Karthikeyan G

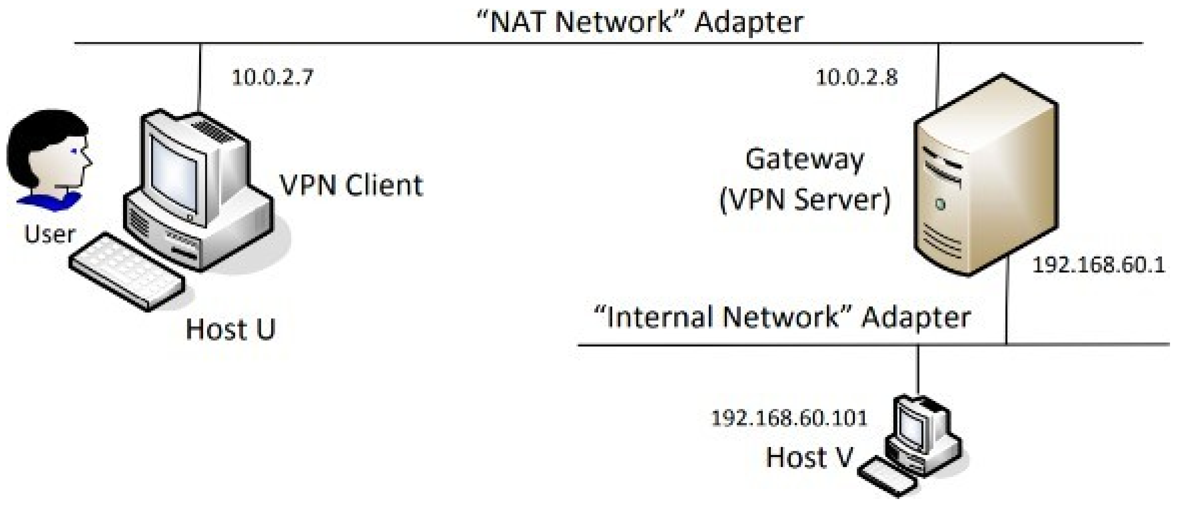
# 24CYS682 - Cyber Security Lab
# Assignment – 9
# Virtual Private Network Lab

## Task 1 – Setting Up Virtual Machines

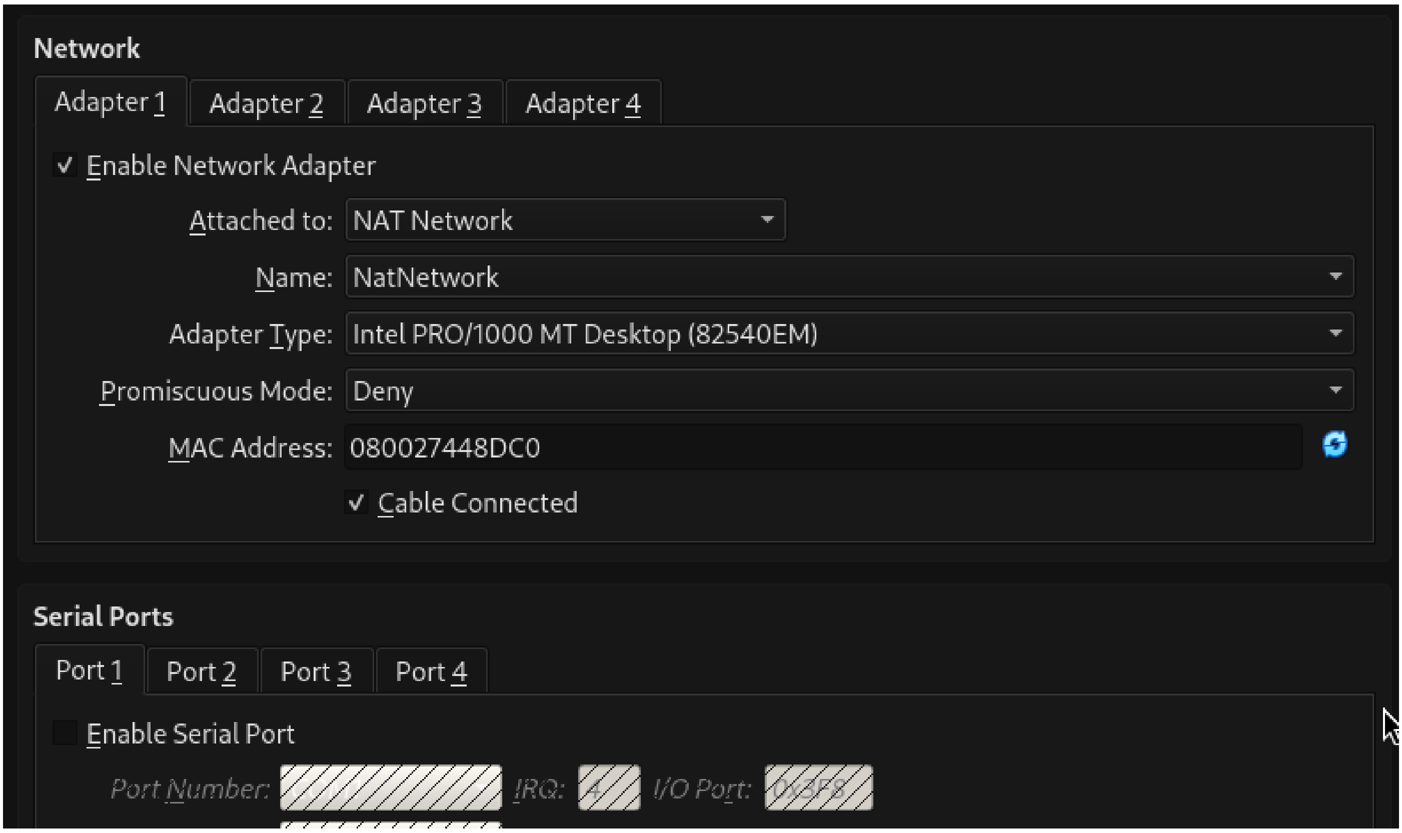
In this task, we will establish a VPN tunnel between a client computer and a gateway, enabling secure access to a private network through the gateway. This setup requires at least three virtual machines (VMs):

1. **VPN Client (Host U):** Acts as the client initiating the VPN connection.
2. **VPN Server (Gateway):** Functions as the VPN server, forwarding traffic between the client and the private network.
3. **Host V (Private Network Host):** A machine within the private network that Host U will access via the VPN.

The network topology illustrating this setup is shown in the figure.



To implement this setup, both the client and server will be connected through a **NAT network**. This configuration ensures proper communication between them while allowing the client to establish a secure VPN connection with the server.



Network

✓ Enable Network Adapter

Attached to: NAT Network

Name: NatNetwork

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Deny

MAC Address: 080027448DC0

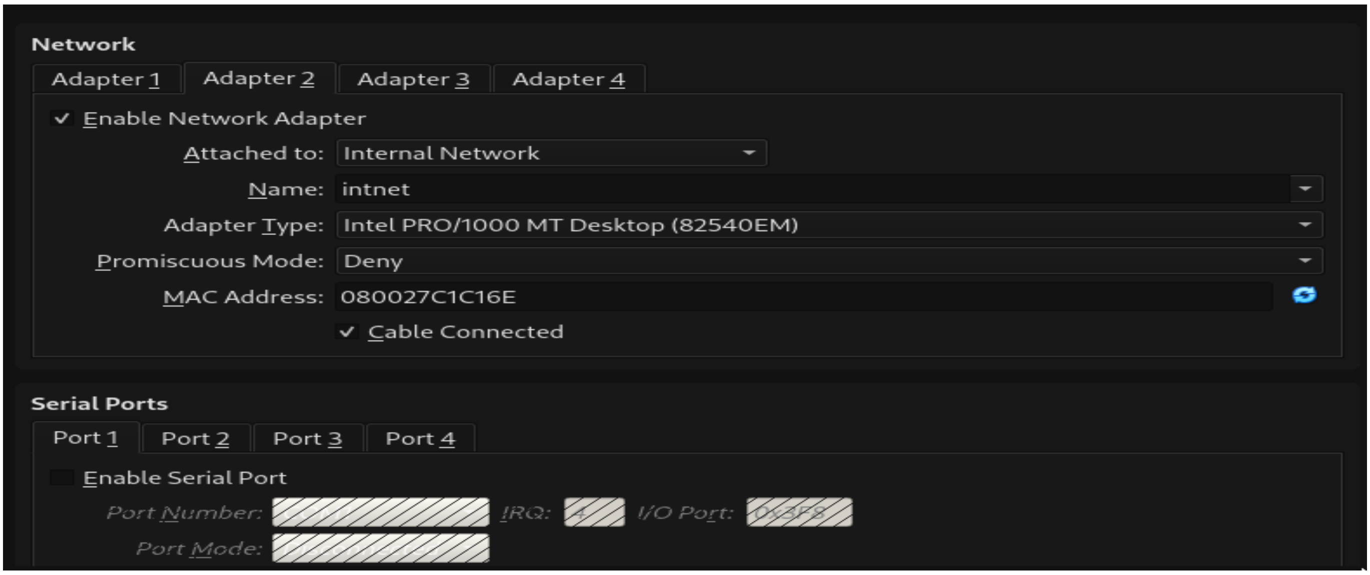
✓ Cable Connected

Serial Ports

□ Enable Serial Port

Port Number: IRQ: 4 I/O Port: 0x3F8

Meanwhile, the server and the host machine will be linked via an **internal network**, ensuring that the client and the host remain disconnected from each other.



**Network**

Adapter 1 | Adapter 2 | Adapter 3 | Adapter 4

✓ Enable Network Adapter

| | |
|---|---|
| Attached to: | Internal Network |
| Name: | intnet |
| Adapter Type: | Intel PRO/1000 MT Desktop (82540EM) |
| Promiscuous Mode: | Deny |
| MAC Address: | 080027C1C16E |

✓ Cable Connected

**Serial Ports**

Port 1 | Port 2 | Port 3 | Port 4

□ Enable Serial Port

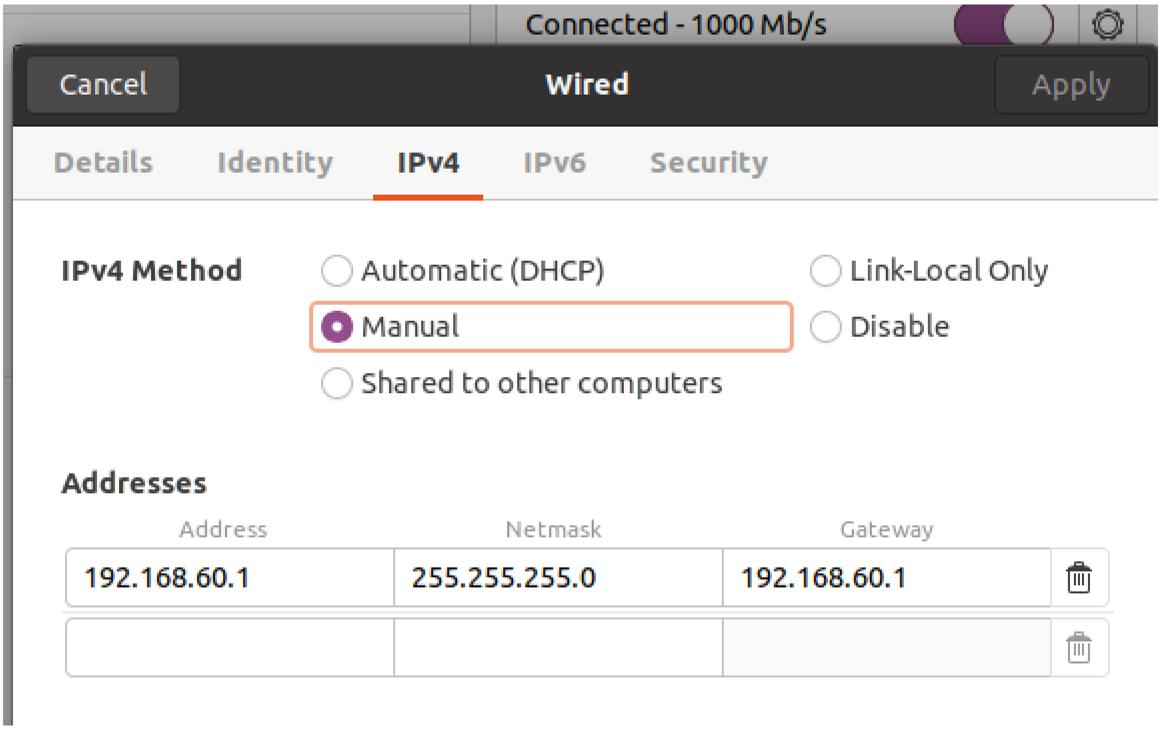
Port Number: | IRQ: 4 | I/O Port: 0x3F8

Port Mode:

## Server VM Configuration

The **Server VM** acts as the VPN gateway, facilitating communication between the client and the private network. It is configured with:

- A **NAT network** to connect with the VPN client.
- An **internal network** to communicate with the private network (Host V).
- Proper routing and forwarding settings to allow traffic between the client and the private network.

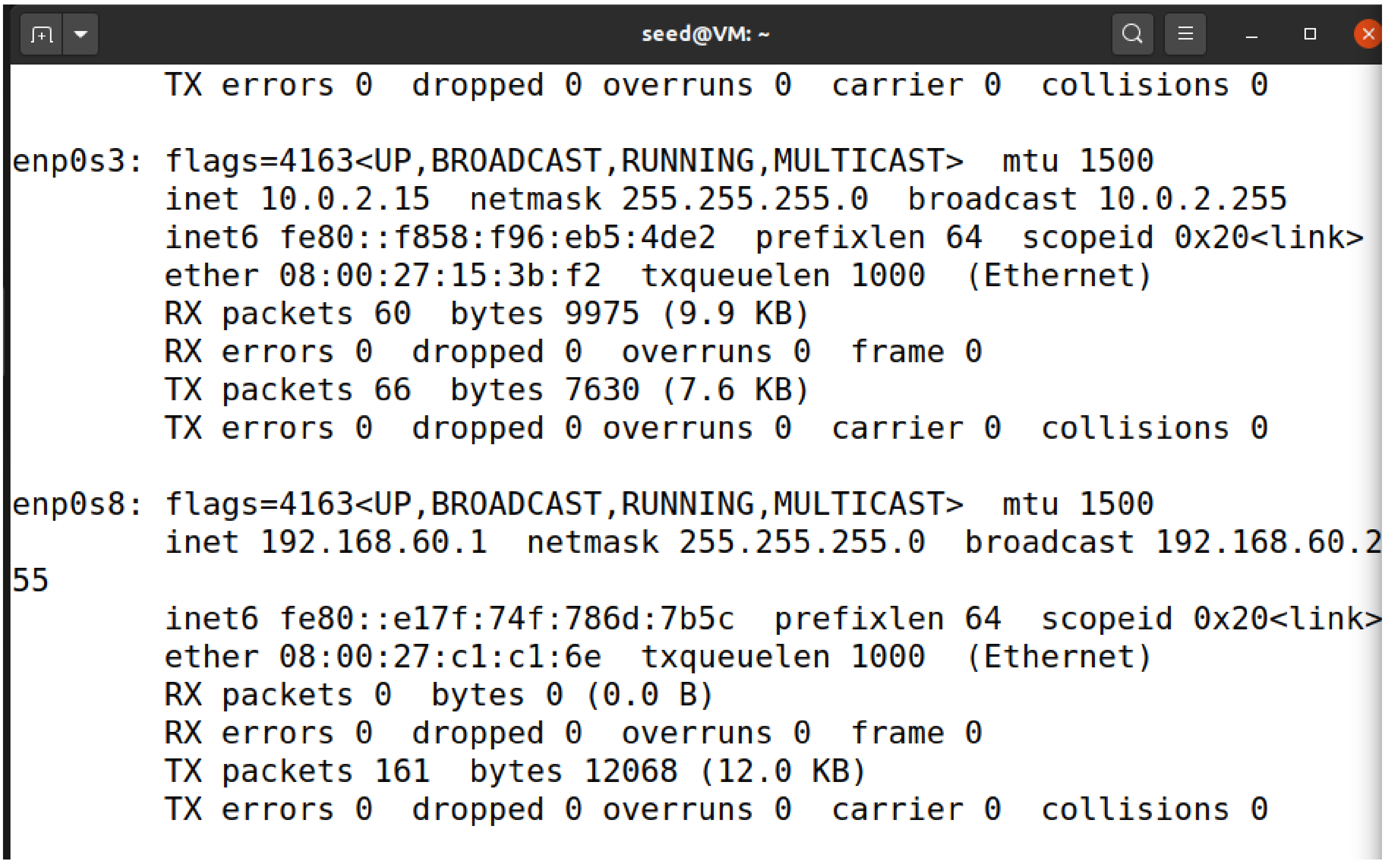


IPv4 Method

○ Automatic (DHCP)
○ Link-Local Only
● Manual
○ Disable
○ Shared to other computers

**Addresses**

| Address | Netmask | Gateway |
|---|---|---|
| 192.168.60.1 | 255.255.255.0 | 192.168.60.1 |
| | | |

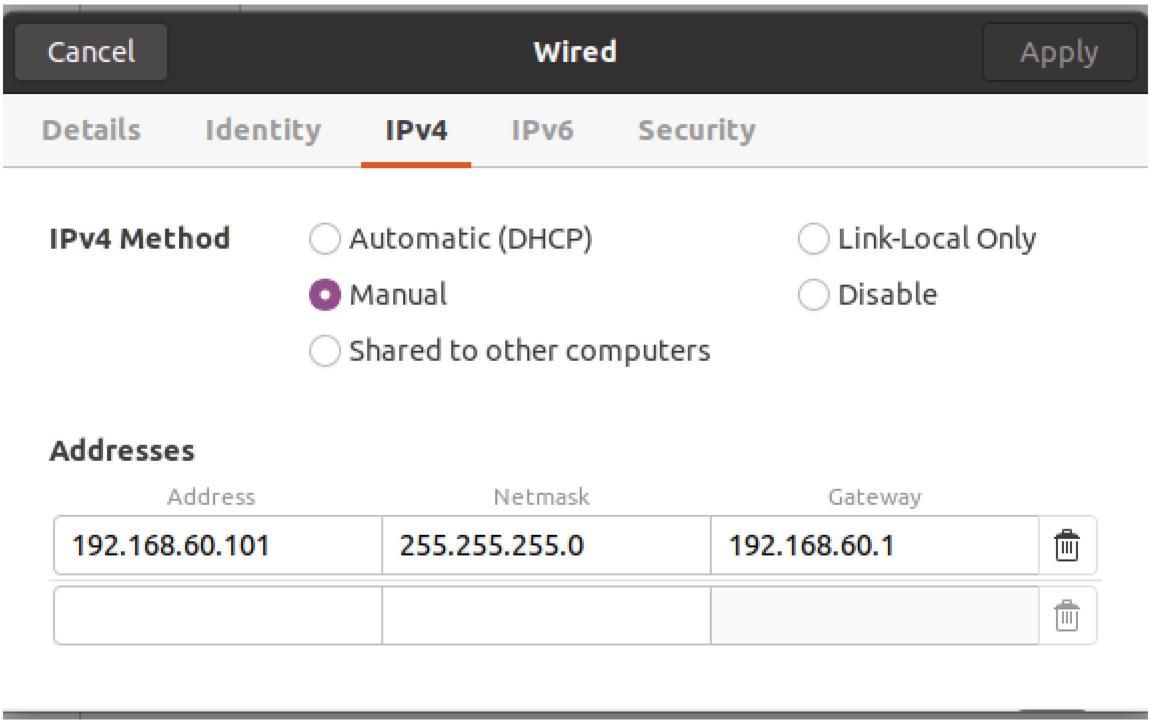


```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::f858:f96:eb5:4de2  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:15:3b:f2  txqueuelen 1000  (Ethernet)
        RX packets 60  bytes 9975 (9.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 66  bytes 7630 (7.6 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 192.168.60.2
55
        inet6 fe80::e17f:74f:786d:7b5c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:c1:c1:6e  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 161  bytes 12068 (12.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

## Host VM Configuration

The **Host VM (Host V)** represents a machine within the private network. It is configured with:

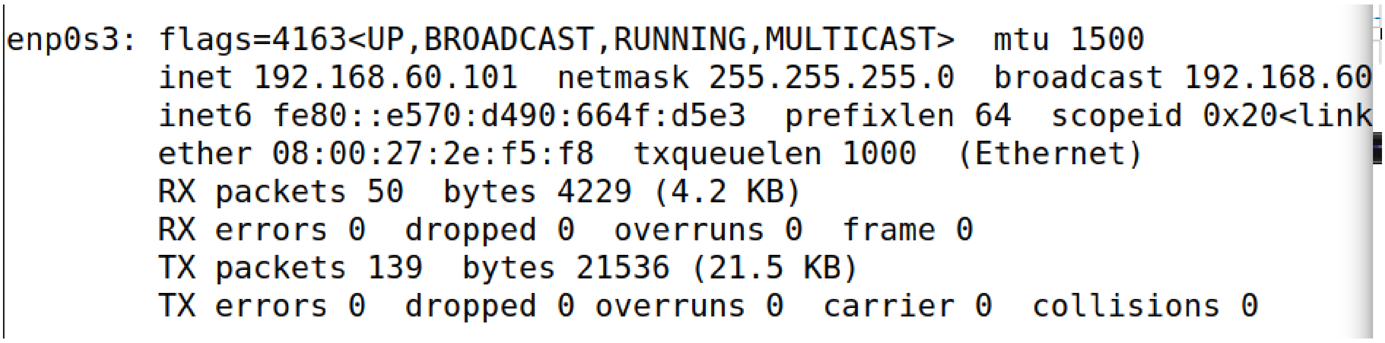
- An **internal network connection** to communicate with the **Server VM (VPN Gateway)**.
- No direct connection to the **Client VM (Host U)**, ensuring all traffic passes through the **VPN server**.
- Proper IP settings to allow communication with the VPN server and respond to forwarded traffic.



IPv4 Method
○ Automatic (DHCP)   ○ Link-Local Only
● Manual   ○ Disable
○ Shared to other computers

Addresses

| Address | Netmask | Gateway |
|---|---|---|
| 192.168.60.101 | 255.255.255.0 | 192.168.60.1 |
| | | |

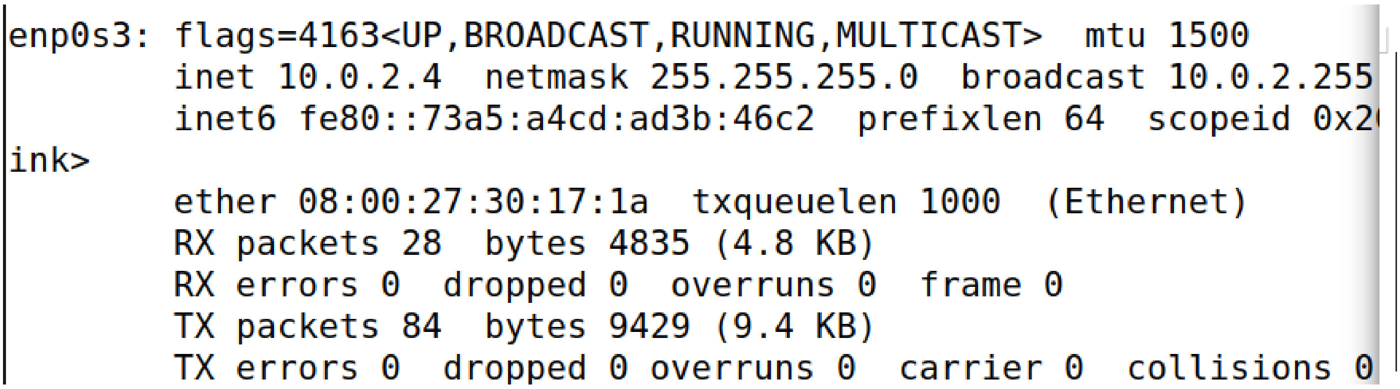


```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.60.101  netmask 255.255.255.0  broadcast 192.168.60
        inet6 fe80::e570:d490:664f:d5e3  prefixlen 64  scopeid 0x20<link
        ether 08:00:27:2e:f5:f8  txqueuelen 1000  (Ethernet)
        RX packets 50  bytes 4229 (4.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 139  bytes 21536 (21.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

### VPN Client Configuration

The **VPN Client (Host U)** is responsible for establishing a secure connection to the **VPN Server**. It is configured with:

- A **NAT network** to connect to the **VPN Server**.
- A **VPN tunnel interface** (e.g., `tun0`) to securely route traffic through the **VPN Server**.
- Proper routing rules to ensure that traffic destined for the private network is sent through the **VPN tunnel**.



```
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::73a5:a4cd:ad3b:46c2  prefixlen 64  scopeid 0x2
ink>
        ether 08:00:27:30:17:1a  txqueuelen 1000  (Ethernet)
        RX packets 28  bytes 4835 (4.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 84  bytes 9429 (9.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

### Established Network Connections

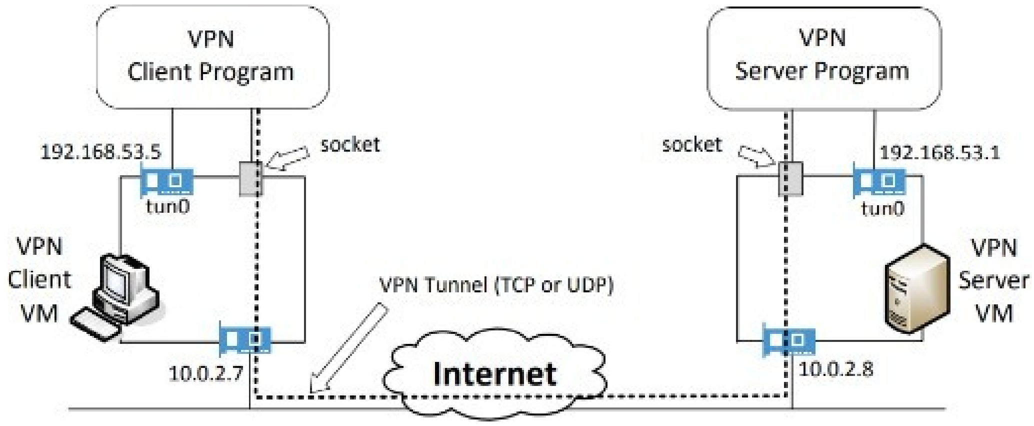
Based on the configurations shown in the screenshots, the following network connections are set up:

- **VPN Client (Host U)** – Adapter 1: **NAT Network**
- **VPN Server (Gateway)** – Adapter 1: **NAT Network**
  – Adapter 2: **Internal Network**
- **Host V (Private Network Host)** – Adapter 1: **Internal Network**

---

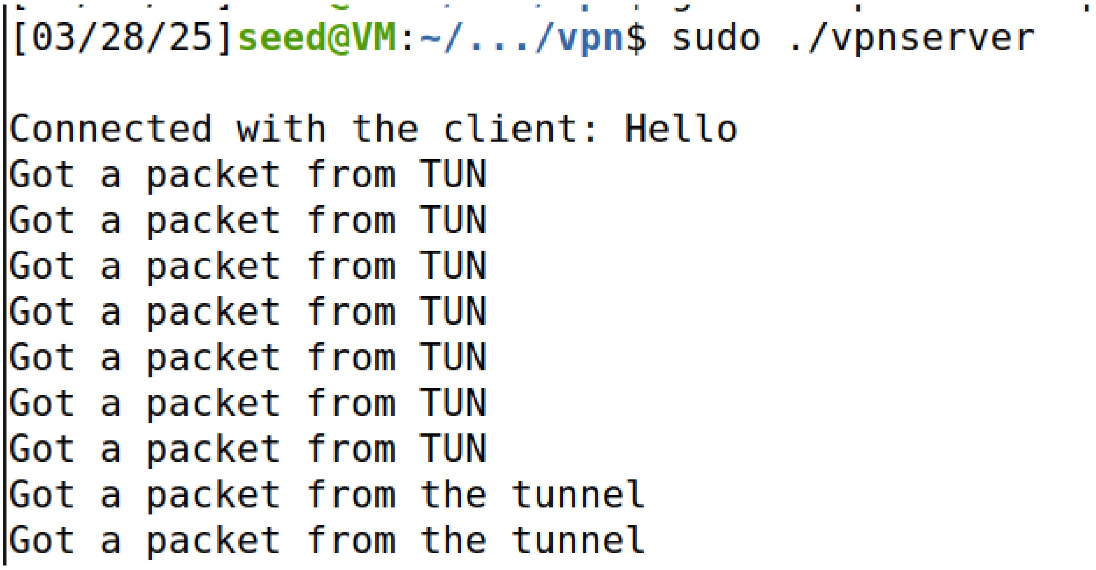
## Task 2: Setting Up a VPN Tunnel Using TUN/TAP

In this task, we will create a **VPN tunnel** using **TUN/TAP interfaces**, allowing secure communication between the **VPN Client** and the **private network via the VPN Server**.


VPN
Client Program
VPN
Server Program
192.168.53.5
socket
socket
192.168.53.1
tun0
tun0
VPN
Client
VM
VPN Tunnel (TCP or UDP)
VPN
Server
VM
10.0.2.7
Internet
10.0.2.8

### Step 1: Start the VPN Server

First, launch the **VPN Server** and configure the IP address for its interface. Then, execute the `vpnserver.c` code on the server machine to initialize the VPN service.

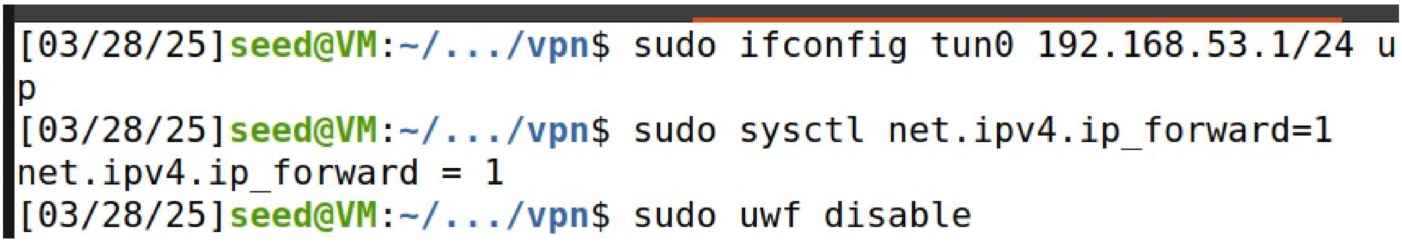


```
[03/28/25]seed@VM:~/.../vpn$ sudo ./vpnserver

Connected with the client: Hello
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
```

Next, we configure the **`tun0` interface** by assigning it an IP address and activating it.

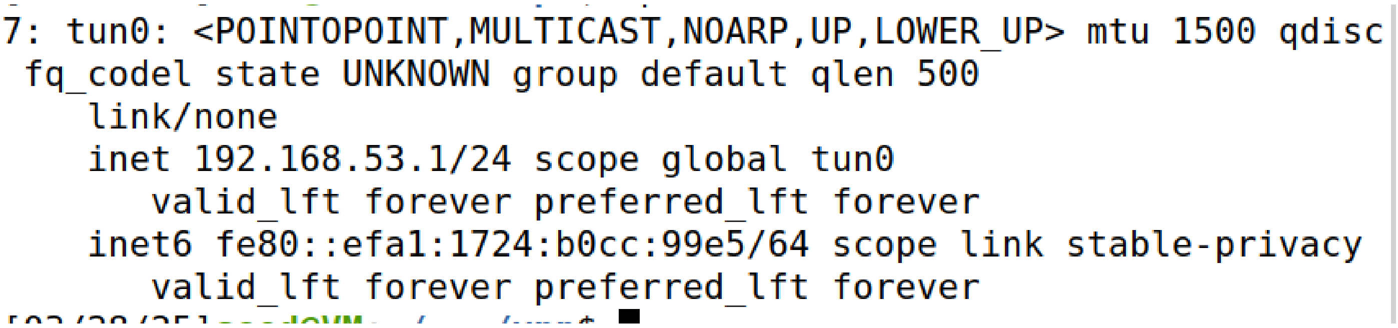
- **Assigned IP Address:** `192.168.53.1/24`
- **Enable IP forwarding** to allow traffic to pass through the VPN.

After verifying with `ifconfig`, we confirm that the **VPN tunnel is successfully established**



```
[03/28/25]seed@VM:~/.../vpn$ sudo ifconfig tun0 192.168.53.1/24 u
p
[03/28/25]seed@VM:~/.../vpn$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[03/28/25]seed@VM:~/.../vpn$ sudo uwf disable
```

*$ ip a show tun0*

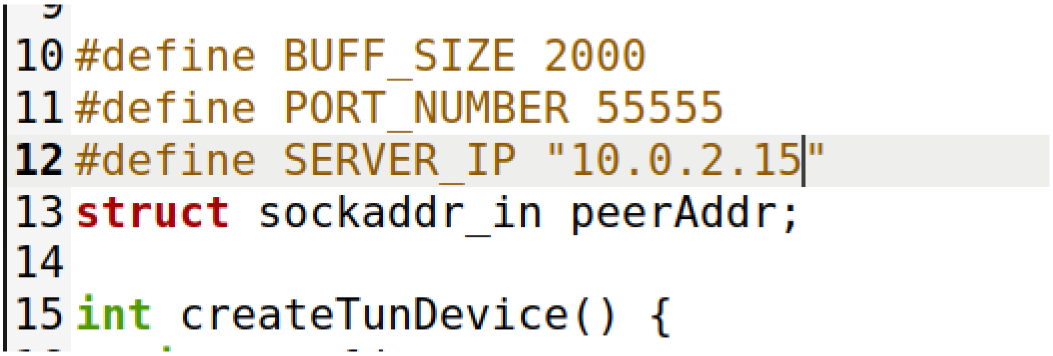


```
7: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
 fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.1/24 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::efa1:1724:b0cc:99e5/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

The tunnel is now active. Since the **VPN Server** must forward packets to other destinations, it needs to operate as a **gateway**. To achieve this, we must enable **IP forwarding**, allowing the system to route traffic between networks.

### Step 2: Run VPN Client

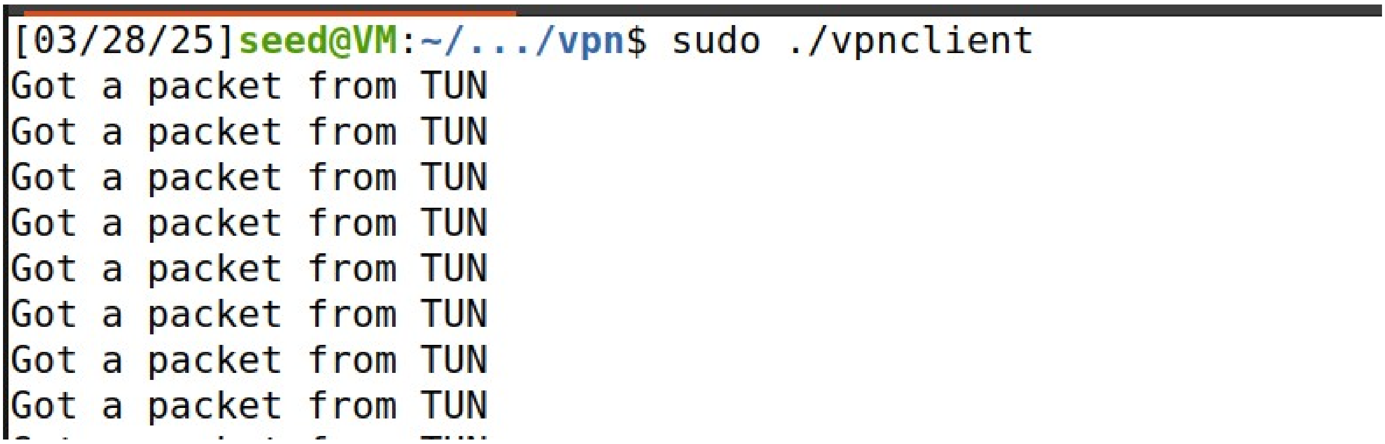
Set server ip in client code.



```
#define BUFF_SIZE 2000
#define PORT_NUMBER 55555
#define SERVER_IP "10.0.2.15"
struct sockaddr_in peerAddr;

int createTunDevice() {
```

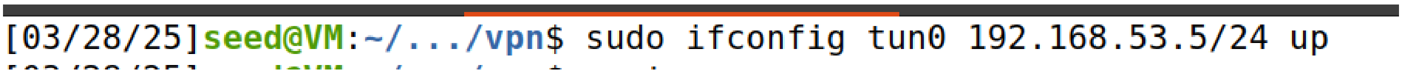
### Start the VPN Client

Next, launch the **VPN Client** and configure the IP address for its interface. Then, execute the `vpnclient.c` code on the client machine to establish the VPN connection



```
[03/28/25]seed@VM:~/.../vpn$ sudo ./vpnclient
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
```

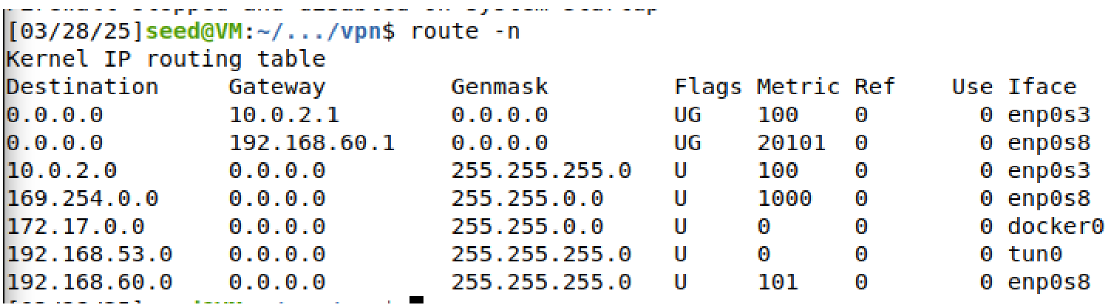
Then we assign an IP address to the tun0 inferface and activate it. IP Address assigned: 192.168.53.5/24



```
[03/28/25]seed@VM:~/.../vpn$ sudo ifconfig tun0 192.168.53.5/24 up
```

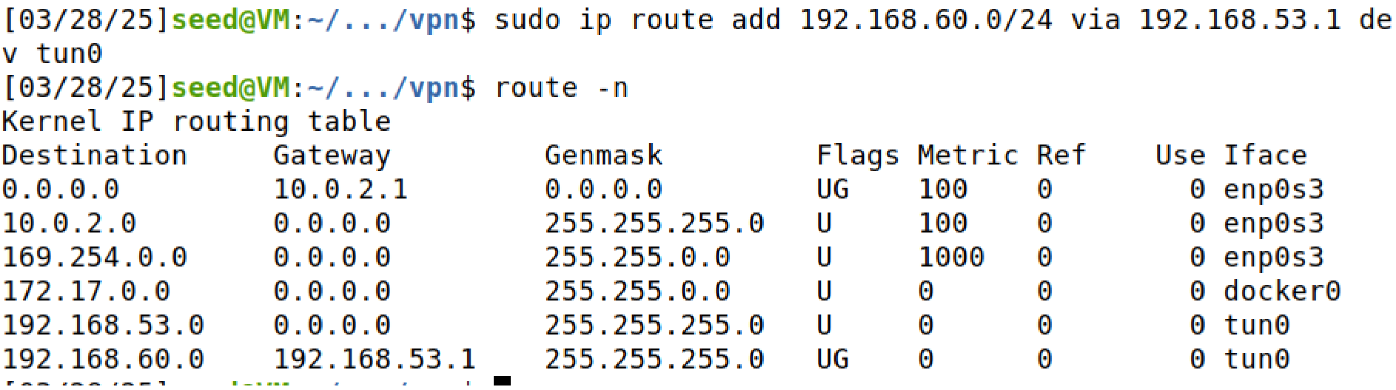
### Step 3: setting up routing table in client and server

VPN Server routing table



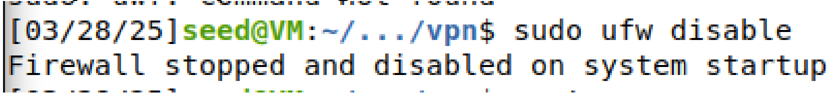
```
[03/28/25]seed@VM:~/.../vpn$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG    100    0        0 enp0s3
0.0.0.0         192.168.60.1    0.0.0.0         UG    20101  0        0 enp0s8
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 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     101    0        0 enp0s8
```

VPN Client routing table



```
[03/28/25]seed@VM:~/.../vpn$ sudo ip route add 192.168.60.0/24 via 192.168.53.1 de
v tun0
[03/28/25]seed@VM:~/.../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    192.168.53.1    255.255.255.0   UG    0      0        0 tun0
```

**Step 4: Set up routing on HOST**



```
[03/28/25]seed@VM:~/.../vpn$ sudo ufw disable
Firewall stopped and disabled on system startup
```

## Configuring Routing on Host V

1. **Disable the firewall** to avoid any disruptions:

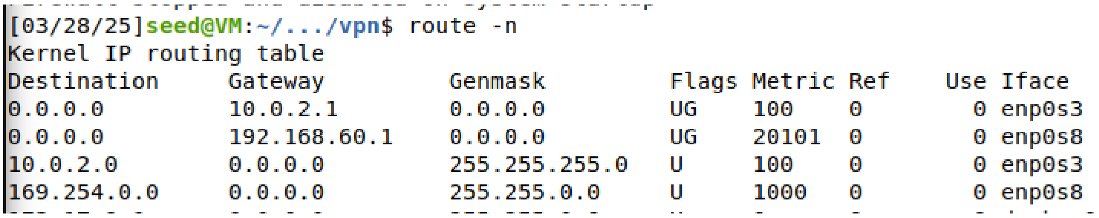
   ```
   sudo ufw disable
   ```

2. **Set up a route** to send traffic for the VPN network (`192.168.53.0/24`) through the appropriate gateway:

   ```
   sudo ip route add 192.168.53.0/24 via 192.168.60.1 dev enp0s3
   ```

3. **Check the routing table** to confirm the route has been successfully added:

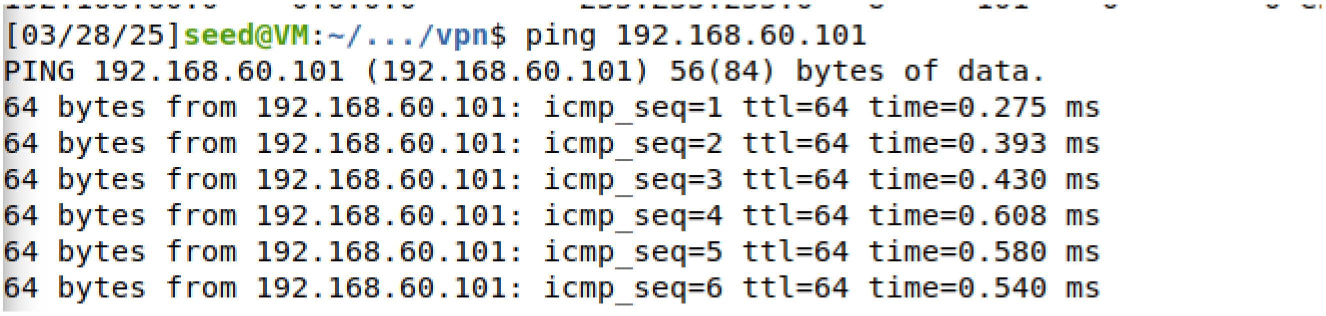
   ```
   route -n
   ```



```
[03/28/25]seed@VM:~/.../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
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 enp0s8
```

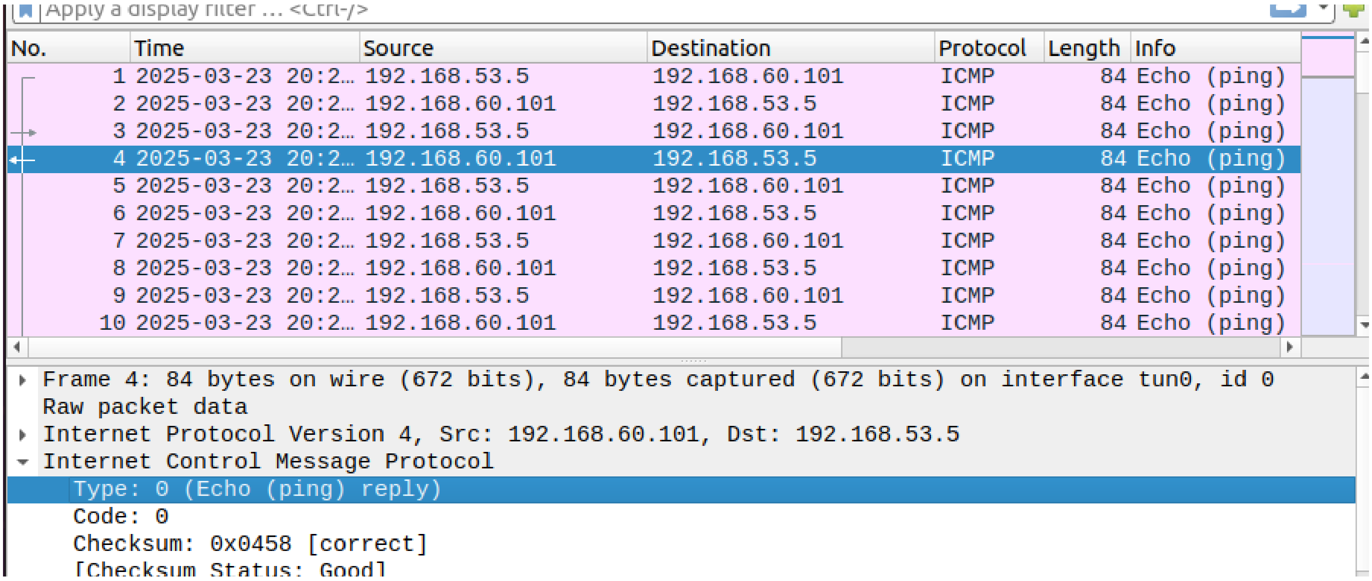
## Step 5: Testing the VPN Tunnel (Ping and Telnet)

To verify that the **VPN tunnel** is successfully established, we first execute the **ping command** to check connectivity.



```
[03/28/25]seed@VM:~/.../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=64 time=0.275 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=64 time=0.393 ms
64 bytes from 192.168.60.101: icmp_seq=3 ttl=64 time=0.430 ms
64 bytes from 192.168.60.101: icmp_seq=4 ttl=64 time=0.608 ms
64 bytes from 192.168.60.101: icmp_seq=5 ttl=64 time=0.580 ms
64 bytes from 192.168.60.101: icmp_seq=6 ttl=64 time=0.540 ms
```

Connectivity has been successfully established, as confirmed by the **ping response**. The **Wireshark screenshot** offers a detailed view of the **ICMP packet exchange**, demonstrating communication between the source and destination through the **VPN tunnel**.

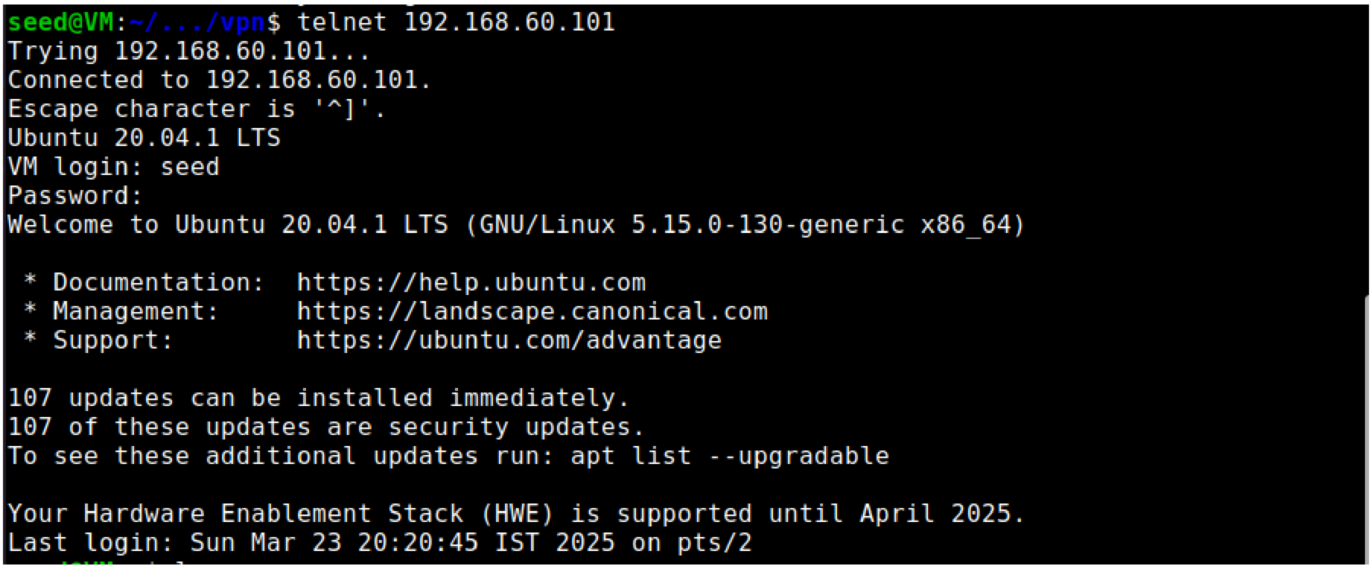


| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 2025-03-23 20:2… | 192.168.53.5 | 192.168.60.101 | ICMP | 84 | Echo (ping) |
| 2 | 2025-03-23 20:2… | 192.168.60.101 | 192.168.53.5 | ICMP | 84 | Echo (ping) |
| 3 | 2025-03-23 20:2… | 192.168.53.5 | 192.168.60.101 | ICMP | 84 | Echo (ping) |
| 4 | 2025-03-23 20:2… | 192.168.60.101 | 192.168.53.5 | ICMP | 84 | Echo (ping) |
| 5 | 2025-03-23 20:2… | 192.168.53.5 | 192.168.60.101 | ICMP | 84 | Echo (ping) |
| 6 | 2025-03-23 20:2… | 192.168.60.101 | 192.168.53.5 | ICMP | 84 | Echo (ping) |
| 7 | 2025-03-23 20:2… | 192.168.53.5 | 192.168.60.101 | ICMP | 84 | Echo (ping) |
| 8 | 2025-03-23 20:2… | 192.168.60.101 | 192.168.53.5 | ICMP | 84 | Echo (ping) |
| 9 | 2025-03-23 20:2… | 192.168.53.5 | 192.168.60.101 | ICMP | 84 | Echo (ping) |
| 10 | 2025-03-23 20:2… | 192.168.60.101 | 192.168.53.5 | ICMP | 84 | Echo (ping) |

```
▸ Frame 4: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface tun0, id 0
  Raw packet data
▸ Internet Protocol Version 4, Src: 192.168.60.101, Dst: 192.168.53.5
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x0458 [correct]
    [Checksum Status: Good]
```

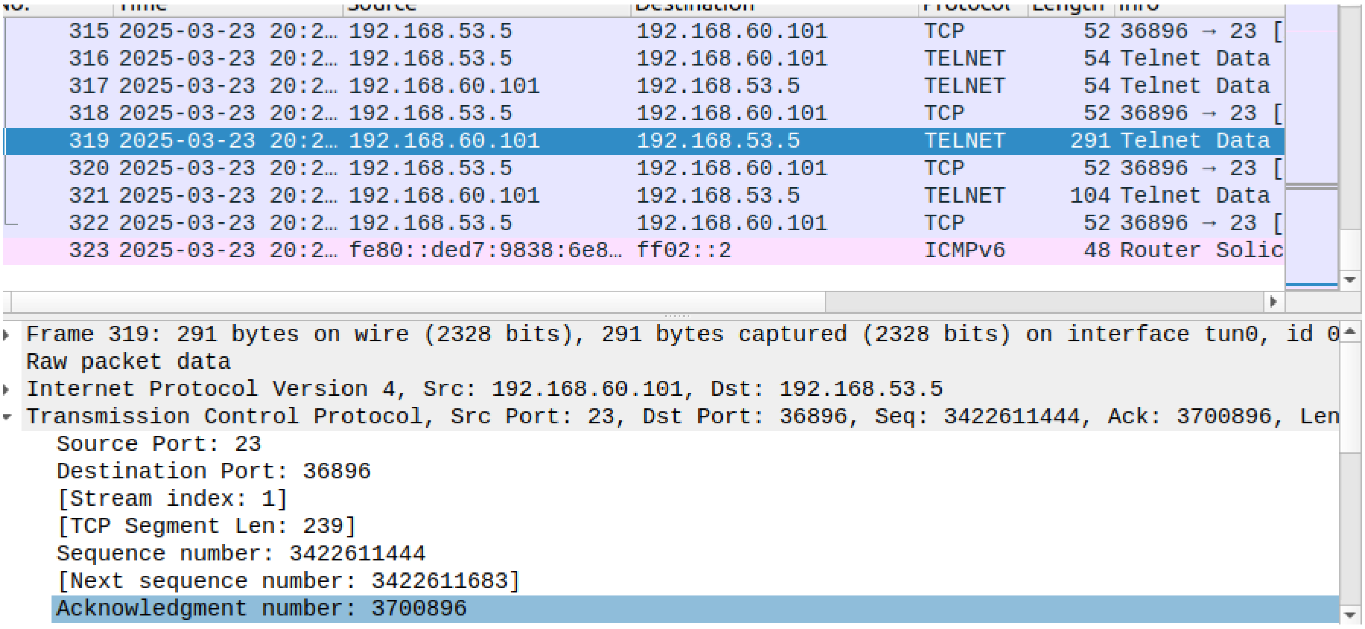
From the **Wireshark capture**, we can see that packets originating from **192.168.53.5** (Client - tun0) and destined for **192.168.60.101** (Host V) are part of the **tunnel traffic.** The rest of the packets belong to regular network communication.

Next, we will initiate a **Telnet connection** to confirm that the **VPN tunnel is working correctly**.



```
seed@VM:~/.../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
VM login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-130-generic x86_64)

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

107 updates can be installed immediately.
107 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Mar 23 20:20:45 IST 2025 on pts/2
```

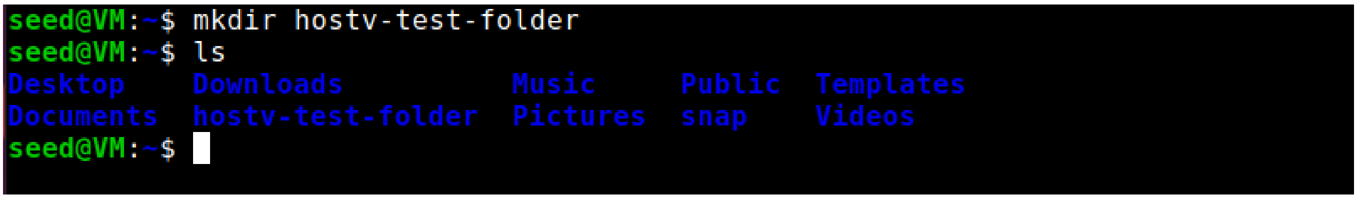
We have successfully established the **Telnet connection**, as evidenced by the **Wireshark screenshot** confirming the connection.



```
315 2025-03-23 20:2… 192.168.53.5         192.168.60.101   TCP        52 36896 → 23 [
316 2025-03-23 20:2… 192.168.53.5         192.168.60.101   TELNET     54 Telnet Data
317 2025-03-23 20:2… 192.168.60.101       192.168.53.5     TELNET     54 Telnet Data
318 2025-03-23 20:2… 192.168.53.5         192.168.60.101   TCP        52 36896 → 23 [
319 2025-03-23 20:2… 192.168.60.101       192.168.53.5     TELNET    291 Telnet Data
320 2025-03-23 20:2… 192.168.53.5         192.168.60.101   TCP        52 36896 → 23 [
321 2025-03-23 20:2… 192.168.60.101       192.168.53.5     TELNET    104 Telnet Data
322 2025-03-23 20:2… 192.168.53.5         192.168.60.101   TCP        52 36896 → 23 [
323 2025-03-23 20:2… fe80::ded7:9838:6e8… ff02::2          ICMPv6     48 Router Solic
```

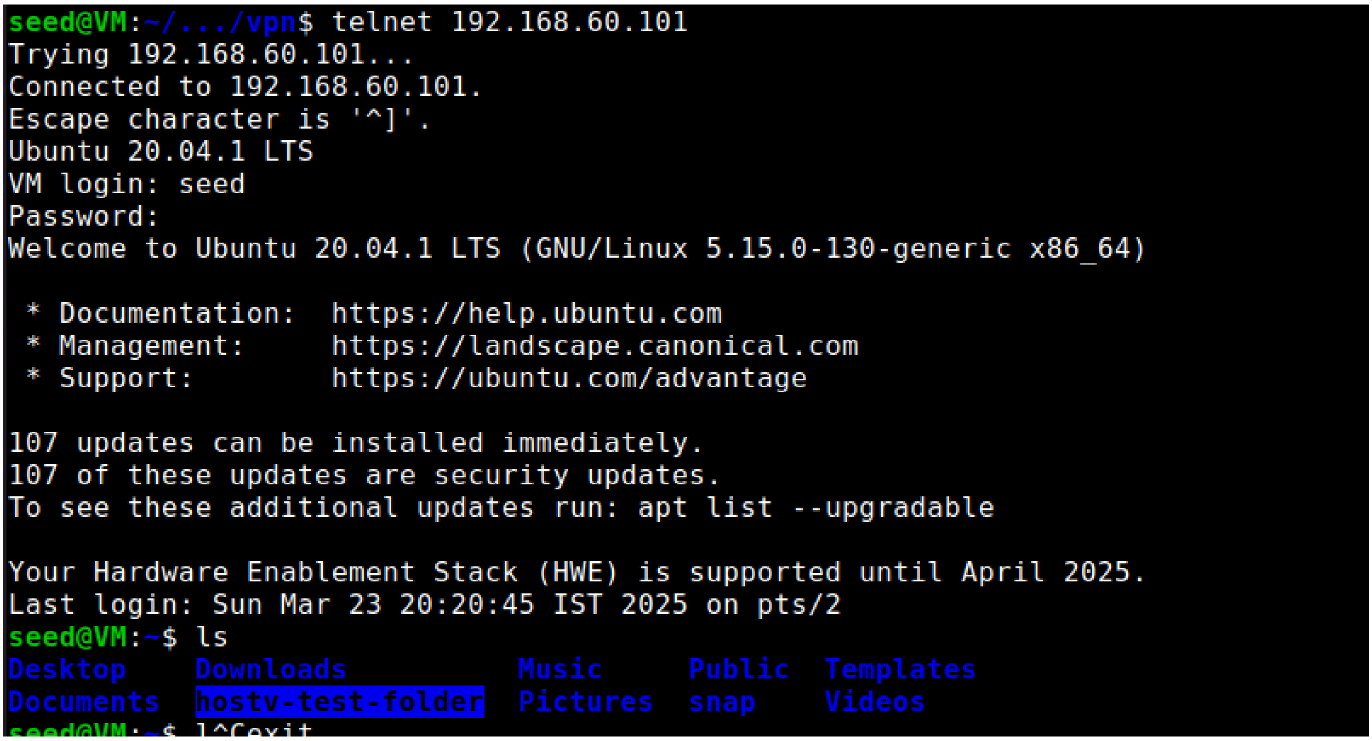
```
Frame 319: 291 bytes on wire (2328 bits), 291 bytes captured (2328 bits) on interface tun0, id 0
Raw packet data
Internet Protocol Version 4, Src: 192.168.60.101, Dst: 192.168.53.5
Transmission Control Protocol, Src Port: 23, Dst Port: 36896, Seq: 3422611444, Ack: 3700896, Len
    Source Port: 23
    Destination Port: 36896
    [Stream index: 1]
    [TCP Segment Len: 239]
    Sequence number: 3422611444
    [Next sequence number: 3422611683]
    Acknowledgment number: 3700896
```

The screenshot verifies that the **VPN connection was successfully set up**. To further confirm access, we executed the `ls` command on the **VPN Host** and created a new folder named **hostv-test-folder**, as depicted in the screenshot.



```
seed@VM:~$ mkdir hostv-test-folder
seed@VM:~$ ls
Desktop    Downloads          Music     Public   Templates
Documents  hostv-test-folder  Pictures  snap     Videos
seed@VM:~$ 
```

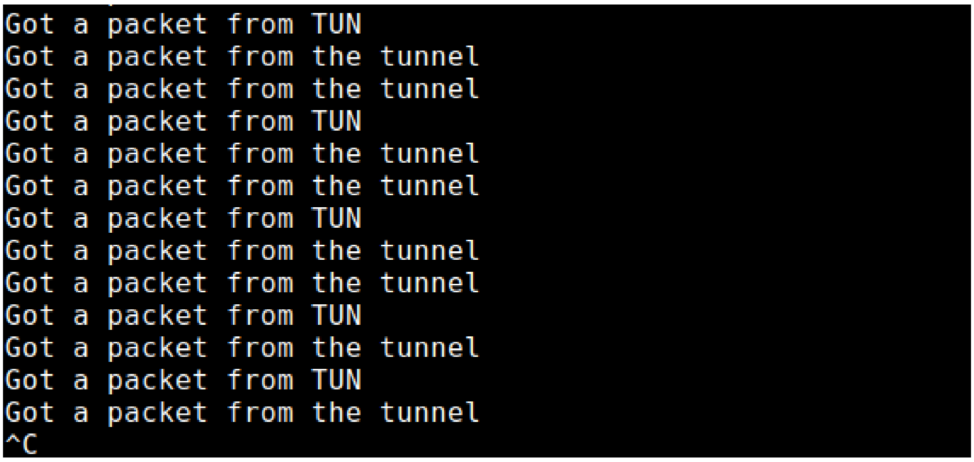
Now when we run ‘ls’ command on the telnet connection, we are able to notice that the new folder create is visible:



```
seed@VM:~/.../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
VM login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-130-generic x86_64)

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

107 updates can be installed immediately.
107 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Mar 23 20:20:45 IST 2025 on pts/2
seed@VM:~$ ls
Desktop    Downloads         Music     Public  Templates
Documents  hostv-test-folder Pictures  snap    Videos
seed@VM:~$ ]^Cexit
```

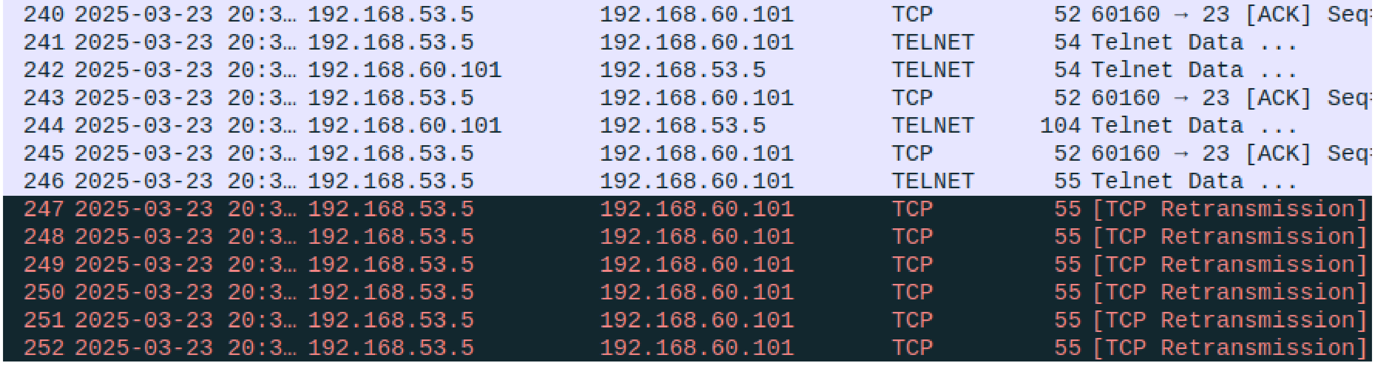
## Step 6: Tunnel-Breaking Test

To test the impact of a broken VPN connection, we **terminate the `vpnserver` program**, intentionally disrupting the **VPN tunnel**, as shown in the screenshot.



```
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
^C
```

After shutting down the **VPN server**, the **Telnet connection** fails to execute the `ls` command. This confirms that the **VPN tunnel** was essential for communication, and without it, the connection is lost.



```
240 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TCP        52 60160 → 23 [ACK] Seq
241 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TELNET     54 Telnet Data ...
242 2025-03-23 20:3… 192.168.60.101      192.168.53.5       TELNET     54 Telnet Data ...
243 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TCP        52 60160 → 23 [ACK] Seq
244 2025-03-23 20:3… 192.168.60.101      192.168.53.5       TELNET    104 Telnet Data ...
245 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TCP        52 60160 → 23 [ACK] Seq
246 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TELNET     55 Telnet Data ...
247 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TCP        55 [TCP Retransmission]
248 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TCP        55 [TCP Retransmission]
249 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TCP        55 [TCP Retransmission]
250 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TCP        55 [TCP Retransmission]
251 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TCP        55 [TCP Retransmission]
252 2025-03-23 20:3… 192.168.53.5        192.168.60.101     TCP        55 [TCP Retransmission]
```

As seen in the **Wireshark capture**, a **TCP redirect message** is being received, indicating that network traffic is either being rerouted or there is a problem with the current path. This suggests that once the **VPN is disconnected**, the **Telnet connection** can no longer reach its intended destination.