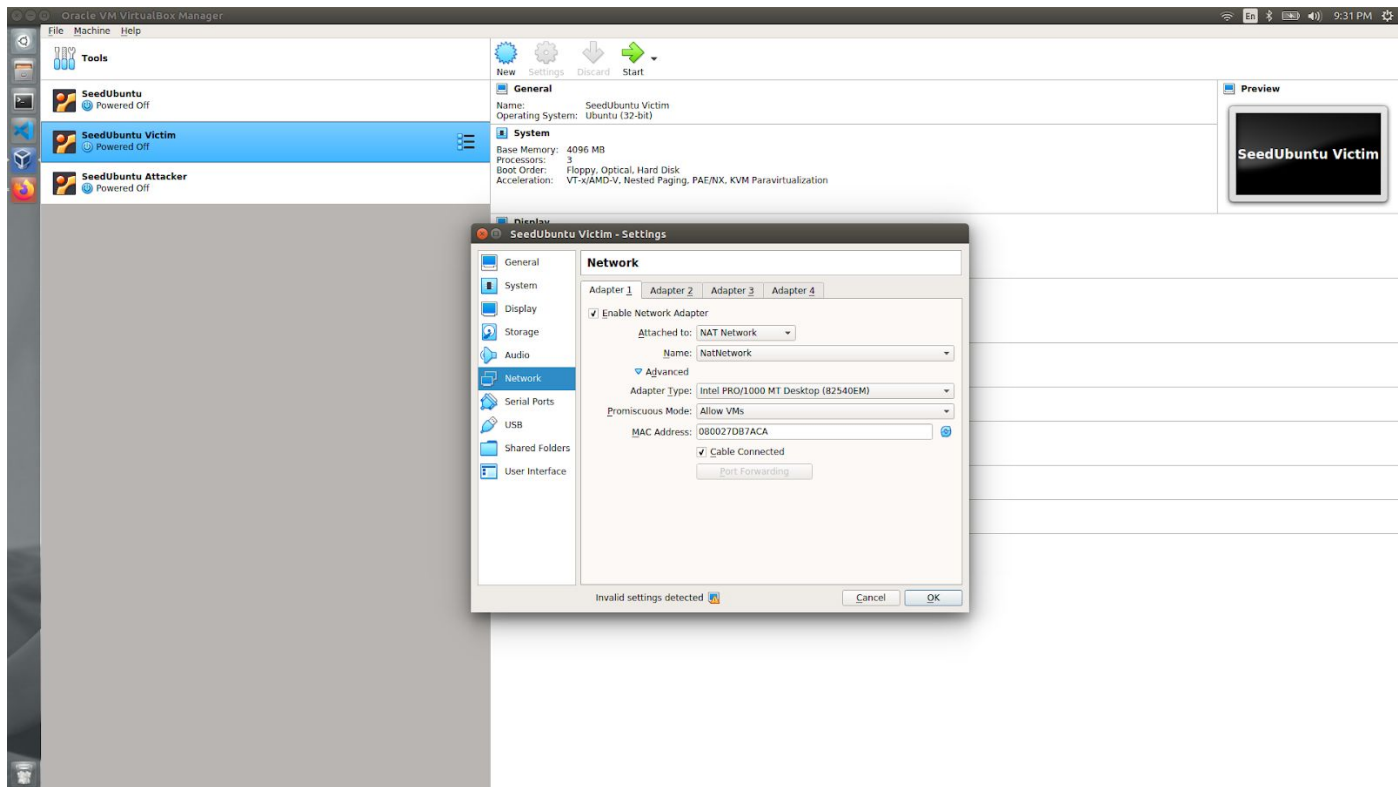


## Task 1 : VM Setup:

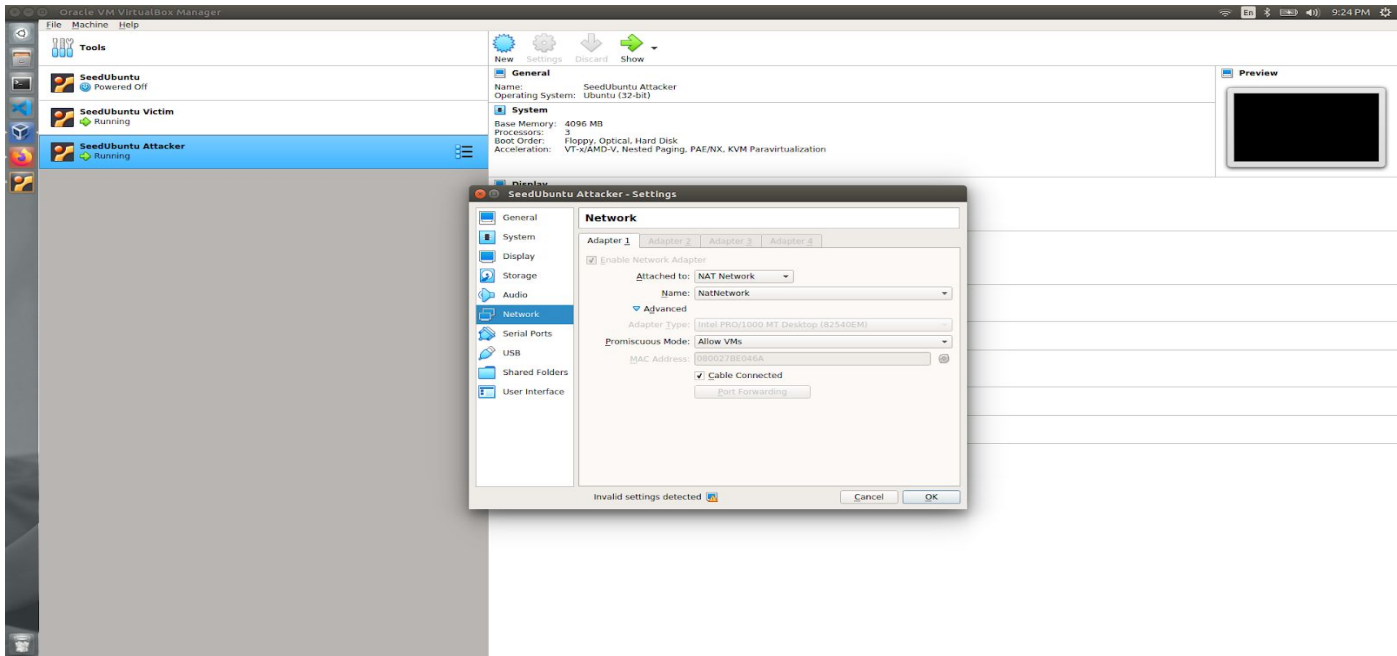
For all the tasks following in this lab, we need a minimum of 2 VM's... VM1 and VM2. These two machines are connected via the internet through the Routers. Here we use a LAN to emulate the internet Connection. Here we connect the two VM's to a LAN using the NAT Network Adapter. The screenshot below shows the setup we've done for these lab tasks.

The configuration of the VM1 which is named **SeedUbuntu Victim** is shown below:



A similar configuration is done to the VM2 which is named as **SeedUbuntu Attacker**. Having NAT adapter set, makes the 2 VM's a part of a LAN.

The configuration of the VM1 which is named **SeedUbuntu Attacker** is shown below:



## Task 2 : Set up Firewall:

In this task, we're setting up the firewall rules on VM1 to block the access of a target website. The IP of the target website, [www.iit.edu](http://www.iit.edu) we'll be using is **50.19.226.237**. And the IP of the VM1 that we'll be using is **10.0.2.4**. We can see that from the screenshot below.

```

Terminal
[05/05/20]seed@VM:~$ ping www.iit.edu
PING www.iit.edu (50.19.226.237) 56(84) bytes of data:
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=1 ttl=49 time=37.4 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=2 ttl=49 time=33.9 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=3 ttl=49 time=33.3 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=4 ttl=49 time=33.6 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=5 ttl=49 time=33.1 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=6 ttl=49 time=34.2 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=7 ttl=49 time=34.7 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=8 ttl=49 time=33.7 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=9 ttl=49 time=34.0 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=10 ttl=49 time=33.7 ms
^C
--- www.iit.edu ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 33.154/34.217/37.443/1.158 ms
[05/05/20]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:db:7a:ca
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::d074:80e0:ceb1:9220/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20906 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11656 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:23849794 (23.8 MB)  TX bytes:2077133 (2.0 MB)

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:2167 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2167 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:217098 (217.0 KB)  TX bytes:217098 (217.0 KB)

[05/05/20]seed@VM:~$
    
```

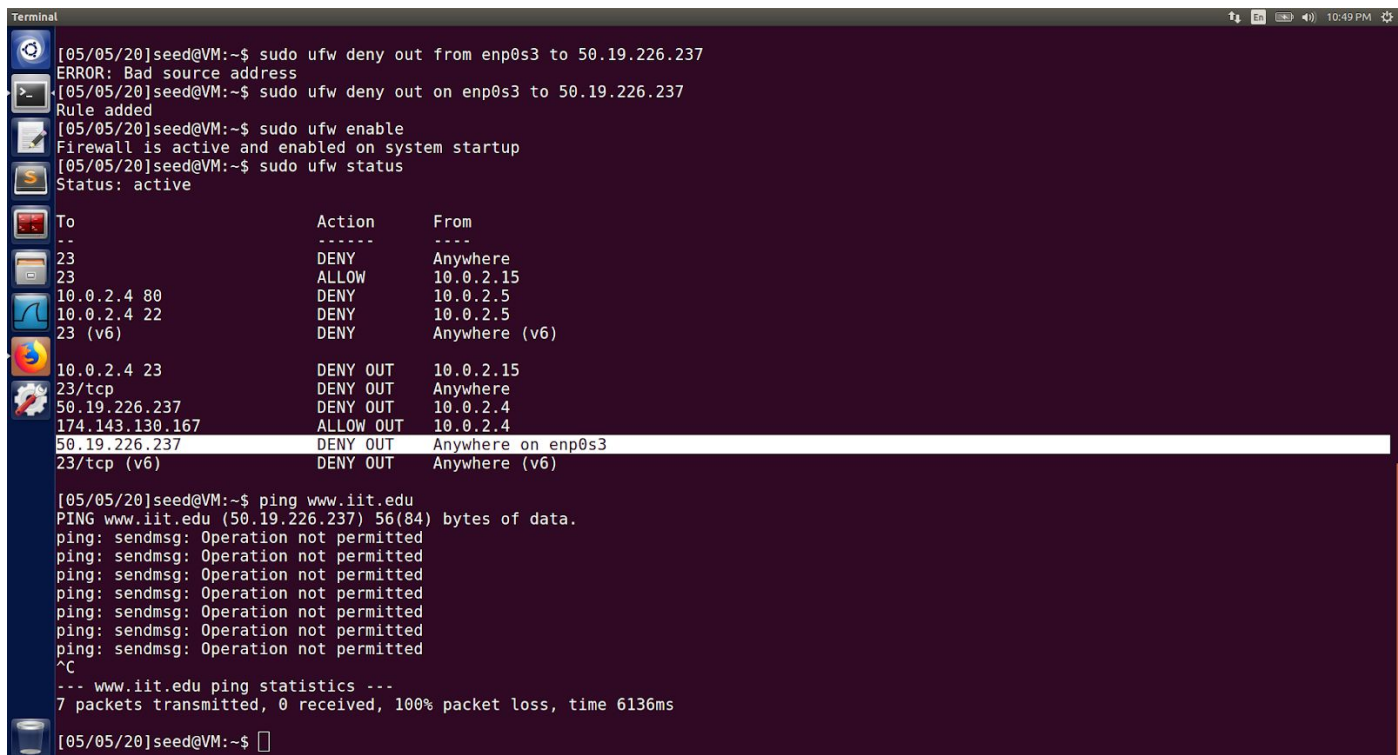
In this task, taking the IP of [www.iit.edu](http://www.iit.edu) we got from the ping command above as **50.19.226.237**, we update the firewall rule by executing the ufw command as shown below.

### **sudo ufw deny out from 10.0.2.4 to 50.19.226.237**

Once the above command is executed, the above firewall rule is updated. But this updated rule has to be activated. To activate the updated firewall rules, we execute the following command.

### **sudo ufw enable**

Once the firewall rules are activated, I now execute the ping command with [www.iit.edu](http://www.iit.edu) as an argument to the ping command. Since the firewall is updated, we can see the ping to [www.iit.edu](http://www.iit.edu) fails as an access to [www.iit.edu](http://www.iit.edu) is blocked from VM1. We can see all this happening from the screenshot below.



```
[05/05/20]seed@VM:~$ sudo ufw deny out from enp0s3 to 50.19.226.237
ERROR: Bad source address
[05/05/20]seed@VM:~$ sudo ufw deny out on enp0s3 to 50.19.226.237
Rule added
[05/05/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[05/05/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
23 DENY Anywhere
23 ALLOW 10.0.2.15
10.0.2.4 80 DENY 10.0.2.5
10.0.2.4 22 DENY 10.0.2.5
23 (v6) DENY Anywhere (v6)
10.0.2.4 23 DENY OUT 10.0.2.15
23/tcp DENY OUT Anywhere
50.19.226.237 DENY OUT 10.0.2.4
174.143.130.167 ALLOW OUT 10.0.2.4
50.19.226.237 DENY OUT Anywhere on enp0s3
23/tcp (v6) DENY OUT Anywhere (v6)

[05/05/20]seed@VM:~$ ping www.iit.edu
PING www.iit.edu (50.19.226.237) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- www.iit.edu ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6136ms

[05/05/20]seed@VM:~$
```

From the highlighted section from the screenshot we can see that access to **50.19.226.237** is denied from anywhere on the **enp0s3** interface. And the ping to [www.iit.edu](http://www.iit.edu) fails as in the output we can see the operation not being permitted.

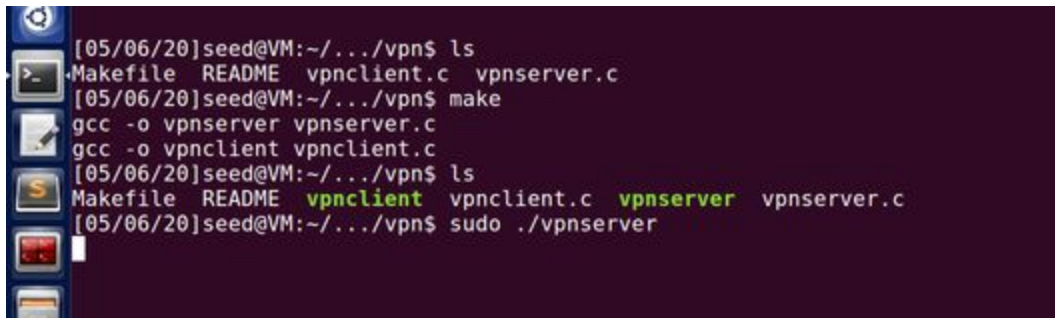
### Task 3: Bypassing Firewall using VPN

For this task, we make use of a VPN tunnel by executing the client and server programs downloaded from the lab website.

#### Step 1: Run VPN Server.

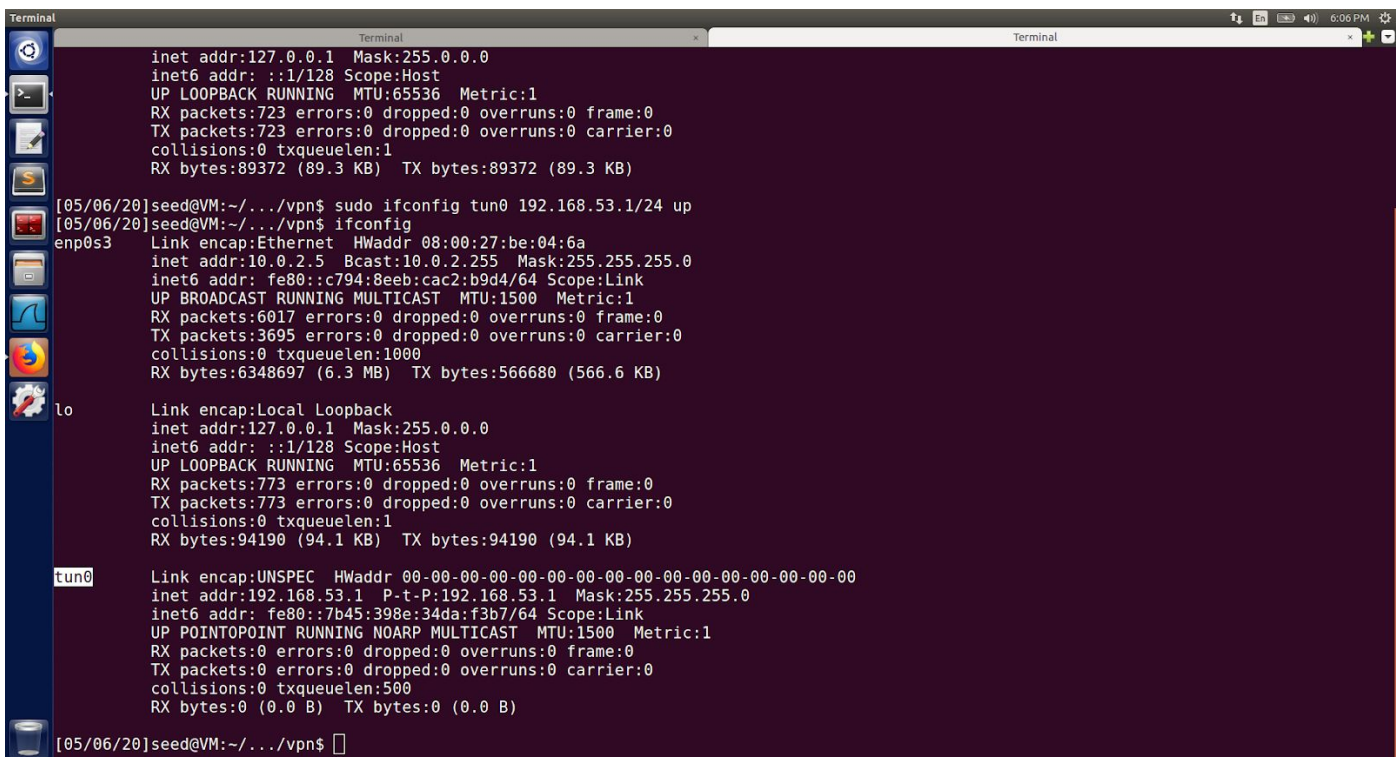
Here for this task, I chose VM2 as a server and ran vpnserver program. After we execute the make command, it generates the 2 executables vpnclient and vpnserver.

On the VM2 I ran vpnserver executable with sudo



```
[05/06/20]seed@VM:~/.../vpn$ ls
Makefile  README  vpnclient.c  vpnserver.c
[05/06/20]seed@VM:~/.../vpn$ make
gcc -o vpnserver vpnserver.c
gcc -o vpnclient vpnclient.c
[05/06/20]seed@VM:~/.../vpn$ ls
Makefile  README  vpnclient  vpnclient.c  vpnserver  vpnserver.c
[05/06/20]seed@VM:~/.../vpn$ sudo ./vpnserver
```

Now, in another tab, I configure the new interface tun0 and gave it an IP 192.168.53.1. **sudo ifconfig tun0 192.168.53.1/24 up** is executed and here is how it looks.



```
Terminal
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:723 errors:0 dropped:0 overruns:0 frame:0
TX packets:723 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:89372 (89.3 KB) TX bytes:89372 (89.3 KB)

[05/06/20]seed@VM:~/.../vpn$ sudo ifconfig tun0 192.168.53.1/24 up
[05/06/20]seed@VM:~/.../vpn$ ifconfig
enp0s3
Link encap:Ethernet HWaddr 08:00:27:be:04:6a
inet addr:10.0.2.5 Bcast:10.0.2.255 Mask:255.255.255.0
inet6 addr: fe80::c794:8eeb:cac2:b9d4/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:6017 errors:0 dropped:0 overruns:0 frame:0
TX packets:3695 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:6348697 (6.3 MB) TX bytes:566680 (566.6 KB)

tun0
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:773 errors:0 dropped:0 overruns:0 frame:0
TX packets:773 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:94190 (94.1 KB) TX bytes:94190 (94.1 KB)

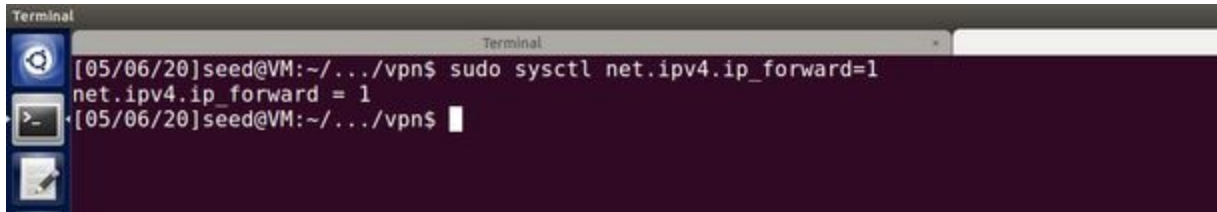
tun0
Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
inet addr:192.168.53.1 P-t-P:192.168.53.1 Mask:255.255.255.0
inet6 addr: fe80::7b45:398e:34da:f3b7/64 Scope:Link
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX 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)

[05/06/20]seed@VM:~/.../vpn$
```



Now that the interface is configured, unless specified this computer will only act as a host. We need to configure it in such a way that it has to act as a gateway so that it can forward packets to other destinations. We need to enable IP forwarding to make it behave like a gateway. We run this command in order to make it a gateway:

**sudo sysctl net.ipv4.ip\_forward=1**

A terminal window with a dark purple background. The prompt is [05/06/20]seed@VM:~/.../vpn\$. The command 'sudo sysctl net.ipv4.ip\_forward=1' has been entered and executed. The output shows 'net.ipv4.ip\_forward = 1'.

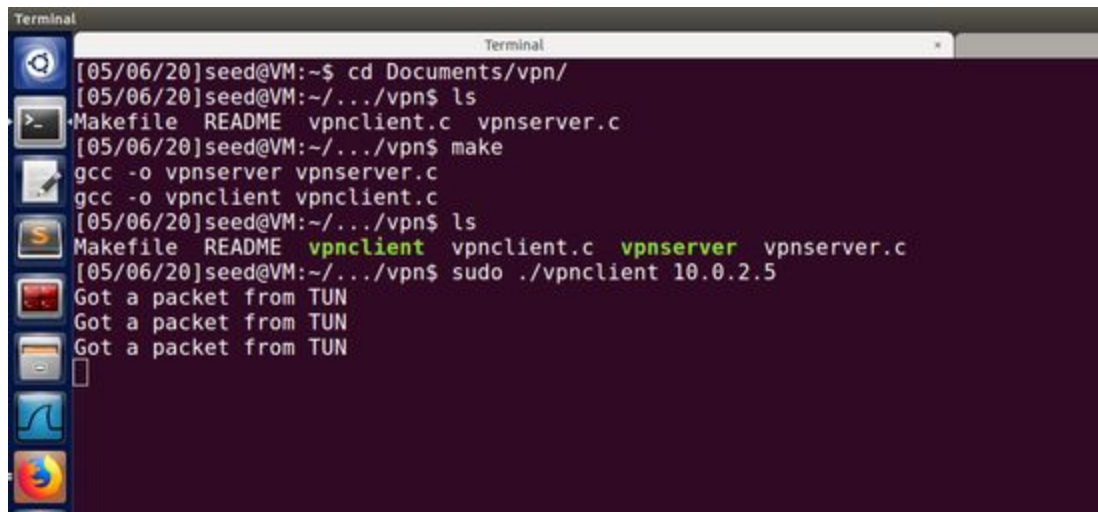
```
Terminal
[05/06/20]seed@VM:~/.../vpn$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[05/06/20]seed@VM:~/.../vpn$
```

### Step 2: Run VPN client.

For this, I chose VM1 as a client and ran a VPN client program on the VM2. We execute the following command

**sudo ./vpncclient 10.0.2.5**

Here 10.0.2.5 is the ip of the machine on which the Server program is running.

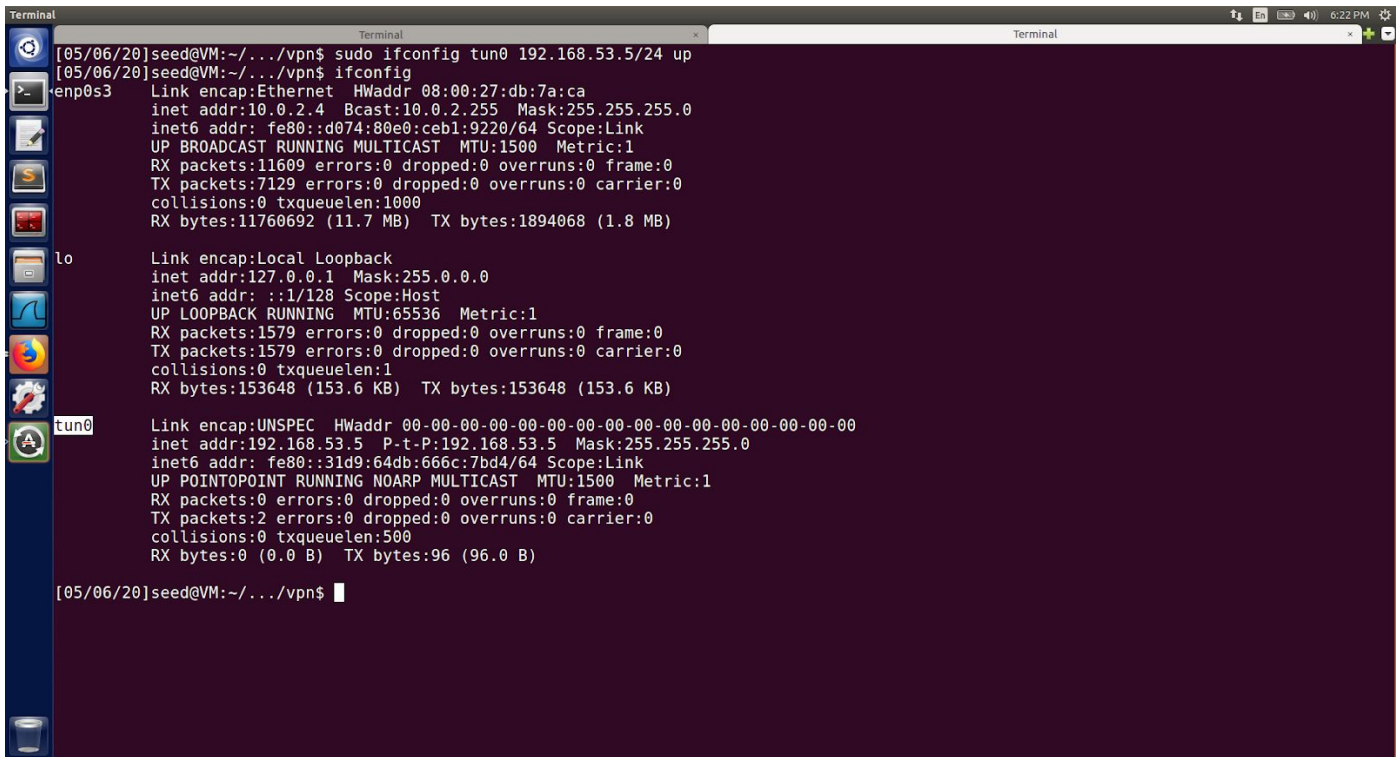
A terminal window with a dark purple background. The prompt is [05/06/20]seed@VM:~\$. The user navigates to ~/Documents/vpn/ and lists files. Then they run 'make' to compile 'vpnsrv.c' and 'vpncclient.c'. After listing files again, they run 'sudo ./vpncclient 10.0.2.5'. The output shows 'Got a packet from TUN' three times.

```
Terminal
[05/06/20]seed@VM:~$ cd Documents/vpn/
[05/06/20]seed@VM:~/.../vpn$ ls
Makefile  README  vpncclient.c  vpnsrv.c
[05/06/20]seed@VM:~/.../vpn$ make
gcc -o vpnsrv vpnsrv.c
gcc -o vpncclient vpncclient.c
[05/06/20]seed@VM:~/.../vpn$ ls
Makefile  README  vpncclient  vpncclient.c  vpnsrv  vpnsrv.c
[05/06/20]seed@VM:~/.../vpn$ sudo ./vpncclient 10.0.2.5
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
```

Now in another tab, I configured the IP to the tun0 interface by executing the following command:

**sudo ifconfig tun0 192.168.53.5/24 up**

Once we've configured the IP to this interface, here is how it looks.



```

[05/06/20]seed@VM:~/.../vpn$ sudo ifconfig tun0 192.168.53.5/24 up
[05/06/20]seed@VM:~/.../vpn$ ifconfig
enp0s3
    Link encap:Ethernet  HWaddr 08:00:27:db:7a:ca
    inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
    inet6 addr: fe80::d074:80e0:ceb1:9220/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
    RX packets:11609 errors:0 dropped:0 overruns:0 frame:0
    TX packets:7129 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:11760692 (11.7 MB)  TX bytes:1894068 (1.8 MB)

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:1579 errors:0 dropped:0 overruns:0 frame:0
    TX packets:1579 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1
    RX bytes:153648 (153.6 KB)  TX bytes:153648 (153.6 KB)

tun0
    Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
    inet addr:192.168.53.5  P-t-P:192.168.53.5  Mask:255.255.255.0
    inet6 addr: fe80::31d9:64db:666c:7bd4/64 Scope:Link
    UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
    RX packets:0 errors:0 dropped:0 overruns:0 frame:0
    TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:500
    RX bytes:0 (0.0 B)  TX bytes:96 (96.0 B)

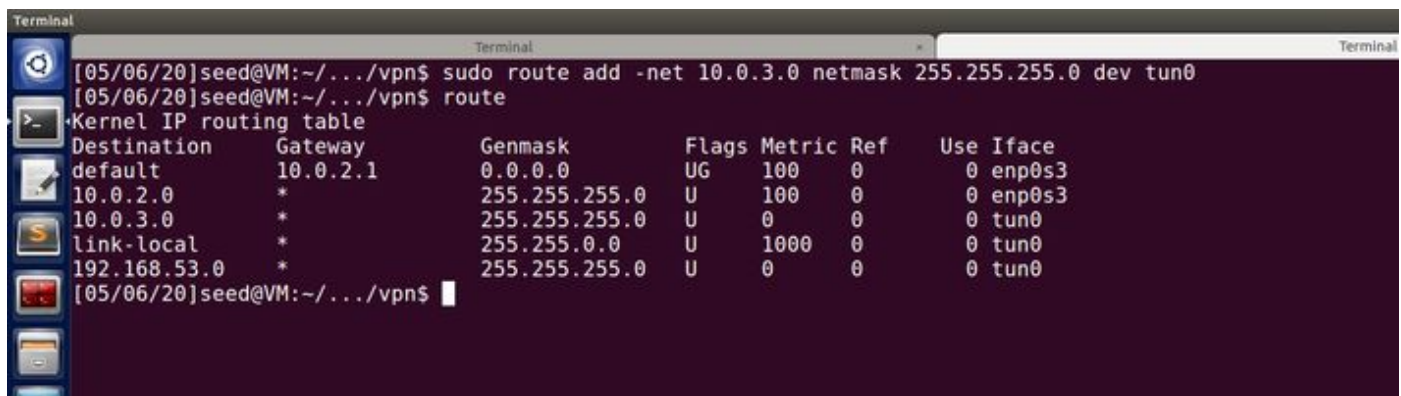
[05/06/20]seed@VM:~/.../vpn$
  
```

### **Step 3 : Setup Routing on Client and Server VMs**

On completing the above 2 steps, a tunnel is not established between the Client and the Server. We now need to set up the routing paths between the client and the server to direct the intended traffic through the tunnel. We make use of the Route command to add a routing entry. **sudo route add -net 10.0.3.0 netmask 255.255.255.0 dev tun0**

The same command is run on Server side as well. We can see from the screenshots below.

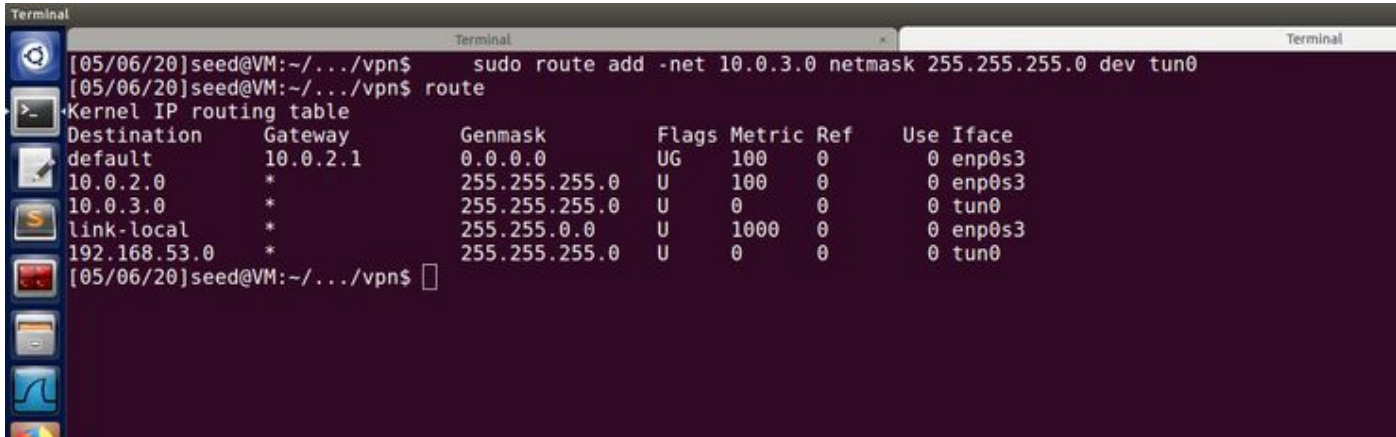
#### **On Client Side:**



```

[05/06/20]seed@VM:~/.../vpn$ sudo route add -net 10.0.3.0 netmask 255.255.255.0 dev tun0
[05/06/20]seed@VM:~/.../vpn$ route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
default        10.0.2.1        0.0.0.0         UG    100    0      0 enp0s3
10.0.2.0       *               255.255.255.0   U    100    0      0 enp0s3
10.0.3.0       *               255.255.255.0   U     0     0      0 tun0
link-local     *               255.255.0.0     U    1000    0      0 tun0
192.168.53.0   *               255.255.255.0   U     0     0      0 tun0

[05/06/20]seed@VM:~/.../vpn$
  
```

**On Server Side:**


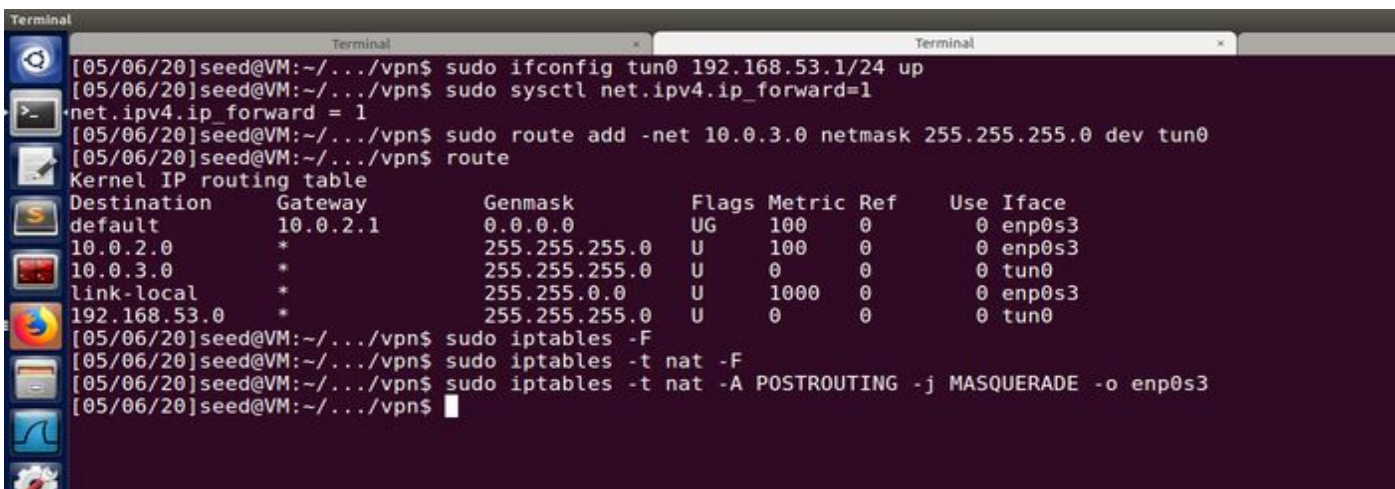
```

[05/06/20]seed@VM:~/.../vpn$ sudo route add -net 10.0.3.0 netmask 255.255.255.0 dev tun0
[05/06/20]seed@VM:~/.../vpn$ route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
default        10.0.2.1        0.0.0.0         UG    100    0      0 enp0s3
10.0.2.0       *               255.255.255.0   U     100    0      0 enp0s3
10.0.3.0       *               255.255.255.0   U      0      0      0 tun0
link-local     *               255.255.0.0     U    1000    0      0 enp0s3
192.168.53.0   *               255.255.255.0   U      0      0      0 tun0
[05/06/20]seed@VM:~/.../vpn$

```

**Step 4: Set Up NAT on Server VM.**

When the final destination sends packets back to users, the packet will be sent to the VPN Server first. **This is because the, since the firewall on the client that makes the calls to the [www.iit.edu](http://www.iit.edu) is configured in such a way that it blocks, all the connections from the Client are redirected to the Server through the tunnel and the call to [www.iit.edu](http://www.iit.edu) will be made from the server on behalf of the Client. And conventionally, the response from the destination hits the source which in our case is Server program on behalf of the Client.** We'll now be executing the set of commands shown in the below screenshot will setup IP forwarding and then clears the iptables rules and then adds a rule on postrouting position to the natnetwork adapter connected to the VPN server.



```

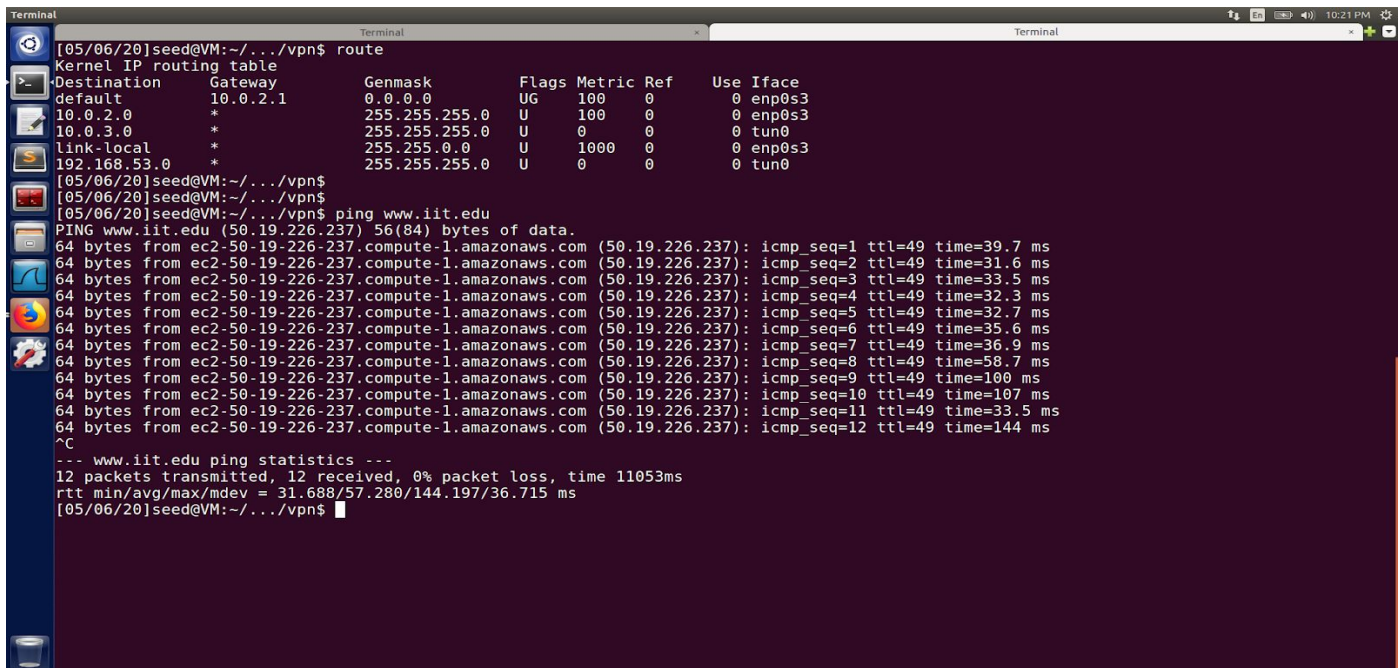
[05/06/20]seed@VM:~/.../vpn$ sudo ifconfig tun0 192.168.53.1/24 up
[05/06/20]seed@VM:~/.../vpn$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[05/06/20]seed@VM:~/.../vpn$ sudo route add -net 10.0.3.0 netmask 255.255.255.0 dev tun0
[05/06/20]seed@VM:~/.../vpn$ route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
default        10.0.2.1        0.0.0.0         UG    100    0      0 enp0s3
10.0.2.0       *               255.255.255.0   U     100    0      0 enp0s3
10.0.3.0       *               255.255.255.0   U      0      0      0 tun0
link-local     *               255.255.0.0     U    1000    0      0 enp0s3
192.168.53.0   *               255.255.255.0   U      0      0      0 tun0
[05/06/20]seed@VM:~/.../vpn$ sudo iptables -F
[05/06/20]seed@VM:~/.../vpn$ sudo iptables -t nat -F
[05/06/20]seed@VM:~/.../vpn$ sudo iptables -t nat -A POSTROUTING -j MASQUERADE -o enp0s3
[05/06/20]seed@VM:~/.../vpn$

```

In our case the interface that we're using is enp0s3.

Doing this the packets received at the server end is not meant to be at the server. They must be forwarded. This is why we set up IP forwarding. We then explore the limitation of NAT by adding a rule in the POSTROUTING position to the NAT network adapter connected to the VPN server.

Now that the tunnel is set, configured and the route is also configured, we now ping [www.iit.edu](http://www.iit.edu) from the client machine. We can see from the screenshot below how the ping is successful.



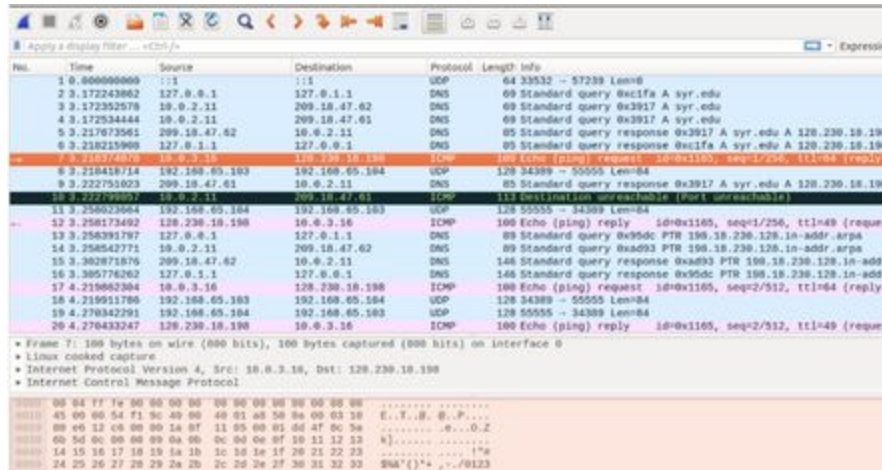
```
[05/06/20]seed@VM:~/.../vpn$ route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 10.0.2.1 0.0.0.0 UG 100 0 0 enp0s3
10.0.2.0 * 255.255.255.0 U 100 0 0 enp0s3
10.0.3.0 * 255.255.255.0 U 0 0 0 tun0
link-local * 255.255.0.0 U 1000 0 0 enp0s3
192.168.53.0 * 255.255.255.0 U 0 0 0 tun0

[05/06/20]seed@VM:~/.../vpn$
[05/06/20]seed@VM:~/.../vpn$
[05/06/20]seed@VM:~/.../vpn$ ping www.iit.edu
PING www.iit.edu (50.19.226.237) 56(84) bytes of data:
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=1 ttl=49 time=39.7 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=2 ttl=49 time=31.6 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=3 ttl=49 time=33.5 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=4 ttl=49 time=32.3 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=5 ttl=49 time=32.7 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=6 ttl=49 time=35.6 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=7 ttl=49 time=36.9 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=8 ttl=49 time=58.7 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=9 ttl=49 time=100 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=10 ttl=49 time=107 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=11 ttl=49 time=33.5 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=12 ttl=49 time=144 ms
^C
--- www.iit.edu ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11053ms
rtt min/avg/max/mdev = 31.688/57.280/144.197/36.715 ms
[05/06/20]seed@VM:~/.../vpn$
```

When we ping [www.iit.edu](http://www.iit.edu) the ping to the IP **50.19.226.237** is now successful and we get the response to the Server's IP. The response is sent to the client through the tunnel. The wireshark capture when the ping is executed is shown below.



The ping's ICMP packets to the IP **50.19.226.237** will go through the tun0 interface to the client side of the tunnel. The packets are sent through the tunnel to the server side. The packets are then sent to the destination from the VPN server. Here the packets are actually originated from the Client but then sent by the server on behalf of the server.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	1:1	1:1	UDP	64	33532 → 57289 Len=6
2	3.17243862	127.0.0.1	127.0.1.1	DNS	69	Standard query 0xc1fa A syr.edu
3	3.17252578	10.0.2.11	209.18.47.62	DNS	69	Standard query 0xc3917 A syr.edu
4	3.17253444	10.0.2.11	209.18.47.61	DNS	69	Standard query 0xc3917 A syr.edu
5	3.217673561	209.18.47.62	10.0.2.11	DNS	85	Standard query response 0xc3917 A syr.edu A 128.230.18.198
6	3.218215906	127.0.1.1	127.0.0.1	DNS	85	Standard query response 0xc1fa A syr.edu A 128.230.18.198
7	8.701276734	10.0.3.16	10.0.3.16	ICMP	100	Echo (ping) request id=0x1165, seq=2/512, ttl=64 (reply ...)
8	8.218418714	192.168.65.184	192.168.65.184	UDP	128	34389 → 55555 Len=64
9	8.222751603	209.18.47.61	10.0.2.11	DNS	85	Standard query response 0xc3917 A syr.edu A 128.230.18.198
10	8.222798997	10.0.2.11	209.18.47.61	ICMP	112	Destination unreachable (Port unreachable)
11	3.258023664	192.168.65.184	192.168.65.184	UDP	128	55555 → 34389 Len=64
12	3.258173492	128.230.18.198	10.0.3.16	ICMP	100	Echo (ping) reply id=0x1165, seq=1/256, ttl=48 (request ...)
13	3.258391797	127.0.0.1	127.0.1.1	DNS	89	Standard query 0x95dc PTR 198.18.230.128.in-addr.arpa
14	3.258542771	10.0.2.11	209.18.47.62	DNS	89	Standard query 0xad93 PTR 198.18.230.128.in-addr.arpa
15	3.362871876	209.18.47.62	10.0.2.11	DNS	146	Standard query response 0xad93 PTR 198.18.230.128.in-addr...
16	3.365776262	127.0.1.1	127.0.0.1	DNS	146	Standard query response 0x95dc PTR 198.18.230.128.in-addr...
17	4.218902384	10.0.3.16	128.230.18.198	ICMP	100	Echo (ping) request id=0x1165, seq=2/512, ttl=64 (reply ...)
18	4.219951796	192.168.65.184	192.168.65.184	UDP	128	34389 → 55555 Len=64
19	4.270342291	192.168.65.184	192.168.65.184	UDP	128	55555 → 34389 Len=64
20	4.270433247	128.230.18.198	10.0.3.16	ICMP	100	Echo (ping) reply id=0x1165, seq=2/512, ttl=48 (request ...)

\* Frame 7: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0  
 \* Linux cooked capture  
 \* Internet Protocol Version 4, Src: 10.0.3.16, Dst: 128.230.18.198  
 \* Internet Control Message Protocol

```

0000  08 64 ff 7e 00 00 00 00 00 00 00 00 00 00 00 00  .....E..T..R..P...
0010  45 00 00 54 f1 0c 40 00 40 01 40 50 8a 00 03 18  ...V...R...O.Z...
0020  00 e0 12 c6 00 00 1a 0f 11 05 00 01 6d 47 0c 3a  .....R.....!*#
0030  00 50 0c 00 00 09 0a 00 0c 00 0e 0f 10 11 12 13  8}.....
0040  14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23  .....
0050  24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33  9NA*{}*4.../0123
  
```