Virtual Private Network (VPN) Lab Report

Mudit Vats
mpvats@syr.edu
3/9/2019

Table of Contents

Overview

This lab report presents observations and explanations for the tasks described in the Virtual Private Network (VPN) Lab.

Task 1: VM Setup

In this part, we setup our VM's for the VPN lab. The figure below, represents the VM setup (copy/pasted from the VPN Lab document).

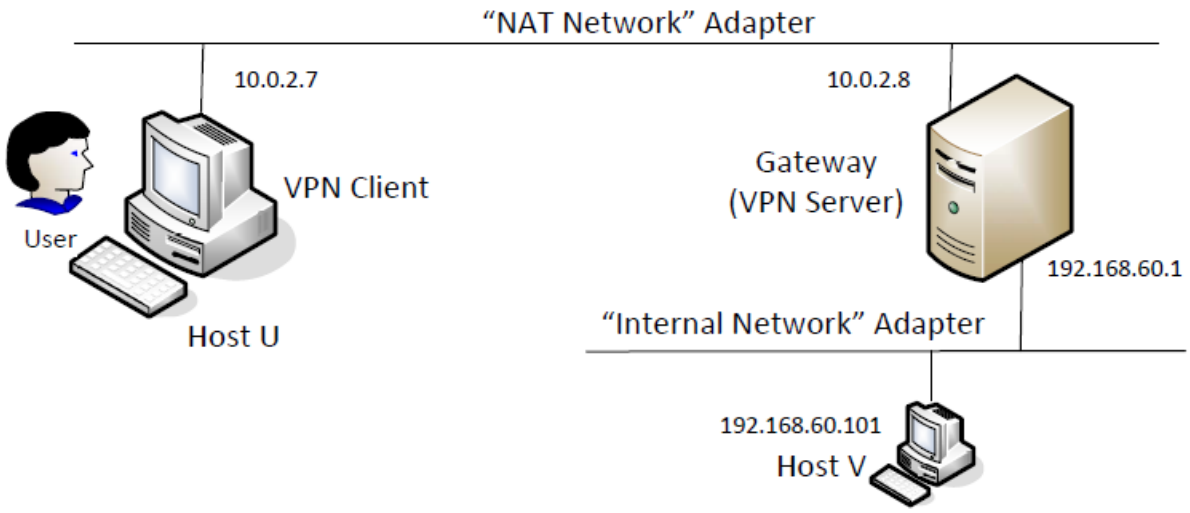


Figure 1: VM Setup

We are using these VMs:

VM Background	Role	Network	IP Address	Tunnel IP Address
Blue	Gateway (VPN Server)	NAT	10.0.2.5	192.168.53.1
		Internal Network	192.168.60.1	
Green	Host U	NAT	10.0.2.6	192.168.53.5
Red	Host V	Internal Network	192.168.60.101	N/A

Figure 2: VM Adapters

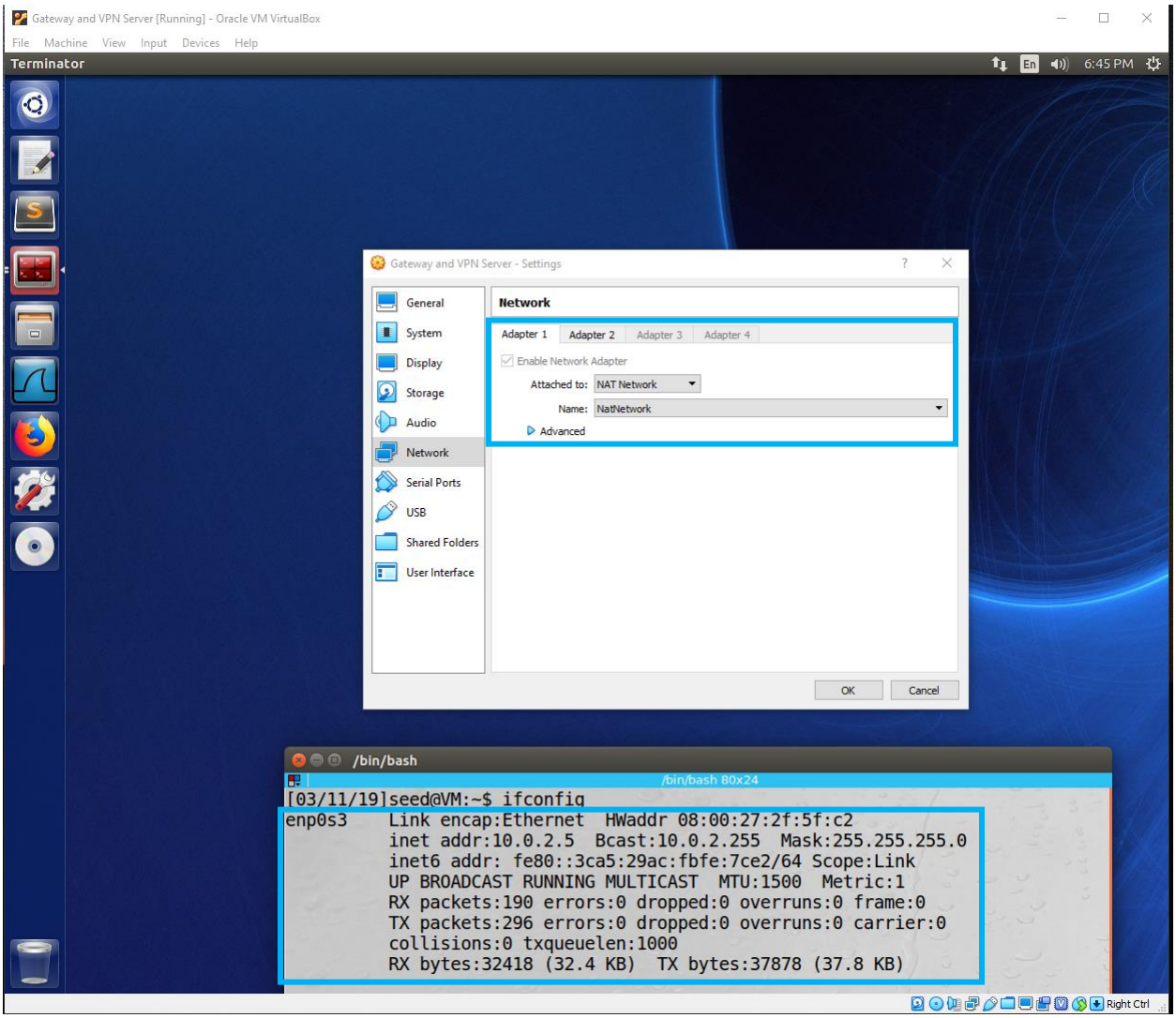
We are using these Networks (subnets):

Network	Subnet	Purpose
NAT	10.0.2.0/24	Represents the Internet. VPN Client and Gateway connect to this network.
Internal Network	192.168.60.0/24	Represents the Private Network. Gateway (VPN Server) and Host V connected to this network.
VPN Network	192.168.53.0/24	Represents the network that the VPN Tunnel interfaces use to encrypt / tunnel data.

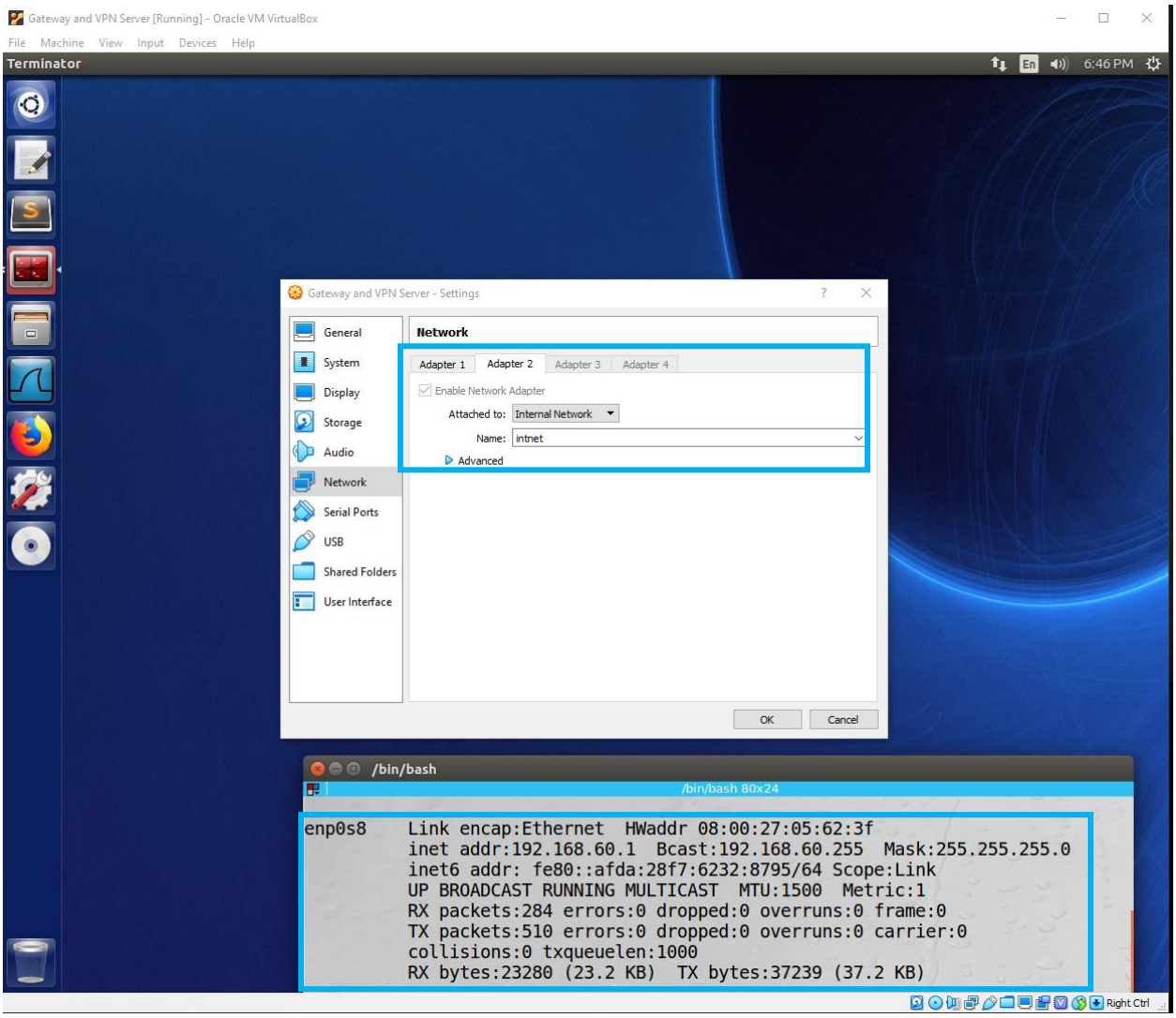
Figure 3: Networks

In the next several figures, we setup the ethernet adapters in the VMs. As indicated in the VM Setup figure, we see that the Blue VM has two ethernet adapters, the Green VM has one ethernet adapter and the Red VM has one ethernet adapter.

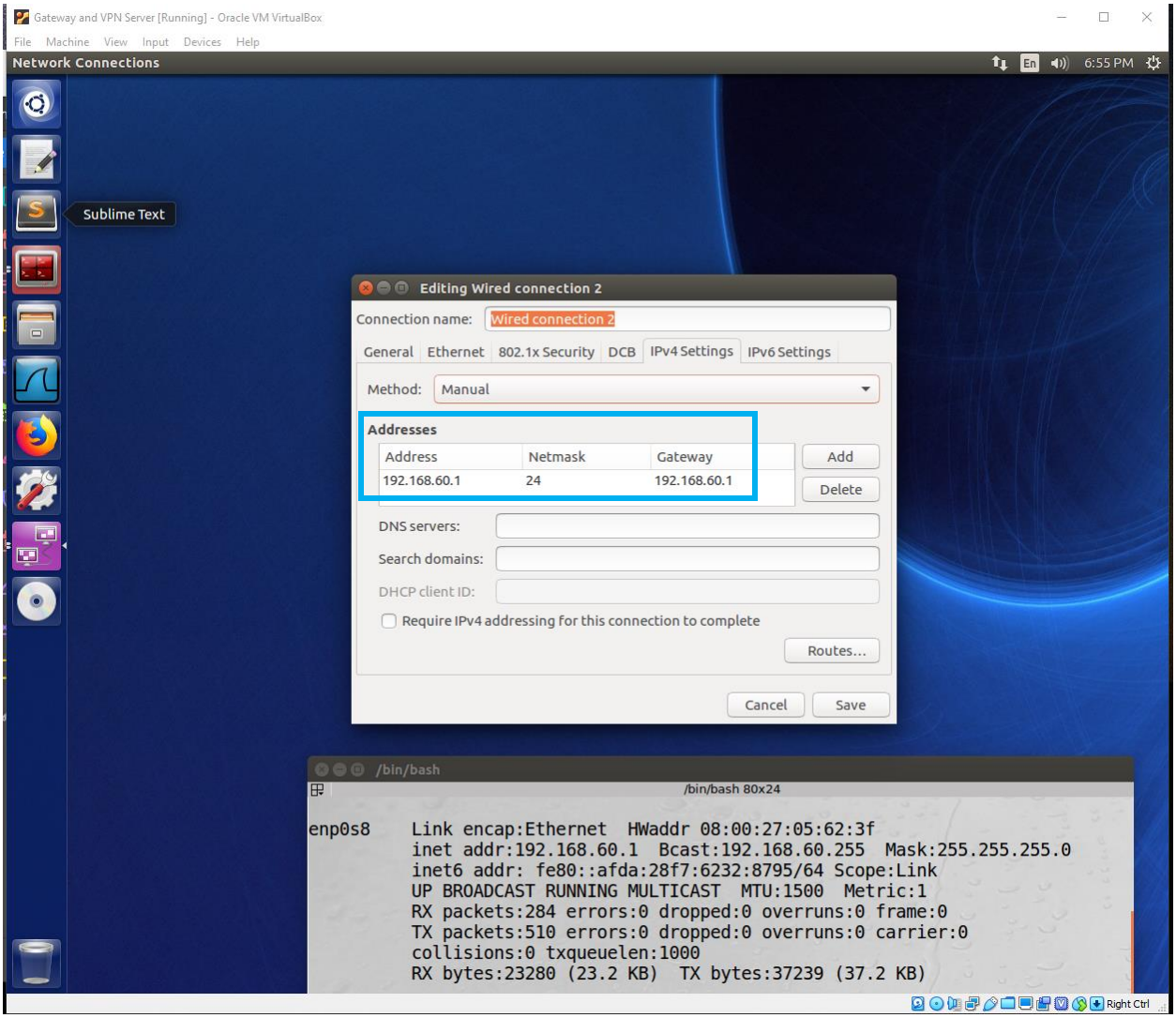
In the VM Setup figure, we see that Adapter 1 is on the NAT Network. Its adapter is enp0s3 with the IP 10.0.2.5.



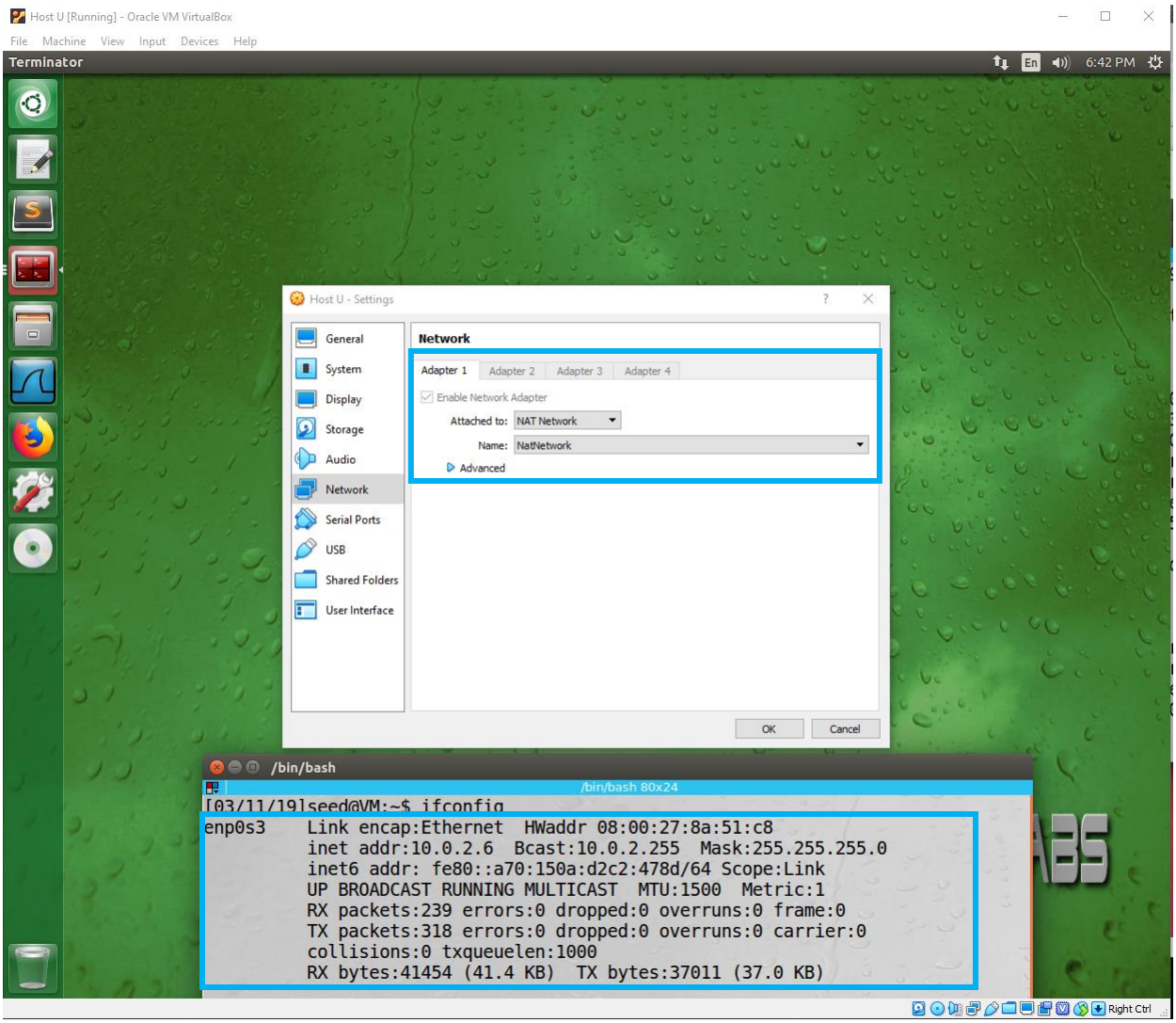
In the figure below, we see the second adapter, Adapter 2, which is on the Internal Network. Its adapter is enp0s8 with the IP 192.168.60.1.



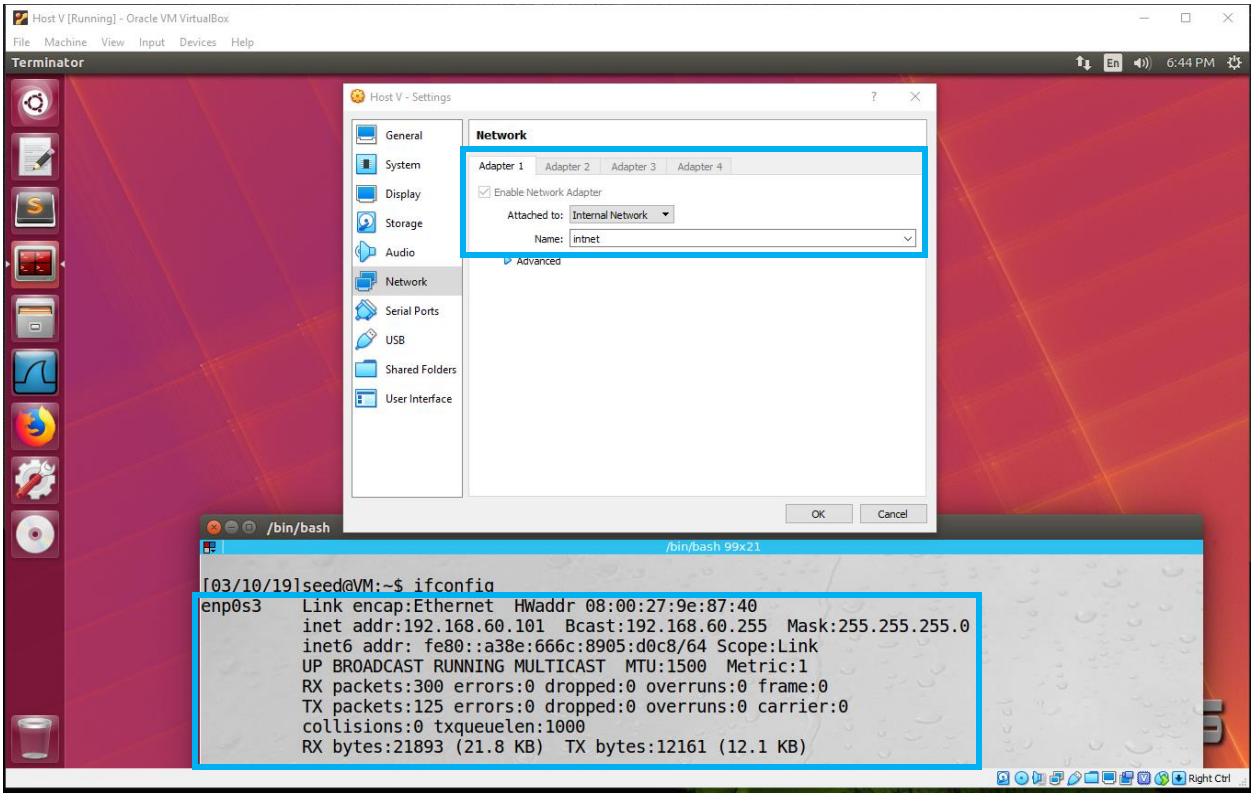
In the figure below, we see the configuration for the second adapter, Adapter 2, which is on the Internal Network. Its adapter is manually set with the IP 192.168.60.1/24 and Gateway 192.168.60.1 (which is the same VM).



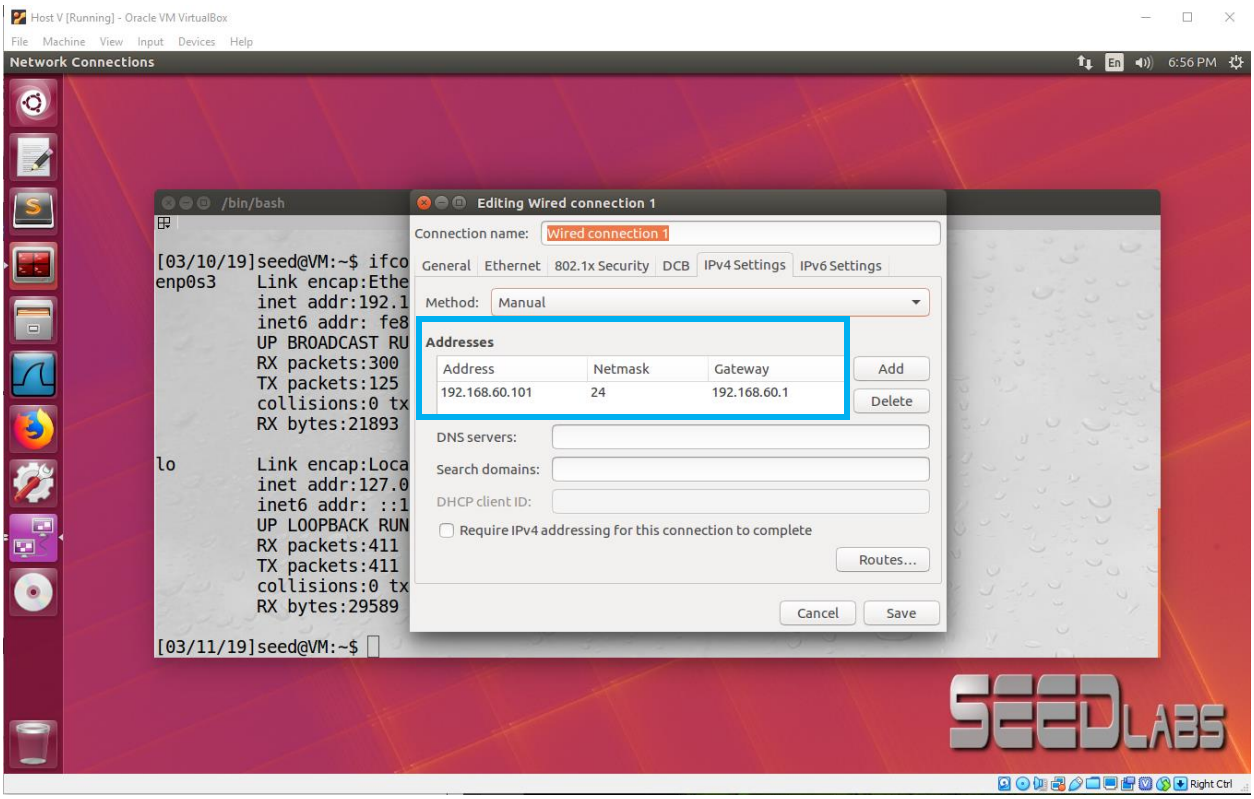
In the following figure, we see the Green VM which is the Host U. The Host U VM has an Adapter 1 on the NAT Network. Its adapter is enp0s3 with the IP 10.0.2.6.



In the figure below, we see the Red VM which is the Host V VM. The Host V VM has an Adapter 1 which is on the Internal Network. Its adapter is enp0s3 with the IP 192.168.60.101.



In the figure below, we see the configuration for the adapter, Adapter 1, which is on the Internal Network. Its adapter is manually set with the IP 192.168.60.101/24 and Gateway 192.168.60.1 (which is the Blue VM).



Observations / Explanations

In this task we setup the VM's as previous described in the Figure 1: VM Setup and Figure 2: Adapters.

The setup is such that Host U and VPN Server/Gateway are connected via the NAT Network representing the Internet Network. They will create a tunnel between each other for secure (VPN) communication. They also will have tunnel adapters to help facilitate packets going into the VPN tunnel and coming out of the VPN tunnel. We do both of these in the next step. We also have the Host V and VPN Server/Gateway connected via the Internal Network. This network represents our private network. Only Host V and VPN Server/Gateway are on this network.

After this VM setup, we have our network adapters (not including the tunnel adapters), our NAT/Internet Network and Internal/Private Network in place. While this is interesting, without routes properly setup, Host U cannot communicate with Host V. We'll do that, plus the tunnel interface setup next.

Task 2: Creating a VPN Tunnel using TUN/TAP

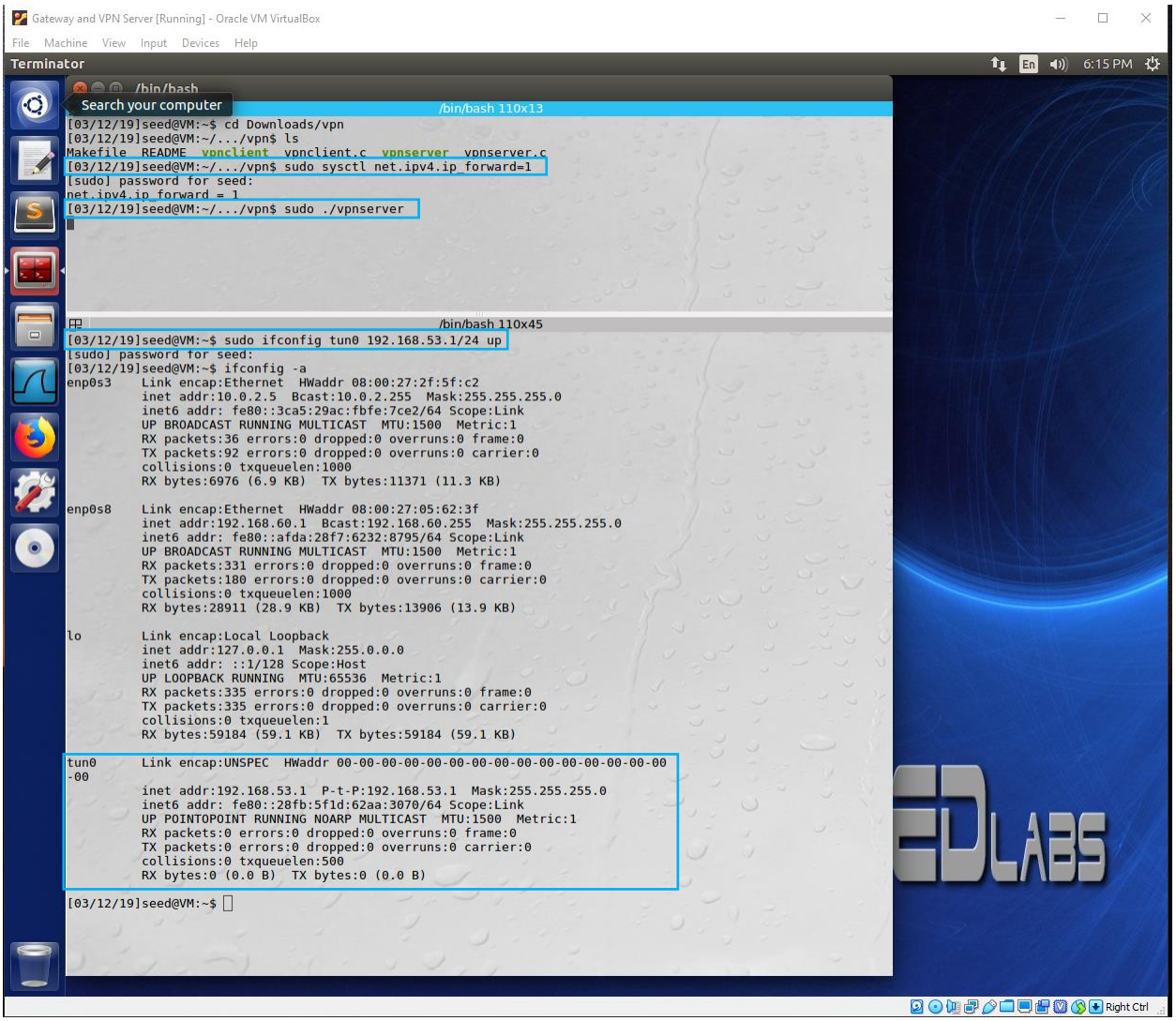
In this task create a tunnel using the TUN/TAP interface and test it out. We also setup routes within the routing tables for the VMs so they can communicate to each other.

Step 1: Run VPN Server

In the figure below, we first see us enabling IP forward. This is necessary for this VPN Server / Gateway VM to forward packets between network interfaces.

Thereafter, you see on the top half of the window that we start the vpnserver which creates the tunnel adapter. On the bottom half, we see the configuration for the tunnel adapter. We set the IP to 192.168.53.1 and tell it to start by specifying the "up" option. Finally, we can see that the assignment was correctly applied by using "ifconfig -a" to see all the adapters, including the tun0 adapter. The command below is what was used to configure the tunnel (tun0) adapter.

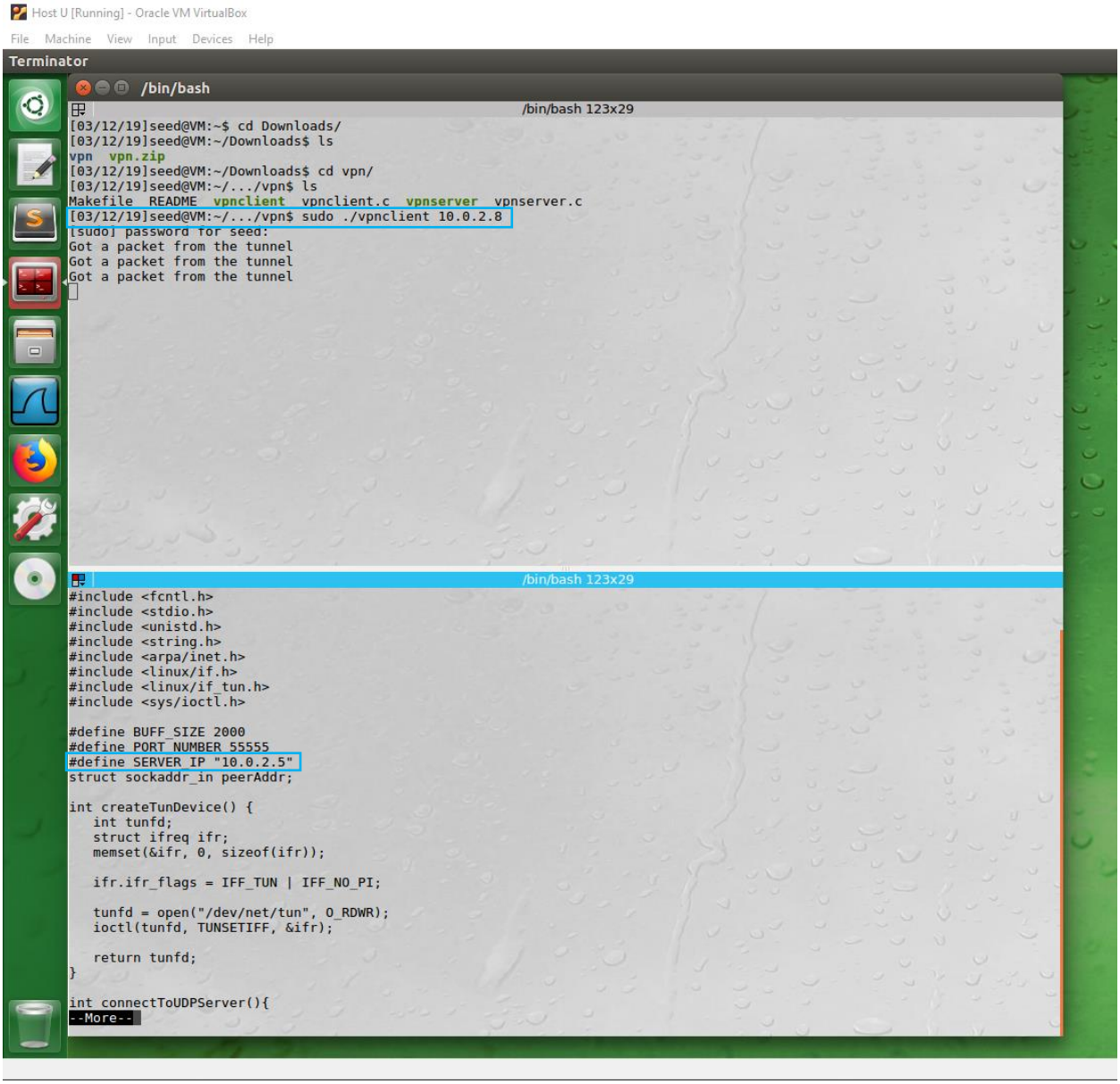
- sudo ifconfig tun0 192.168.53.1/24 up



Step 2: Run VPN Client

In this figure, we start the vpnclient.

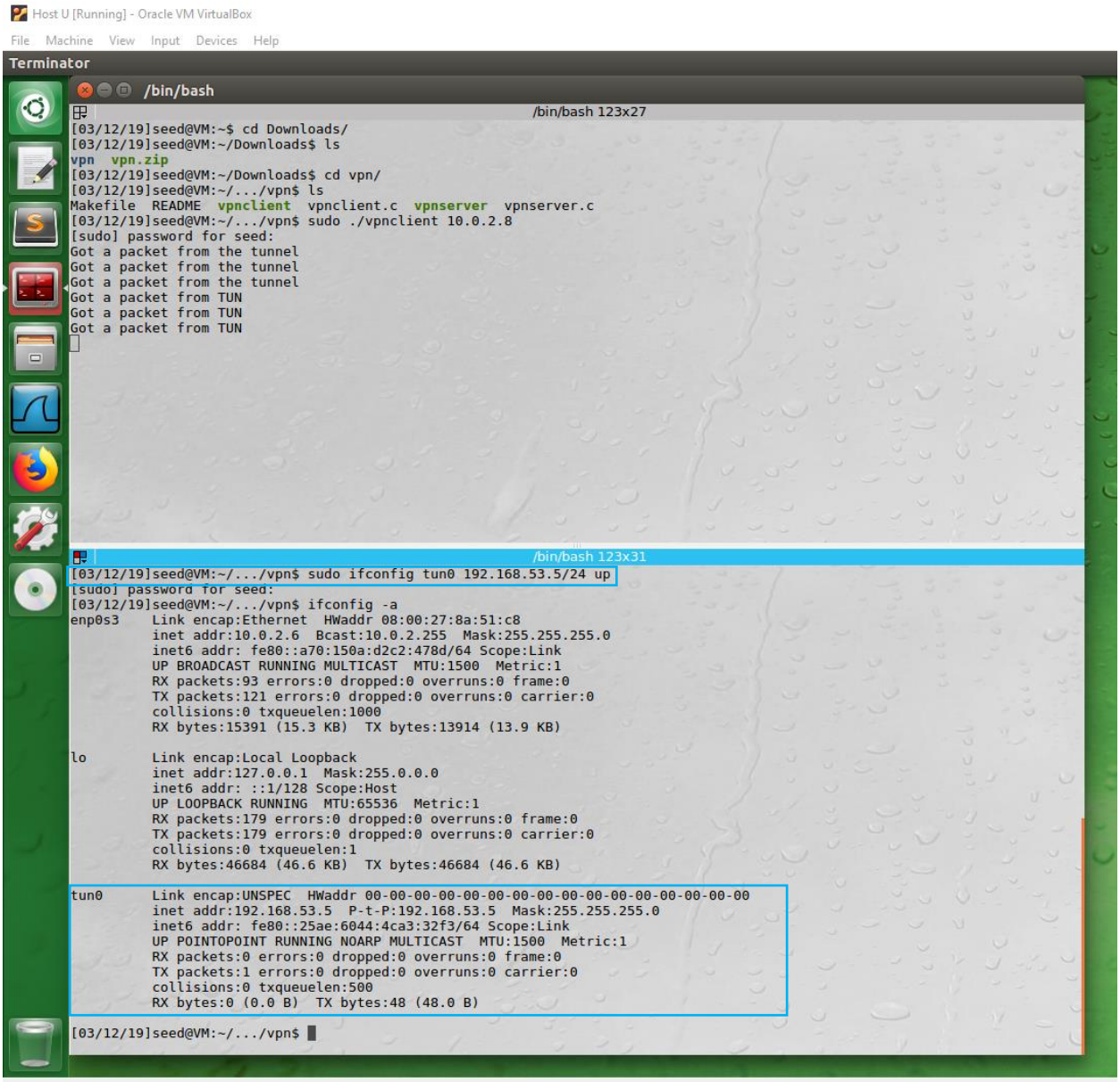
Please note, while the IP address of the VPN Server is passed as an argument to vpnclient, it's not needed. The VPN Server IP is hardcoded in the vpnclient.c file as shown in the figure below. It's only there due to a copy/paste error and can be safely ignored (as is ignored in the program).



In the figure below, you see on the top half of the window that we start the vpnclient (same as figure above) which creates the tunnel adapter. On the bottom half, which is different, we see the configuration for the tunnel adapter. We set the IP to 192.168.53.5 and tell it to start by specifying the "up" option. Finally, we can see that the assignment was correctly applied by using "ifconfig -a" to see all the adapters,

including the tun0 adapter. The command below is what was used to configure the tunnel (tun0) adapter.

- sudo ifconfig tun0 192.168.53.5/24 up

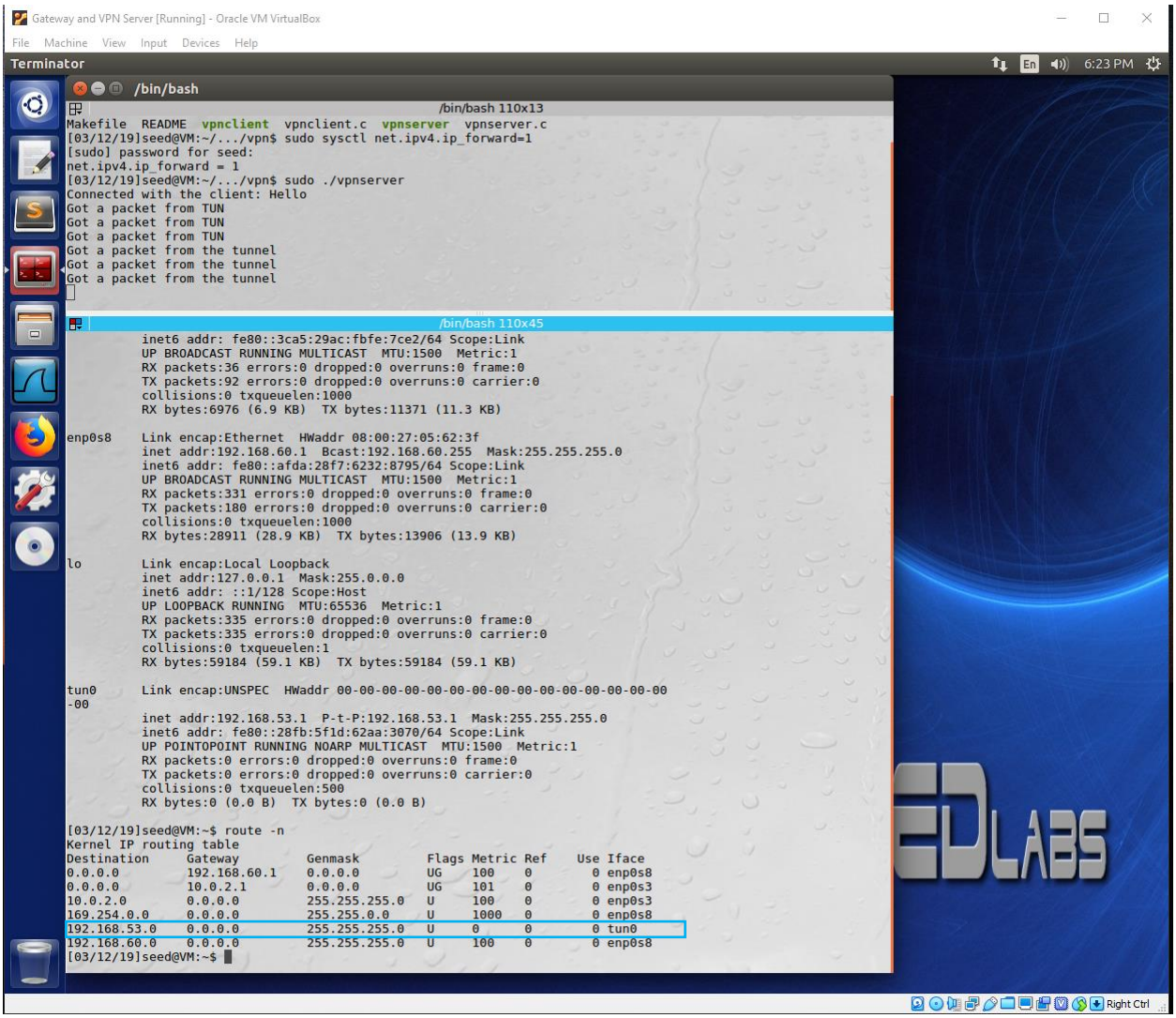


Step 3: Set Up Routing on Client and Server VMs

In this step we setup the routes for the Host U and the VPN Server/Gateway VMs.

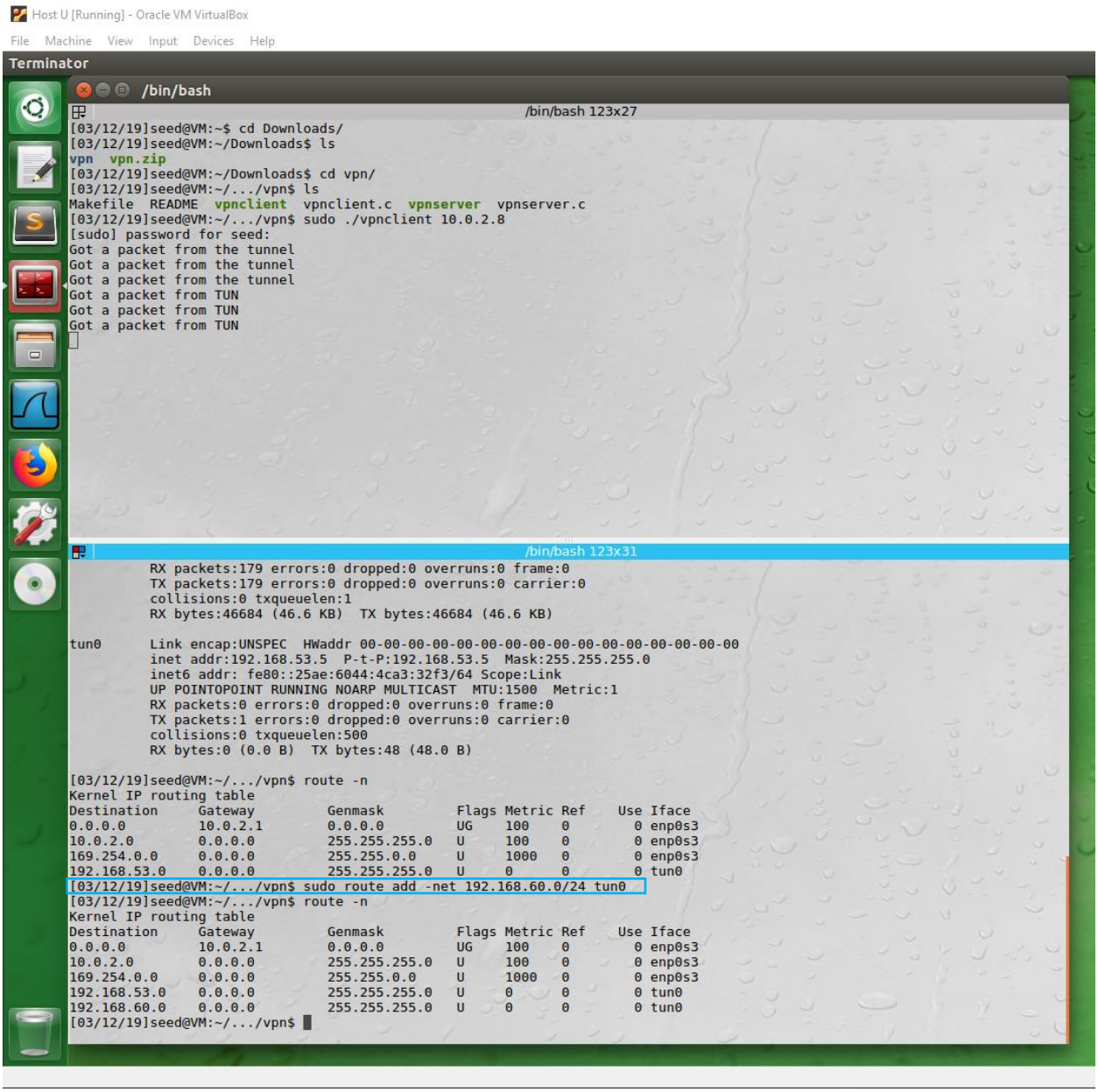
In the figure below, we see the route for VPN tunnel already created for the VPN Server / Gateway VM. It's setup to forward any packets destined for 192.168.53.0/24 subnet to the tunnel adapter. This was setup automatically when we configure the tunnel adapter's IP address. This route is needed so that the VPN Server / Gateway knows how to handle packets destined for the VPN network. If the route was not setup automatically, we would issue the command below.

- sudo route add -net 192.168.53.0/24 tun0



In the figure below, we set the route for the Host U VM to forward any packets destined for the 192.168.60.0/24 network to the tunnel adapter. Recall that this network is the Internal / Private Network. This route is needed so that the Host U knows how to handle packets destined for Private Network. Without this, packets from Host U would not be able to find its way to Host V, for example. They need to know the next hop and this route, provides that "next hop" information to route the packet.

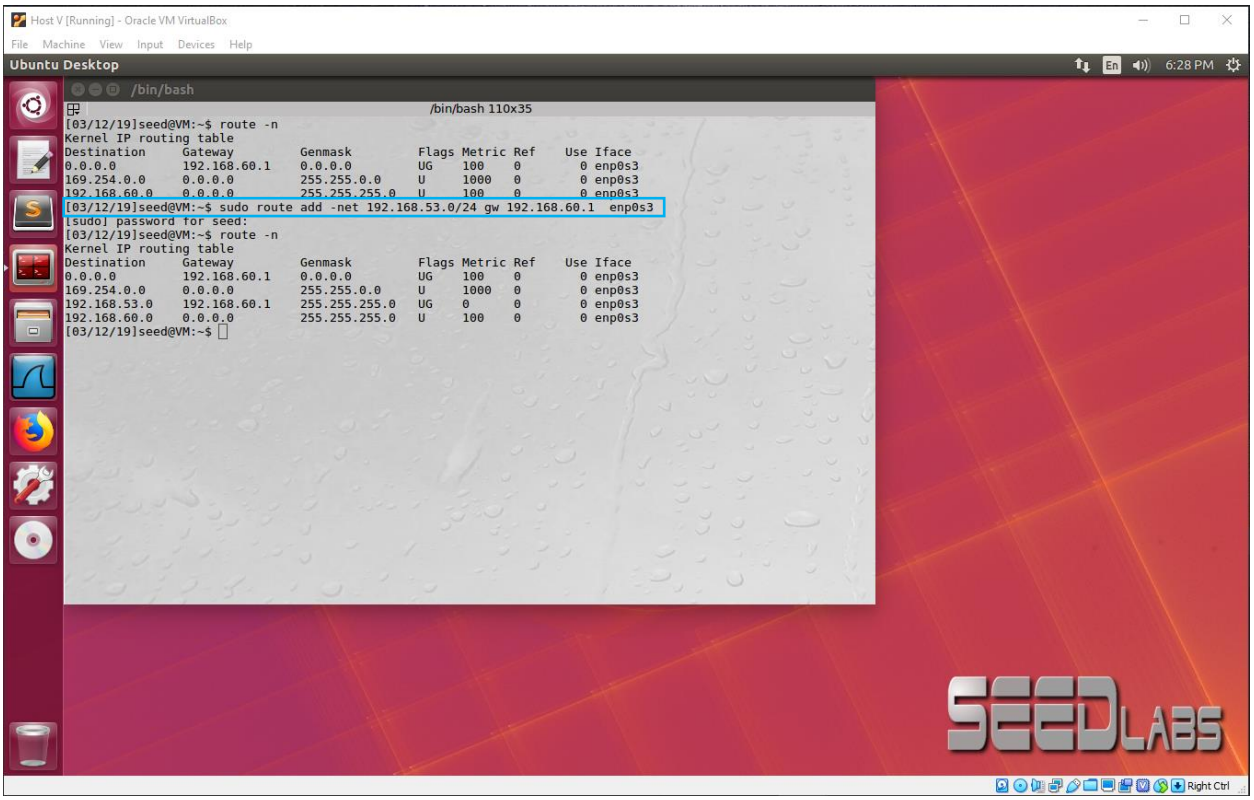
- sudo route add -net 192.168.60.0/24 tun0



Step 4: Setup Up Routing on Host V

In the figure below, we set the routing on the Host V system so that any packets destined for the VPN Network will be routed to the VPN Server / Gateway through the network adapter. This route is needed otherwise replies to packets from Host U, for example, would not be able to be reached since the destination IP form Host U would become the source IP and that would be on the 192.168.53.x network (192.168.53.6 to be precise). Providing the Gateway's IP specifies that we wish the Gateway to be the next hop to handle the packet going to the 192.168.53.0/24 network.

- sudo route add -net 192.168.53.0/24 gw 192.168.60.1 enp0s3

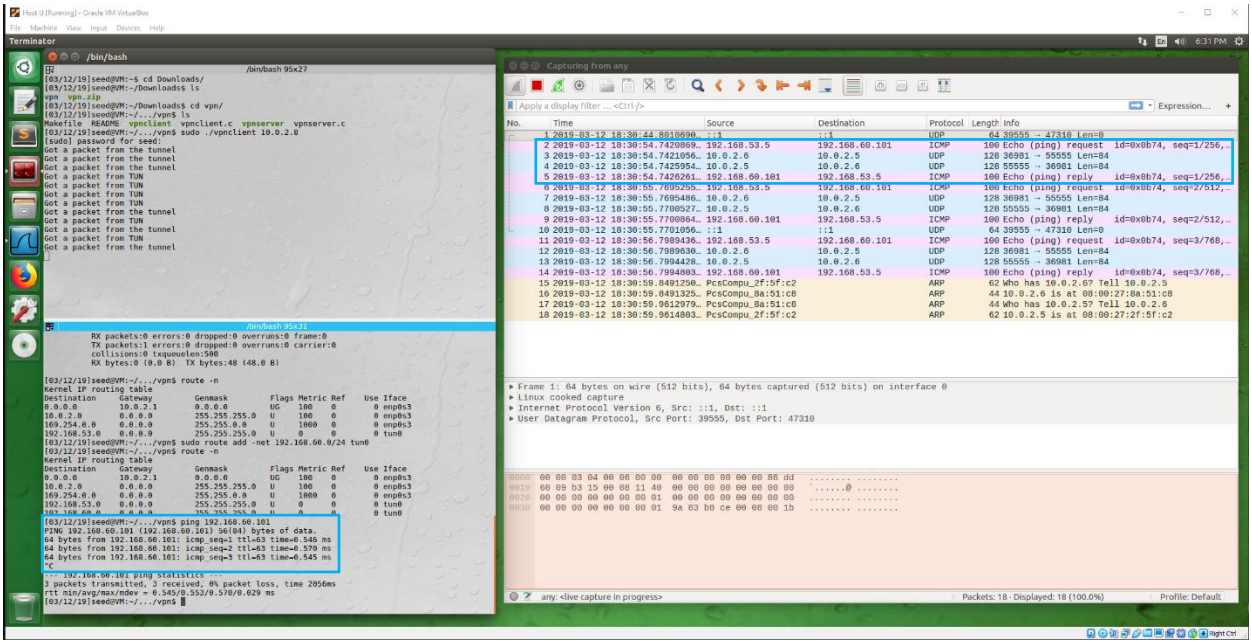


Step 5: Test the VPN Tunnel

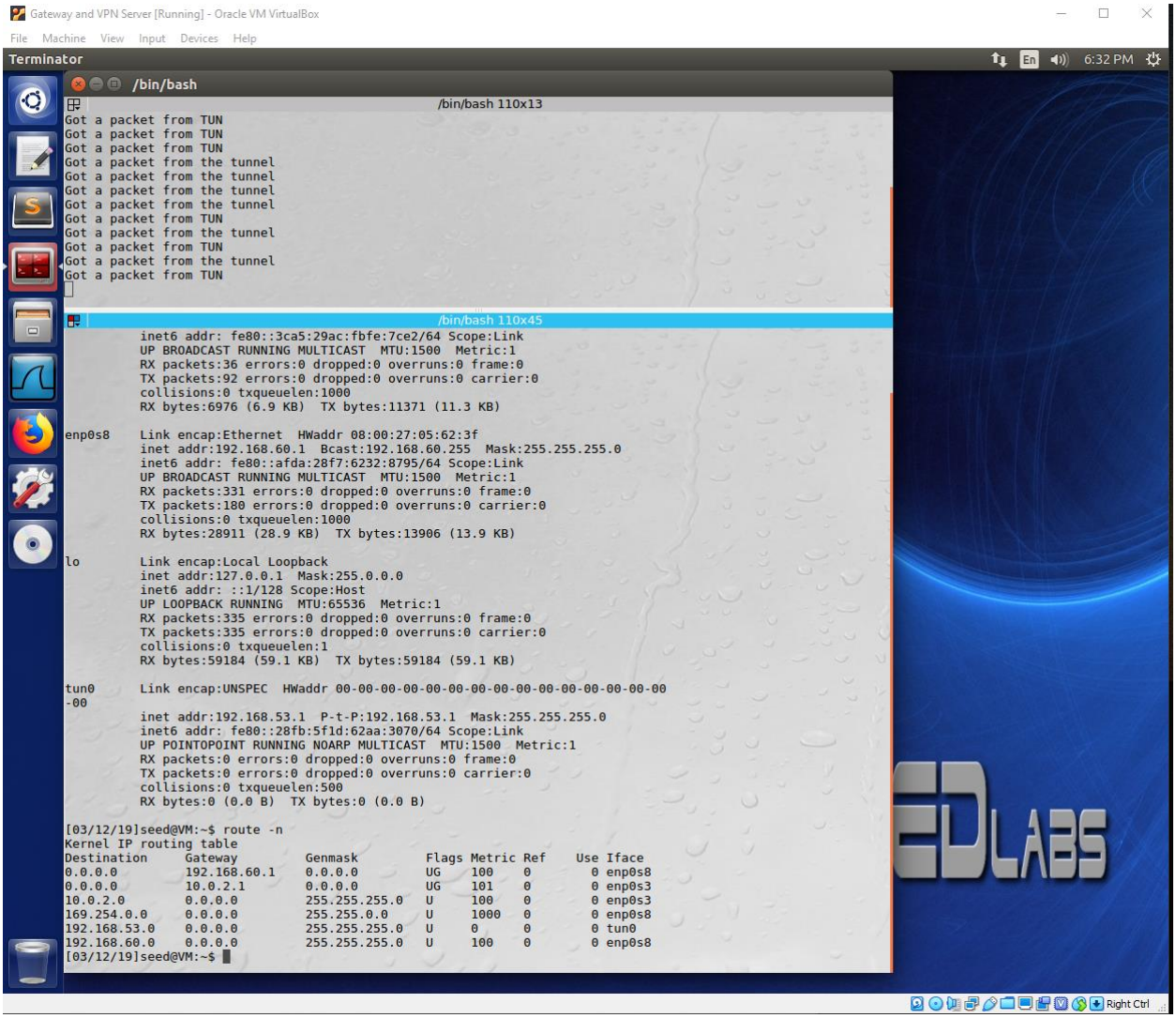
Ping Test

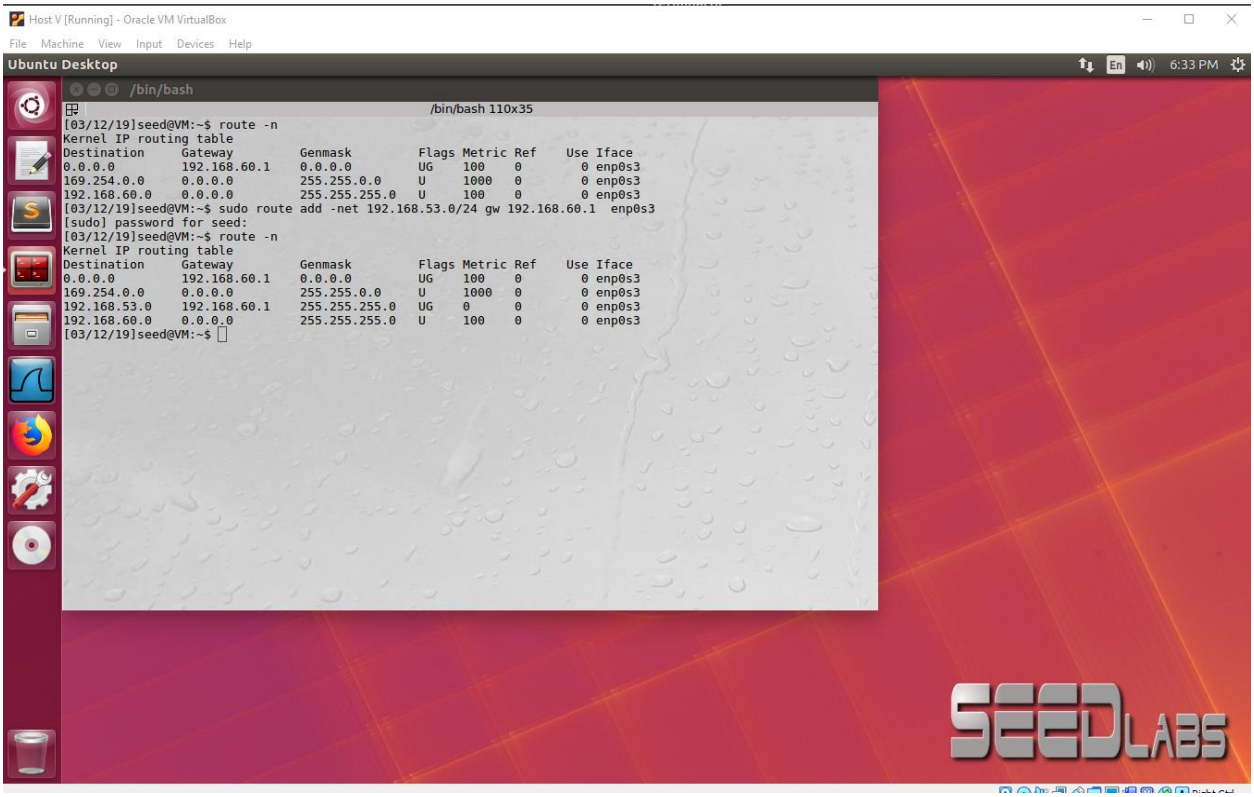
In the figure below, we ping form Host U to Host V. This can be seen in the bottom half of the terminal window. On the right, we have the WireShark capture of the network traffic. We ping from 192.168.53.5 (Host U/tun0) to 192.168.60.101 (Host V) and can see the following packet sequence –

1. **LINE 2:** We see the packet going from 192.168.53.5 (Host U's tun0 IP address) to 192.168.60.101, the final destination. **NOT TUNNEL.**
2. **LINE 3:** We then see the packet transmitted from Host U's ethernet adapter 10.0.2.6 to the VPN Server / Gateway's IP address 10.0.2.5. This is the UDP VPN client to VPN server communication. Please note, the original ping packet is a payload of this tunnel packet. **TUNNEL**.
3. **LINE 4:** We then see a packet returned from the VPN Server / Gateways IP address 10.0.2.5 to the Host U ethernet adapter 10.0.2.6. This is the UDP VPN server to VPN client communication. **TUNNEL.**
4. **LINE 5:** Finally, we see that the VPN client releases the received payload into the network Host U's stack. This payload is a response to the ping and it's sent from 192.168.60.101 to 192.168.53.5, which is right back to the Host U's IP address. **NOT TUNNEL.**



In the next two figures, we see VPN Server / Gateway VM and Host V. In the VPN Server / Gateway VM, we see packets from the tunnel interface and from the VPN tunnel. The Host V merely replies to pings or (later) acts as a telnet server, so there's not much to see there. These are the screenshots regardless.

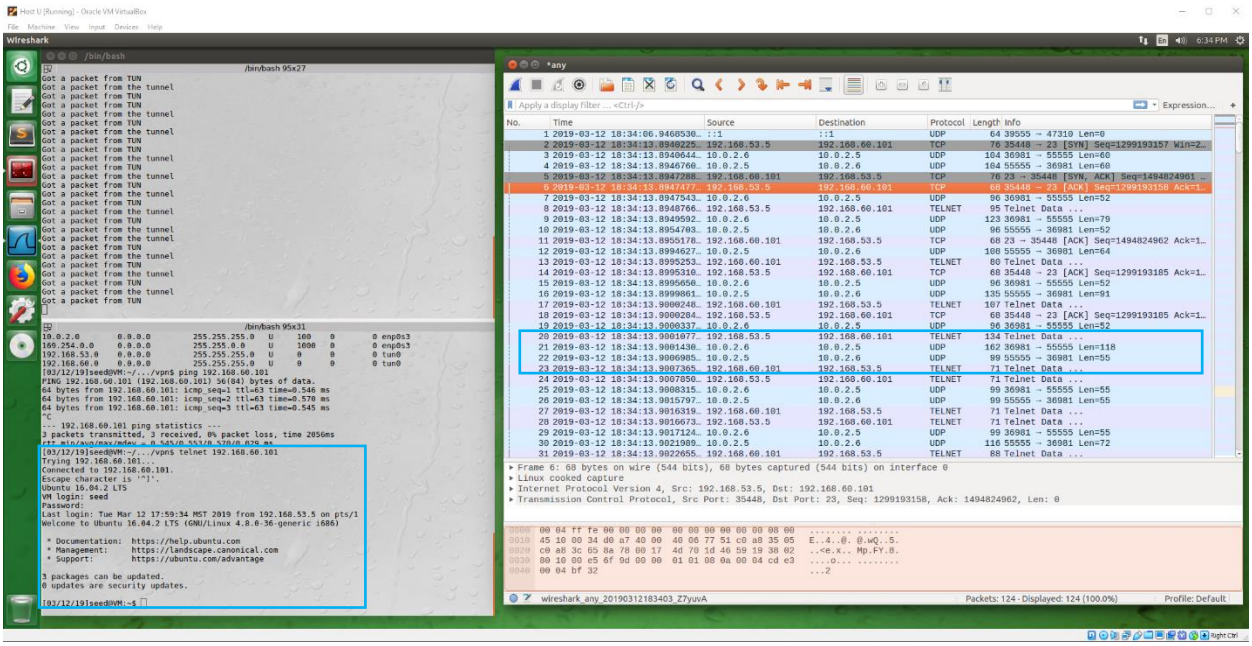

Gateway and VPN Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
6:32 PM
/bin/bash
/bin/bash 110x13
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
/bin/bash 110x45
inet6 addr: fe80::3ca5:29ac:fbfe:7ce2/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:36 errors:0 dropped:0 overruns:0 frame:0
TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:6976 (6.9 KB) TX bytes:11371 (11.3 KB)
enp0s8 Link encap:Ethernet HWaddr 08:00:27:05:62:3f
inet addr:192.168.60.1 Bcast:192.168.60.255 Mask:255.255.255.0
inet6 addr: fe80::afda:28f7:6232:8795/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:331 errors:0 dropped:0 overruns:0 frame:0
TX packets:180 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:28911 (28.9 KB) TX bytes:13906 (13.9 KB)
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:335 errors:0 dropped:0 overruns:0 frame:0
TX packets:335 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:59184 (59.1 KB) TX bytes:59184 (59.1 KB)
tun0 Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
inet addr:192.168.53.1 P-t-P:192.168.53.1 Mask:255.255.255.0
inet6 addr: fe80::28fb:5f1d:62aa:3070/64 Scope:Link
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:500
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
[03/12/19]seed@VM:~$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 192.168.60.1 0.0.0.0 UG 100 0 0 enp0s8
0.0.0.0 10.0.2.1 0.0.0.0 UG 101 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 enp0s8
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
[03/12/19]seed@VM:~$
Right Ctrl



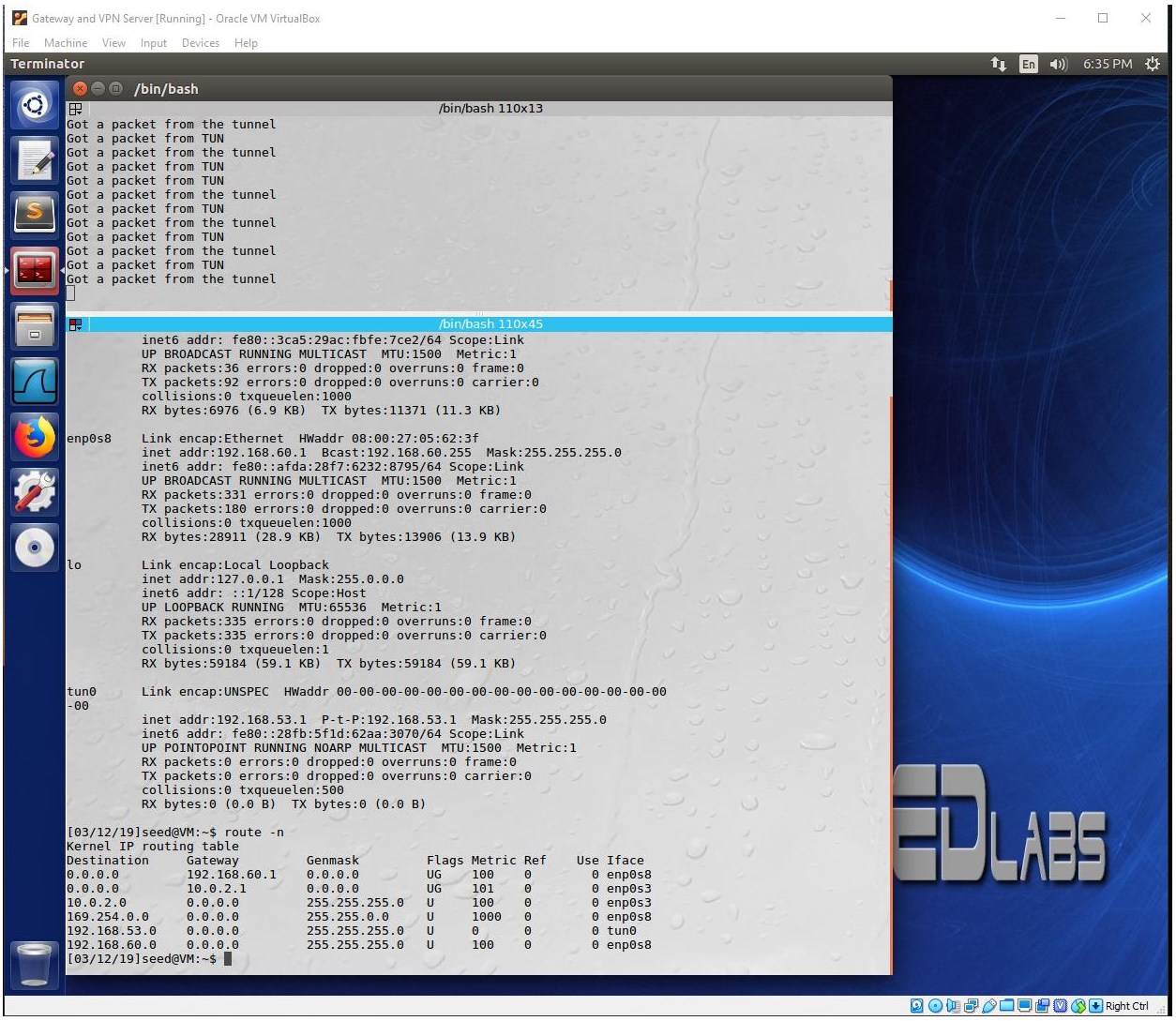
Telnet Test

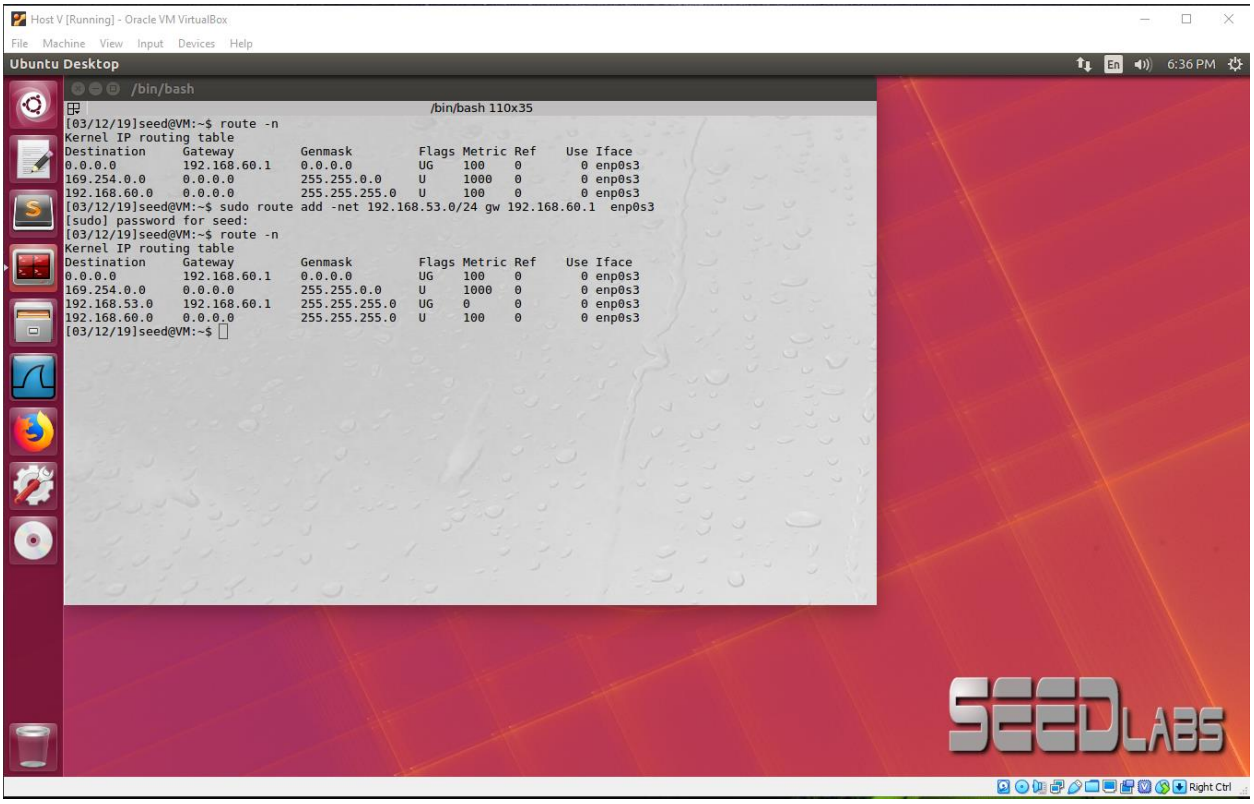
In the figure below, we see the telnet session on the left side. We telnet to 192.168.60.101 and are able to successfully login to the server. On the right side, we see the WireShark trace of the telnet session. We can see a similar pattern that we see with the ping. Highlighting a WireShark section for the telnet session, we see:

1. **LINE 20:** We see the telnet packet going from 192.168.53.5 (Host U's tun0 IP address) to 192.168.60.101, the final destination. **NOT TUNNEL.**
2. **LINE 21:** We then see the packet transmitted from Host U's ethernet adapter 10.0.2.6 to the VPN Server / Gateway's IP address 10.0.2.5. This is the UDP VPN client to VPN server communication. Please note, the original ping packet is a payload of this tunnel packet. **TUNNEL**.
3. **LINE 22:** We then see a packet returned from the VPN Server / Gateways IP address 10.0.2.5 to the Host U ethernet adapter 10.0.2.6. This is the UDP VPN server to VPN client communication. **TUNNEL.**
4. **LINE 23:** Finally, we see that the VPN client releases the received payload into the network Host U's stack. This payload is a from the telnet server and it's sent from 192.168.60.101 to 192.168.53.5, which is right back to the Host U's IP address. **NOT TUNNEL.**



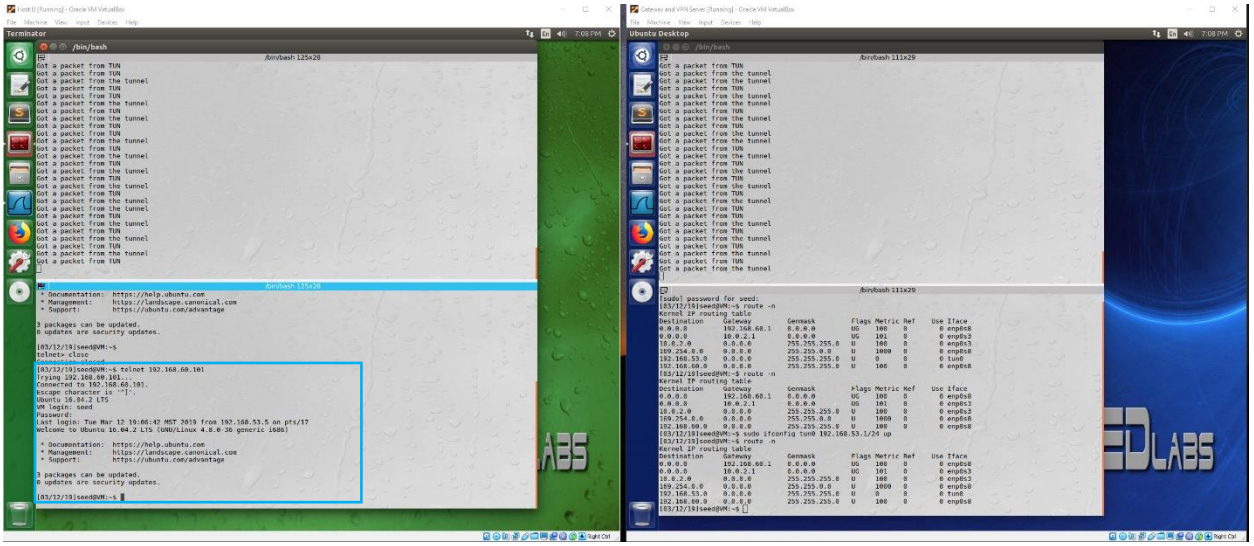
In the next two figures, we see VPN Server / Gateway VM and Host V. In the VPN Server / Gateway VM, we see packets from the tunnel interface and from the VPN tunnel. The Host V merely replies to pings or (later) acts as a telnet server, so there's not much to see there. These are the screenshots regardless.


Gateway and VPN Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
6:35 PM
/bin/bash
/bin/bash 110x13
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 TUN
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
Got a packet from TUN
Got a packet from the tunnel
/bin/bash 110x45
inet6 addr: fe80::3ca5:29ac:fbfe:7ce2/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:36 errors:0 dropped:0 overruns:0 frame:0
TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:6976 (6.9 KB) TX bytes:11371 (11.3 KB)
enp0s8 Link encap:Ethernet HWaddr 08:00:27:05:62:3f
inet addr:192.168.60.1 Bcast:192.168.60.255 Mask:255.255.255.0
inet6 addr: fe80::afda:28f7:6232:8795/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:331 errors:0 dropped:0 overruns:0 frame:0
TX packets:180 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:28911 (28.9 KB) TX bytes:13906 (13.9 KB)
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:335 errors:0 dropped:0 overruns:0 frame:0
TX packets:335 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:59184 (59.1 KB) TX bytes:59184 (59.1 KB)
tun0 Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
inet addr:192.168.53.1 P-t-P:192.168.53.1 Mask:255.255.255.0
inet6 addr: fe80::28fb:5f1d:62aa:3070/64 Scope:Link
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:500
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
[03/12/19]seed@VM:~$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 192.168.60.1 0.0.0.0 UG 100 0 0 enp0s8
0.0.0.0 10.0.2.1 0.0.0.0 UG 101 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 enp0s8
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
[03/12/19]seed@VM:~$
Right Ctrl

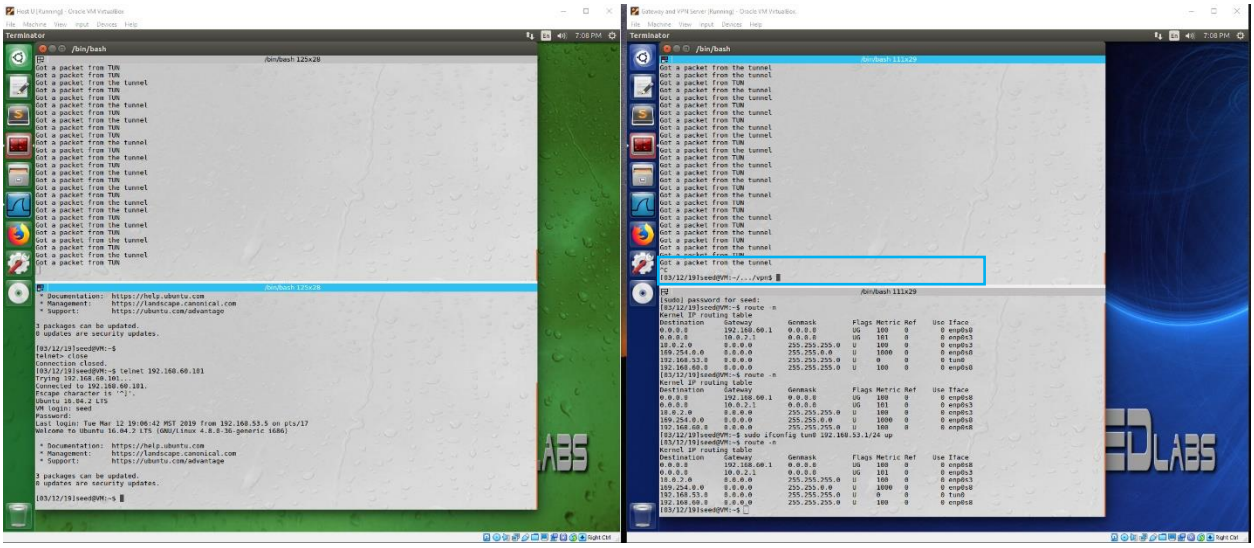


Step 6: Tunnel-Breaking Test

In the figure below, we see a successful telnet session.

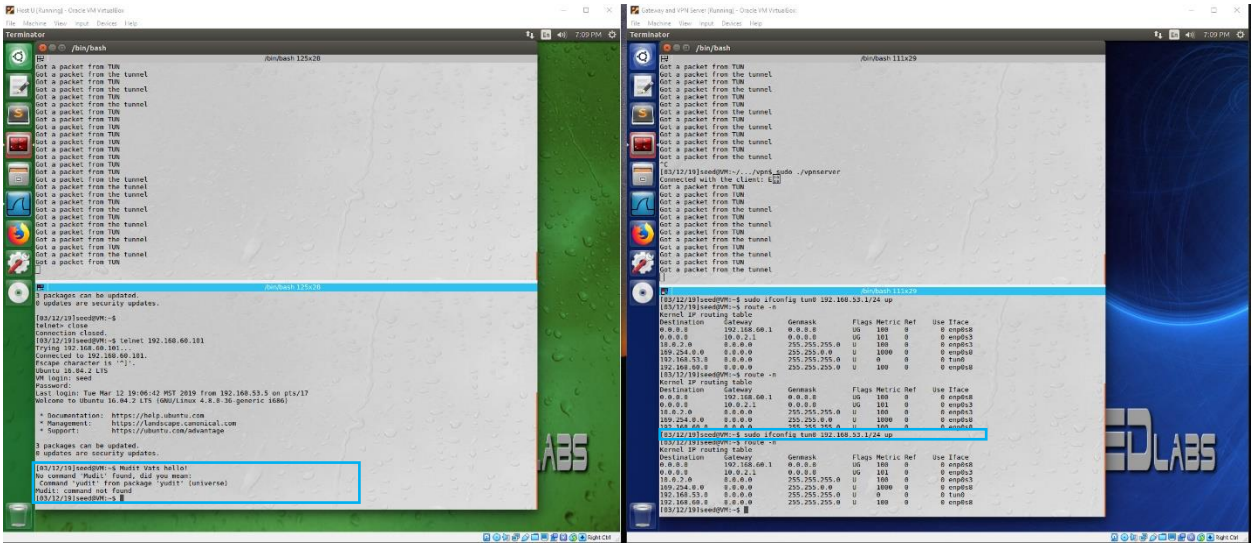


In the figure below, we break the telnet session on the VPN Server / Gateway. In the telnet session on the Host U VM, after the session is broken, we type "Mudit Vats hello!" and the enter key. This cannot be seen, but it will be visible in the WireShark trace.

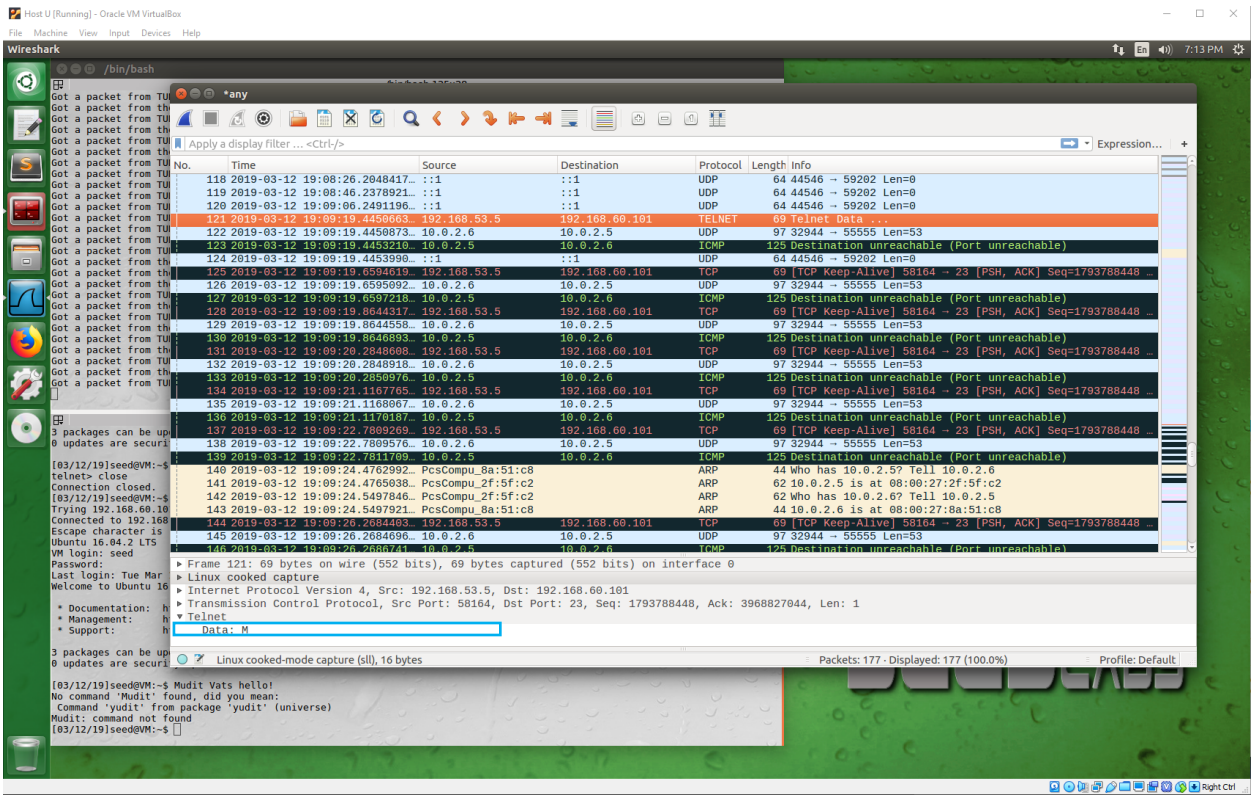


In the figure below, in the bottom half of the terminal session, we see the interface is brought up again. Recall, this assigns an IP to the tun0 adapter and also sets up the route to the VPN tunnel network.

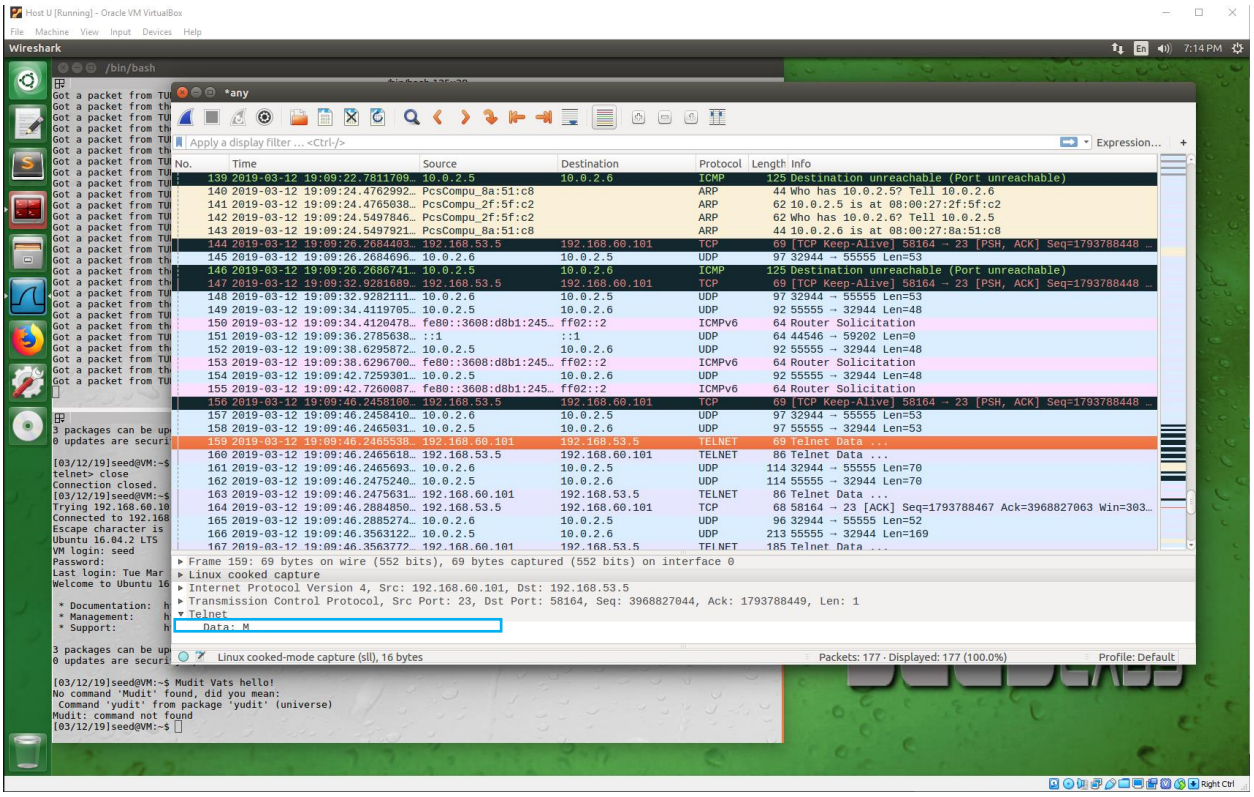
Once the network is established, we see the buffered data in Host U's telnet session.



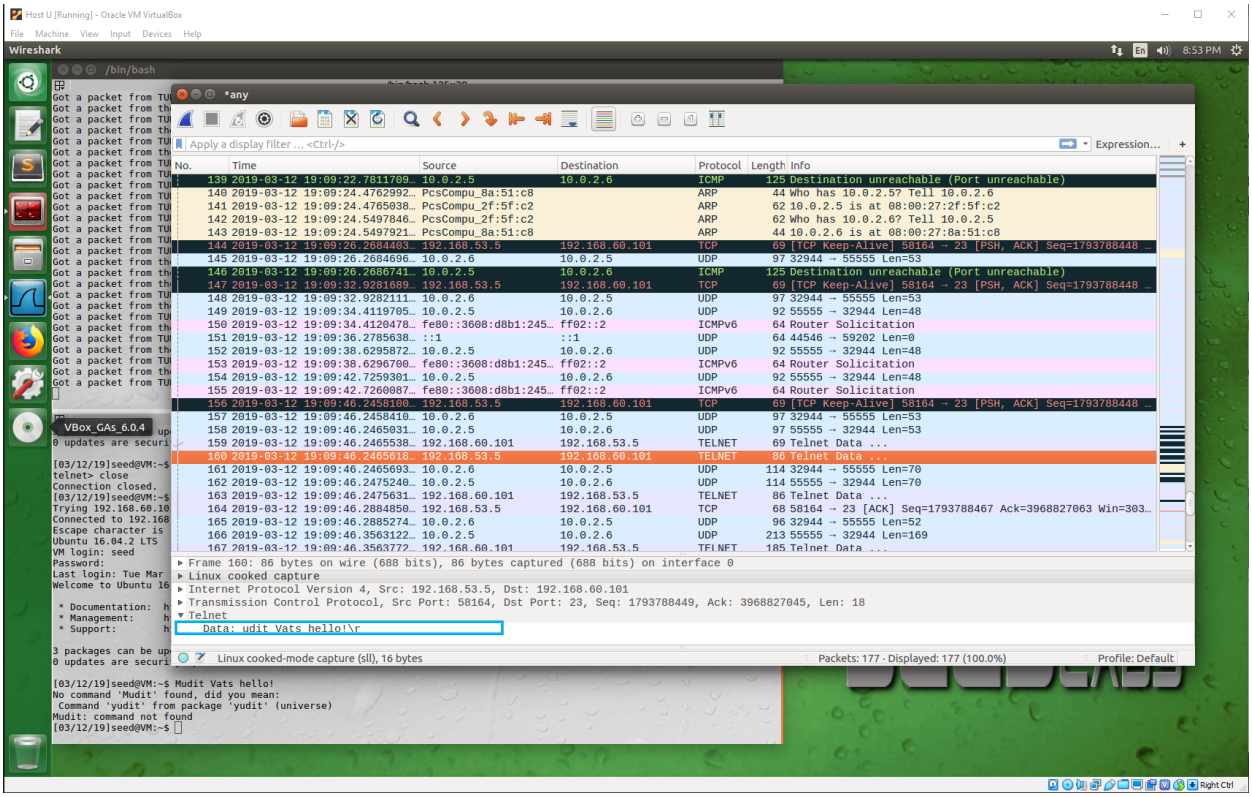
In the figure below, we see the buffered data "M" below.



In the figure below, we see the retransmission of the telnet data “M”.



In the figure below, we see the retransmission of the rest of the data “udit Vats hello!” and the enter key “\r” below.



Regarding the following question in the lab regarding the Tunnel Break test:

`What is going to happen to the telnet connection? Will it be broken or resumed? Please describe and explain your observations.`
When I did the lab, I did not think the packets would be buffered and retransmitted. I thought that the packets would be lost/dropped and the connection unfixable/broken. I was surprised to see this behavior when killing the server and reestablishing the tunnel. While I understand the reliable communication aspect of TCP, I did not consider that (or believe it 😊) upon trying the experiment. Nevertheless, when I went through the exercise and could see the retransmission and, eventually, the text actually appear in the telnet session, I understood better why this was happening and became a believer.

Observations / Explanations

We did quite a bit in this task.

Firstly, we created the tunnel adapters by instantiating the VPN Client and VPN Server. We then manually assigned them their IP address. Finally and, most importantly, we added the routes for Host U, VPN Server / Gateway and Host V. We added routes so Host U can get a packet to the Private Network which VPN Server / Gateway and Host V are on. We also setup Host U and VPN Server / Gateway routes so that packets destined for the tunnel network 192.168.53.0/24 are routed to their respective tun0

interfaces. Finally, we added a route from Host V to VPN Server / Gateway. This was a special route in that we specified that any packets destined for the VPN tunnel network (192.168.53.0/24) should be routed to the Gateway (192.168.60.1). This was the most critical part to get right since without correct routes, packets would not know how to get to their destination or the next hop on the network.

Secondly, we tested our VPN configuration by running Ping and Telnet. Both of these had similar behaviors in that the requests were tunneled to their destination (see explanation of Ping Test above), packets released onto the network and responses received in reverse. Some key learnings:

- Packet always assign the source IP of the interface which it is sent out of.
- If we ping to a destination which is routed to the tunnel adapter, the source IP will be the that of the tunnel adapter's source IP. Seems obvious, but I was confused here as to why the source IP was that of the tunnel adapter and 10.0.2.6, for example in Host U's case.
- Packets which are destined for the tunnel interface, get delivered to the tun0 adapter by means of the network stack. The same holds for when packets are sent via tun0. They go directly to the network stack.
- The "original packet", like the ping or telnet packet, are encapsulated as the payload of the UDP packet which is sent via the VPN tunnel socket interface.
- The payload packets are released from the tunnel adapters into the network stack and then routed per routing rules.
- The UDP socket interface establishes the VPN tunnel between client and server. While the tunnel adapters are a key part of the "VPN", the VPN client and VPN server communicate with each other over sockets which is independent of the tunnel adapters.
- As packets come in via the tunnel adapter (e.g. tun0), the packets become the payload of the new packet which is sent out via the VPN tunnel. As packets come in via the VPN tunnel, the payload is taken out and sent to the tunnel adapter (e.g. tun0) to be release to the network.

Finally, in the Tunnel Break test, we break the tunnel at the server. We type data into the telnet session which gets buffered as part of TCP communication. The data gets continually retransmitted to the server and once we reestablish communication, the data is successfully received in the telnet server and echo'd back to the to the telnet session. Bottom-line, no data (text) was lost in the process, just buffered/retried until the server, tun0 IP address, and routes were restored.

Overall, this was a complex lab and one that was difficult to keep all the parts straight. I got through the lab without much difficulty, but spent much time thinking about the various parts and how they all fit together. I'm glad we did it.