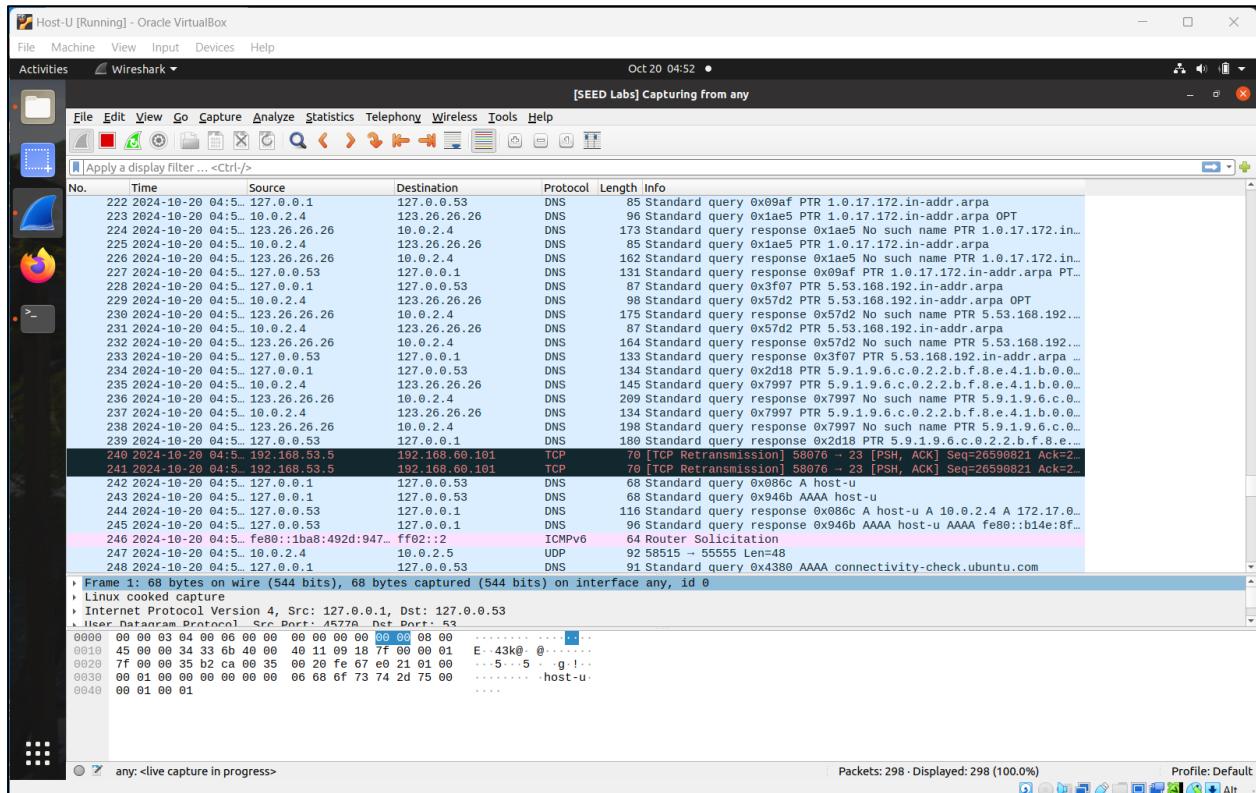
The screenshot shows a terminal window titled "Host-V [Running] - Oracle VirtualBox". The terminal window has two tabs: "seed@host-v: ~.../vpn" and "seed@host-u: ~". The "seed@host-u: ~" tab is active, showing the output of a Telnet session. The session starts with "Trying 192.168.53.5...", followed by "Connected to 192.168.53.5.", "Escape character is '^]'.", and "Ubuntu 20.04.1 LTS". It then prompts for a login with "host-u login: seed" and a password. After logging in, it displays a welcome message: "Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)". It provides documentation links and information about updates: "0 updates can be installed immediately." and "0 of these updates are security updates.". A note at the bottom states: "The list of available updates is more than a week old. To check for new updates run: sudo apt update". It also mentions support until April 2025 and the last login details: "Last login: Sun Oct 20 04:22:22 EDT 2024 from 192.168.60.101 on pts/2". The session ends with "[10/20/24] seed@host-u:~\$".

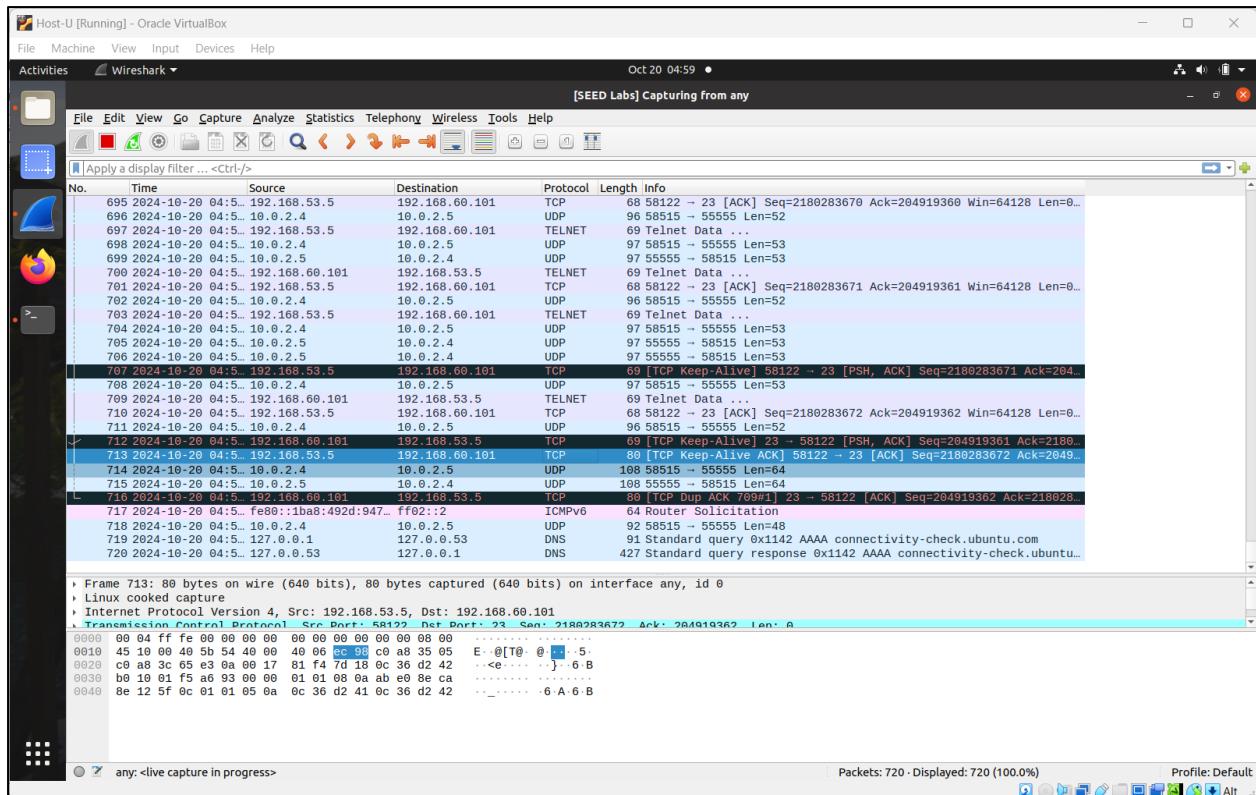
- Wireshark bắt gói tin Telnet từ V sang U



3. TASK 3: Trên Host U, telnet đến Host V. Trong khi vẫn duy trì kết nối telnet, hãy ngắt VPN tunnel. Sau đó, gõ ký tự bất kỳ vào cửa sổ telnet và báo cáo những gì bạn quan sát được. Sau đó, kết nối lại VPN tunnel. Điều gì sẽ xảy ra với kết nối telnet? Nó sẽ bị ngắt hay được tiếp tục? Vui lòng mô tả và giải thích những quan sát của bạn.

- Khi ngắt kết nối VPN tunnel, cửa sổ telnet không hiện những ký tự vừa gõ, hình dưới là wireshark chụp lại Gói tin bị drop khi ngắt kết nối:





- Giải thích: Những ký tự được gõ trong màn hình telnet sẽ được gửi bằng TCP thông qua VPN tunnel đến port 55555 của VPNServer. Nhưng khi ngắt kết nối thì không có gì lắng nghe ở port nên VPNServer sẽ drop gói tin truyền. Gói tin sẽ không đến được telnet server và không hiển thị ra màn hình ký tự được gửi

- Khi kết nối lại thì bên Host V hiện lại những ký tự đã được gõ từ HostU khi Telnet

```
[10/20/24]seed@host-u:~/.../vpn$ telnet 192.168.60.101
Trying 192.168.60.101...
Connected to 192.168.60.101.
Escape character is '^'.
Ubuntu 20.04.1 LTS
host-v 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

0 updates can be installed immediately.
0 of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Oct 20 04:49:47 EDT 2024 on pts/0
[10/20/24]seed@host-v:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[10/20/24]seed@host-v:~$ fddfdffddffddssddsdsds
```

- Giải thích: Ký tự gõ trong telnet sẽ được truyền bằng TCP do đó nó được lưu vào bộ nhớ đệm và chờ truyền lại nếu không truyền được. Khi kết nối lại, việc truyền lại đã đến được telnet server nên xuất hiện lên màn hình

* Cách hoạt động của VPN tunnel:

- Phía server:

```

28 int initUDPServer() {
29     int sockfd;
30     struct sockaddr_in server;
31     char buff[100];
32
33     memset(&server, 0, sizeof(server));
34     server.sin_family = AF_INET;
35     server.sin_addr.s_addr = htonl(INADDR_ANY);
36     server.sin_port = htons(PORT_NUMBER);
37
38     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
39     bind(sockfd, (struct sockaddr*) &server, sizeof(server));
40
41     // Wait for the VPN client to "connect".
42     bzero(buff, 100);
43     int peerAddrLen = sizeof(struct sockaddr_in);
44     int len = recvfrom(sockfd, buff, 100, 0,
45                         (struct sockaddr *) &peerAddr, &peerAddrLen);
46
47     printf("Connected with the client: %s\n", buff);
48     return sockfd;
49 }
```

+ Cấu hình server với dòng 34, 35, 36. server.sin_family = AF_INET là server được thiết lập với địa chỉ Ipv4, server.sin_addr.s_addr = htonl(INADDR_ANY) cho biết rằng chấp nhận bất kì địa chỉ IP nào kết nối đến server và server.sin_port = htons(PORT_NUMBER) nghĩa là server được thiết lập với port 55555.

- + Thiết lập socket cho kết nối UDP bằng cách sử dụng API socket().
- + Lệnh bind() dùng để liên kết socket với port 55555 của server.
- + Dòng 43, 44, 45 được sử dụng để nhận gói tin và đọc dữ liệu UDP đến port 55555 từ VPN client

- Phía Client:

```

28 int connectToUDPServer(){
29     int sockfd;
30     char *hello="Hello";
31
32     memset(&peerAddr, 0, sizeof(peerAddr));
33     peerAddr.sin_family = AF_INET;
34     peerAddr.sin_port = htons(PORT_NUMBER);
35     peerAddr.sin_addr.s_addr = inet_addr(SERVER_IP);
36
37     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
38
39     // Send a hello message to "connect" with the VPN server
40     sendto(sockfd, hello, strlen(hello), 0,
41             (struct sockaddr *) &peerAddr, sizeof(peerAddr));
42
43     return sockfd;
44 }
```

+ Cấu hình client để được thiết lập với địa chỉ Ipv4, chỉ chấp nhận địa chỉ IP của server là 10.0.2.8 và port là 55555.

+ Cũng như bên server, client cũng cần thiết lập socket và sẽ gửi tin nhắn “hello” đến với server để xác nhận kết nối.

```

48 void tunSelected(int tunfd, int sockfd){
49     int len;
50     char buff[BUFF_SIZE];
51
52     printf("Got a packet from TUN\n");
53
54     bzero(buff, BUFF_SIZE);
55     len = read(tunfd, buff, BUFF_SIZE);
56     sendto(sockfd, buff, len, 0, (struct sockaddr *) &peerAddr,
57             sizeof(peerAddr));
58 }
59
60 void socketSelected (int tunfd, int sockfd){
61     int len;
62     char buff[BUFF_SIZE];
63
64     printf("Got a packet from the tunnel\n");
65
66     bzero(buff, BUFF_SIZE);
67     len = recvfrom(sockfd, buff, BUFF_SIZE, 0, NULL, NULL);
68     write(tunfd, buff, len);
69 }
70 }
```

+ Cả hai hàm tunSelected và socketSelected giống nhau ở client và server với các chức năng như sau :

+ Hàm tunSelected được sử dụng khi TUN interface có dữ liệu để đọc dữ liệu từ TUN interface bằng hàm read(), sau đó dùng hàm sendto() để gửi dữ liệu nhận được qua tunnel đã tạo trước đó

+ Hàm socketSelected được sử dụng khi có dữ liệu truyền đến port thông qua socket để đọc dữ liệu sử dụng hàm recvfrom() và ghi dữ liệu đó vào TUN interface dùng hàm write()

- Về luồng chạy:

- + Server và Client sẽ tạo một TUN interface và thiết lập socket với nhau sử dụng UDP
- + Sau khi thiết lập xong chương trình sẽ chạy vào main loop

```

71 int main (int argc, char * argv[]) {
72     int tunfd, sockfd;
73
74     tunfd = createTunDevice();
75     sockfd = connectToUDPServer();
76
77     // Enter the main loop
78     while (1) {
79         fd_set readFDSet;
80
81         FD_ZERO(&readFDSet);
82         FD_SET(sockfd, &readFDSet);
83         FD_SET(tunfd, &readFDSet);
84         select(FD_SETSIZE, &readFDSet, NULL, NULL, NULL);
85
86         if (FD_ISSET(tunfd, &readFDSet)) tunSelected(tunfd, sockfd);
87         if (FD_ISSET(sockfd, &readFDSet)) socketSelected(tunfd, sockfd);
88     }
89 }
```

+ Dòng 82 – 87 dùng để quan sát xem đã nhận dữ liệu từ TUN hay socket hay chưa, hàm select() sẽ block process đến khi một trong 2 TUN và socket có dữ liệu.

+ Sử dụng hàm FD_ISSET để kiểm tra dữ liệu có tồn tại trong TUN hay socket hay không. Nếu có gọi hàm lấy dữ liệu tương ứng

- Luồng chạy khi ping từ hostU đến hostV:

+ Khi gửi gói tin từ Host U, gói tin sẽ đi vào tunnel (tun0) của VPN Client và được đọc bởi hàm tunSelected() và gửi đến VPN Server qua UDP bằng hàm sendto().

+ Tại VPN Server, nhận gói tin được gửi từ VPN Client bằng hàm recvfrom() trong socketSelected() và ghi vào TUN (tun0) qua hàm write().

+ Tiếp đó, gói tin từ TUN sẽ được định tuyến đến Host V bởi Server.

- + Host V nhận gói tin và gửi phản hồi với địa chỉ nguồn là 192.168.60.101 đến địa chỉ đích là 192.168.53.5.
- + Gói tin sẽ đi đến VPN Server qua mạng nội bộ là 192.168.60.1.
- + VPN Server nhận gói tin phản hồi và đọc gói tin từ TUN (tun0) qua hàm tunSelected() của VPN Server.
- + Sau khi đọc gói tin, VPN Server sẽ gửi đến VPN Client qua hàm sendto().
- + VPN Client nhận gói tin phản hồi từ VPN Server qua hàm recvfrom() trong socketSelected() và ghi vào TUN (tun0) bằng hàm write().
- + Sau đó gói tin phản hồi được phản hồi đến HostU

4. TASK 4: Chạy thực thi đoạn code mã hoá TLS. Dùng Wireshark bắt gói tin, để chứng minh nội dung gói tin gửi đi giữa client và server đã được mã hoá thành công.

- Do chứng chỉ cũ đã hết hạn, ta dùng lệnh sau để CA ký CSR của server và tạo ra một chứng chỉ server mới (server-cert.pem). Chứng chỉ này xác thực danh tính của máy chủ, đảm bảo rằng máy chủ là hợp lệ:

```
openssl x509 -req -in server-csr.pem -CA cacert.pem -CAkey cakey.pem -CAcreateserial -out server-cert.pem -days 365 -sha256
```

- Kết quả tạo cert thành công:

```
[10/19/24] seed@VM:~/tls/cert_server$ openssl x509 -req -in server-csr.pem -CA cacert.pem -CAkey cakey.pem -CAcreateserial -out server-cert.pem -days 365 -sha256
Signature ok
subject=C = US, ST = NY, L = SYR, O = SYR, OU = SYR, CN = vpnlabserver.com
Getting CA Private Key
Enter pass phrase for cakey.pem:
[10/19/24] seed@VM:~/tls/cert_server$ make clean
```

- Thực hiện lệnh sau trên máy Client (Host U):

```
sudo nano /etc/hosts
```

- Thêm dòng sau để ánh xạ tên miền vpnlabserver.com với địa chỉ IP của Server (Gateway):

```
10.0.2.9    vpnLabserver.com
```

- Sau khi **cd** vào thư mục **tls**, thực hiện compile trên cả Client và Server:

```
make
```

- Phía Server:

```
sudo ./tlsserver
```

- Phía Client:

```
./tlsclient vpnLabserver.com 4433
```

Lab 02: VPN

- Sau khi thực hiện, ta được kết quả như sau:

+ Server (Gateway):

```
[10/20/24] seed@VM:~/.../tls$ sudo ./tlsserver
SSL connection established!
Received: GET / HTTP/1.1
Host: vpnlabserver.com
```

+ Client (Host U):

```
[10/20/24] seed@VM:~/.../tls$ ./tlsclient vpnlabserver.com 4433
SSL connection is successful
SSL connection using ECDHE-RSA-AES256-GCM-SHA384
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html><html><head><title>Hello World</title></head><style>body
{background-color: black}h1 {font-size:3cm; text-align: center; color:
white;text-shadow: 0 0 3mm yellow}</style></head><body><h1>Hello, worl
d!</h1></body></html>
[Show Applications]
[10/20/24] seed@VM:~/.../tls$
```

- Dùng Wireshark để quan sát các gói tin đã được mã hóa TLS:

No.	Time	Source	Destination	Protocol	Length	Info
17	2024-10-20 05:4...	10.0.2.9	10.0.2.10	TCP	68	4433 - 54718 [ACK] Seq=769614768 Ack=2055627985 Win=65024 Len...
18	2024-10-20 05:4...	10.0.2.9	10.0.2.10	TLSv1.2	294	New Session Ticket, Change Cipher Spec, Encrypted Handshake M...
19	2024-10-20 05:4...	10.0.2.10	10.0.2.9	TCP	68	54718 - 4433 [ACK] Seq=2055627985 Ack=769614994 Win=64128 Len...
20	2024-10-20 05:4...	10.0.2.10	10.0.2.9	TLSv1.2	136	Application Data
21	2024-10-20 05:4...	10.0.2.9	10.0.2.10	TCP	68	4433 - 54718 [ACK] Seq=769614994 Ack=2055628053 Win=65024 Len...
22	2024-10-20 05:4...	10.0.2.9	10.0.2.10	TLSv1.2	375	Application Data
23	2024-10-20 05:4...	10.0.2.9	10.0.2.10	TLSv1.2	98	Encrypted Alert
24	2024-10-20 05:4...	10.0.2.10	10.0.2.9	TCP	68	54718 - 4433 [ACK] Seq=2055628053 Ack=769615301 Win=64128 Len...
25	2024-10-20 05:4...	10.0.2.10	10.0.2.9	TCP	68	54718 - 4433 [FIN, ACK] Seq=2055628053 Ack=769615333 Win=6412...
26	2024-10-20 05:4...	10.0.2.9	10.0.2.10	TCP	68	4433 - 54718 [ACK] Seq=769615333 Ack=2055628054 Win=65024 Len...
27	2024-10-20 05:4...	PcsCompu_4b:73:4d		ARP	62	Who has 10.0.2.10? Tell 10.0.2.9
28	2024-10-20 05:4...	PcsCompu_1e:a1:a5		ARP	44	10.0.2.10 is at 08:00:27:1e:a1:a5
29	2024-10-20 05:4...	PcsCompu_1e:a1:a5		ARP	44	Who has 10.0.2.9? Tell 10.0.2.10
30	2024-10-20 05:4...	PcsCompu_4b:73:4d		ARP	62	10.0.2.9 is at 08:00:27:4b:73:4d
31	2024-10-20 05:4...	10.0.2.10	10.0.2.3	DHCP	324	DHCP Request - Transaction ID 0x8fa60024
32	2024-10-20 05:4...	10.0.2.3	10.0.2.10	DHCP	592	DHCP ACK - Transaction ID 0x8fa60024
33	2024-10-20 05:4...	10.0.2.10	192.168.2.253	DNS	91	Standard query 0xa0b1 A connectivity-check.ubuntu.com
34	2024-10-20 05:4...	192.168.2.253	10.0.2.10	DNS	283	Standard query response 0xa0b1 A connectivity-check.ubuntu.com...
35	2024-10-20 05:4...	192.168.2.253	10.0.2.10	TCP	76	13096 - 80 [SYN] Seq=3711516131 Win=64128 Len=0 MSS=1464 SACK...

- Giải thích:

+ tlsserver.c:

- Chương trình này thiết lập **server SSL/TLS** trên cổng 4433.
- Khi client kết nối, server xử lý yêu cầu và gửi trang HTML đơn giản chào mừng.
- Điều này giúp ta hiểu rõ cách tạo **socket bảo mật** và truyền dữ liệu qua SSL/TLS..

+ tlsclient.c:

- Chương trình tạo một **client SSL/TLS** kết nối tới một server HTTPS.

- Nó thiết lập kết nối TCP, bắt tay TLS, và gửi một yêu cầu HTTP GET.
- Chứng chỉ server được kiểm tra thông qua **callback verify_callback()**.
- Cuối cùng, dữ liệu phản hồi từ server được in ra màn hình.

- Phân tích code:

+ Phía server:

- Đầu tiên server sẽ khởi tạo SSL context, load certificate, key và tạo một cấu trúc SSL để kết nối:

```

13 int main(){
14
15     SSL_METHOD *meth;
16     SSL_CTX* ctx;
17     SSL *ssl;
18     int err;
19
20     // Step 0: OpenSSL library initialization
21     // This step is no longer needed as of version 1.1.0.
22     SSL_library_init();
23     SSL_load_error_strings();
24     SSLeay_add_ssl_algorithms();
25
26     // Step 1: SSL context initialization
27     meth = (SSL_METHOD *)TLSv1_2_method();
28     ctx = SSL_CTX_new(meth);
29     SSL_CTX_set_verify(ctx, SSL_VERIFY_NONE, NULL);
30     // Step 2: Set up the server certificate and private key
31     SSL_CTX_use_certificate_file(ctx, "./cert_server/server-cert.pem", SSL_FILETYPE_PEM);
32     SSL_CTX_use_PrivateKey_file(ctx, "./cert_server/server-key.pem", SSL_FILETYPE_PEM);
33     // Step 3: Create a new SSL structure for a connection
34     ssl = SSL_new (ctx);
35
36     struct sockaddr_in sa_client;
37     size_t client_len;
38     int listen_sock = setupTCPServer();
```

- Tiếp theo tạo kết nối TCP : tạo một socket và gán socket với port 4433. Sau khi thiết lập xong đặt socket vào chế độ lắng nghe các kết nối sử dụng hàm listen():

```

60 int setupTCPServer()
61 {
62     struct sockaddr_in sa_server;
63     int listen_sock;
64
65     listen_sock= socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
66     CHK_ERR(listen_sock, "socket");
67     memset (&sa_server, '\0', sizeof(sa_server));
68     sa_server.sin_family      = AF_INET;
69     sa_server.sin_addr.s_addr = INADDR_ANY;
70     sa_server.sin_port        = htons (4433);
71     int err = bind(listen_sock, (struct sockaddr*)&sa_server, sizeof(sa_server));
72     CHK_ERR(err, "bind");
73     err = listen(listen_sock, 5);
74     CHK_ERR(err, "listen");
75     return listen_sock;
76 }
77

```

- Sau khi thiết lập xong kết nối TCP tạo vòng lặp để server luôn lắng nghe và chấp nhận các kết nối đến. Tạo một tiến trình con để xử lý kết nối vừa chấp nhận, còn tiến trình cha tiếp tục lắng nghe các kết nối. Trong tiến trình con, sử dụng SSL_set_fd để liên kết socket được chấp nhận với đối tượng SSL đã khởi tạo lúc đầu. Thiết lập kết nối SSL và tiến hành xử lý bằng hàm processRequest():

```

40 while(1){
41     int sock = accept(listen_sock, (struct sockaddr*)&sa_client, &client_len);
42     if (fork() == 0) { // The child process
43         close (listen_sock);
44
45         SSL_set_fd (ssl, sock);
46         int err = SSL_accept (ssl);
47         CHK_SSL(err);
48         printf ("SSL connection established!\n");
49
50         processRequest(ssl, sock);
51         close(sock);
52         return 0;
53     } else { // The parent process
54         close(sock);
55     }
56 }
57 }

```

- Hàm “**processRequest**” tiến hành đọc dữ liệu từ client thông qua kết nối SSL và sau đó gửi lại HTML page. Sau khi gửi HTML page thì đóng kết nối và giải phóng tài nguyên.

```

78 void processRequest(SSL* ssl, int sock)
79 {
80     char buf[1024];
81     int len = SSL_read(ssl, buf, sizeof(buf) - 1);
82     buf[len] = '\0';
83     printf("Received: %s\n", buf);
84
85     // Construct and send the HTML page
86     char *html =
87     "HTTP/1.1 200 OK\r\n"
88     "Content-Type: text/html\r\n\r\n"
89     "<!DOCTYPE html><html>"
90     "<head><title>Hello World</title></head>"
91     "<style>body {background-color: black}"
92     "h1 {font-size:3cm; text-align: center; color: white;}"
93     "text-shadow: 0 0 3mm yellow}</style></head>"
94     "<body><h1>Hello, world!</h1></body></html>";
95     SSL_write(ssl, html, strlen(html));
96     SSL_shutdown(ssl); SSL_free(ssl);
97 }
```

+ Bên Client:

- Client khởi tạo thư viện và chọn phương thức TLSv1_2_method(). Sau đó client khởi tạo ngữ cảnh SSL (ctx) và sử dụng SSL_CTX_set_verify() để thiết lập chế độ kiểm tra chứng chỉ của Server. Client cấu hình đường dẫn đến nơi chứa chứng chỉ của client CA_DIR(ca_client), nếu như đường dẫn không đúng, kết nối sẽ bị từ chối:

```

27  ~ SSL* setupTLSclient(const char* hostname)
28  {
29  ~ | // Step 0: OpenSSL library initialization
30  ~ | // This step is no longer needed as of version 1.1.0.
31  ~ | SSL_library_init();
32  ~ | SSL_load_error_strings();
33  ~ | SSL_add_ssl_algorithms();
34
35  ~ | SSL_METHOD *meth;
36  ~ | SSL_CTX* ctx;
37  ~ | SSL* ssl;
38
39  ~ | meth = (SSL_METHOD *)TLSv1_2_method();
40  ~ | ctx = SSL_CTX_new(meth);
41
42  ~ | SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, NULL);
43  ~ | if(SSL_CTX_load_verify_locations(ctx,NULL, CA_DIR) < 1){
44  ~ | printf("Error setting the verify locations. \n");
45  ~ | exit(0);
46  ~ |
47  ~ | ssl = SSL_new (ctx);
48
49  ~ | X509_VERIFY_PARAM *vpm = SSL_get0_param(ssl);
50  ~ | X509_VERIFY_PARAM_set1_host(vpm, hostname, 0);
51
52  ~ | return ssl;
53  }

```

- Tạo kết nối TCP : sử dụng gethostbyname() để lấy địa chỉ IP từ hostname, socket() được dùng để tạo TCP socket và kết nối đến máy chủ theo port được nhập:

```

56 int setupTCPClient(const char* hostname, int port)
57 {
58     struct sockaddr_in server_addr;
59
60     // Get the IP address from hostname
61     struct hostent* hp = gethostbyname(hostname);
62
63     // Create a TCP socket
64     int sockfd= socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
65
66     // Fill in the destination information (IP, port #, and family)
67     memset (&server_addr, '\0', sizeof(server_addr));
68     memcpy(&(server_addr.sin_addr.s_addr), hp->h_addr, hp->h_length);
69     // server_addr.sin_addr.s_addr = inet_addr ("10.0.2.14");
70     server_addr.sin_port = htons (port);
71     server_addr.sin_family = AF_INET;
72
73     // Connect to the destination
74     connect(sockfd, (struct sockaddr*) &server_addr,
75             sizeof(server_addr));
76
77     return sockfd;
78 }
```

- Bắt tay TLS : Thiết lập cấu trúc SSL_set_fd() cho TCP socket và thực hiện bắt tay bằng SSL_connect():

```

95     /*-----TLS handshake -----*/
96     SSL_set_fd(ssl, sockfd);
97     int err = SSL_connect(ssl); CHK_SSL(err);
98     printf("SSL connection is successful\n");
99     printf ("SSL connection using %s\n", SSL_get_cipher(ssl));
100
```

- Client gửi một yêu cầu HTTP GET đến Server. Client đọc phản hồi từ Server bằng SSL_read() và in ra phản hồi. Client sẽ đóng kết nối sau khi nhận dữ liệu từ máy chủ và giải phóng tài nguyên.

```

101     /*-----Send/Receive data -----*/
102     char buf[9000];
103     char sendBuf[200];
104     sprintf(sendBuf, "GET / HTTP/1.1\nHost: %s\n\n", hostname);
105     SSL_write(ssl, sendBuf, strlen(sendBuf));
106
107     int len;
108     do {
109         len = SSL_read (ssl, buf, sizeof(buf) - 1);
110         buf[len] = '\0';
111         printf("%s\n",buf);
112     } while (len > 0);
113 }
```