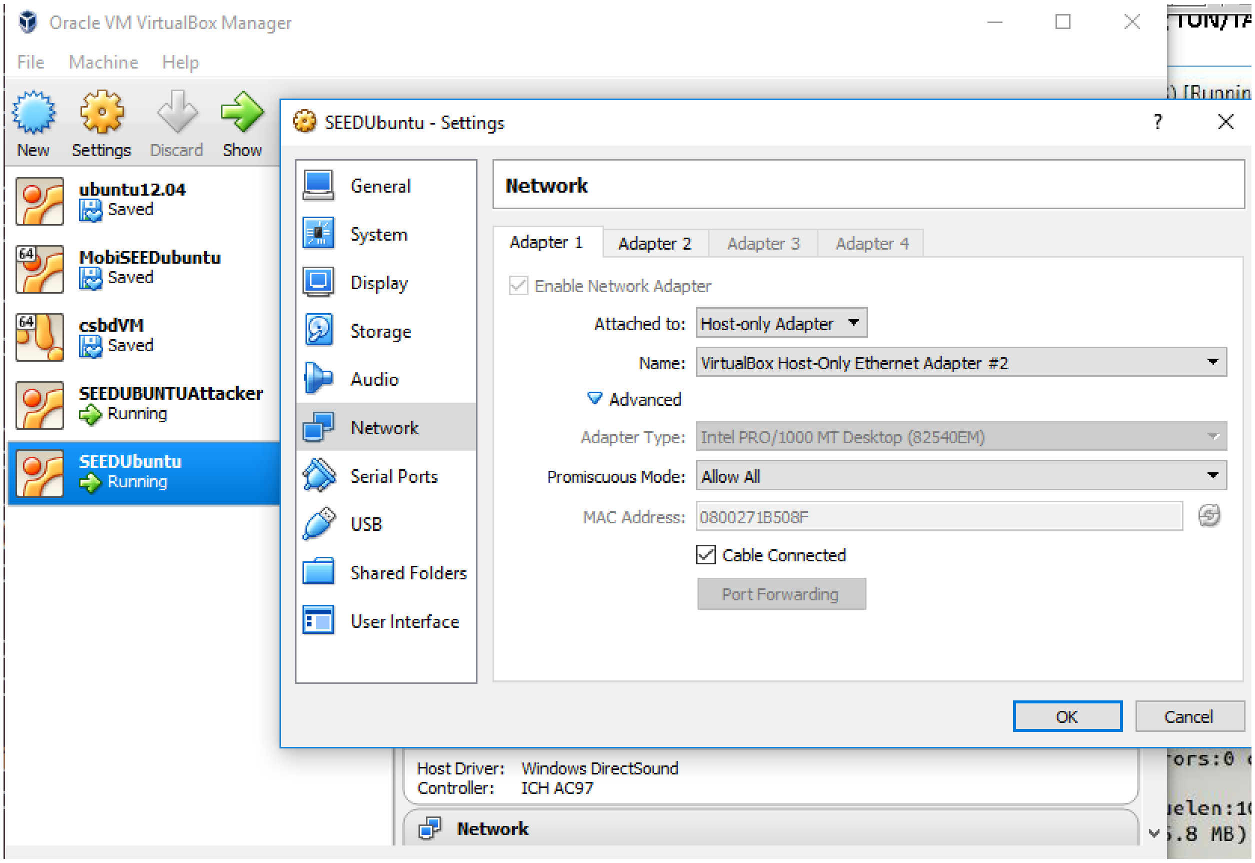
Internet Security CSE644
Lab 3: Bypassing Firewall using VPN

Aastha Yadav (ayadav02@syr.edu)
SUID: 831570679

Task 1: Create a Host-to-Host Tunnel using TUN/TAP



ubuntu12.04 Saved
MobiSEEDubuntu Saved
csbdVM Saved
SEEDUBUNTUAttacker Running
SEEDUbuntu Running

Network

☑ Enable Network Adapter

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter #2

Advanced

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

Promiscuous Mode: Allow All

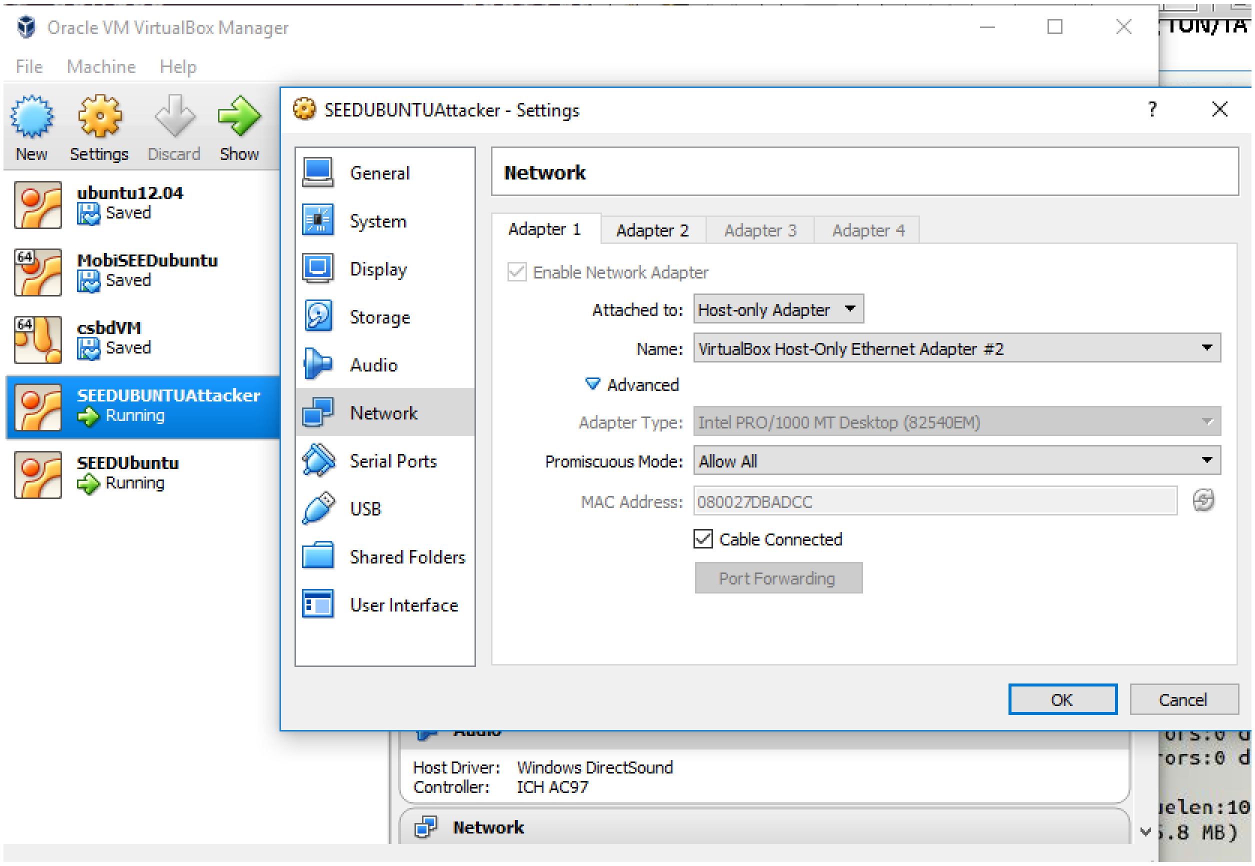
MAC Address: 0800271B508F

☑ Cable Connected

Host Driver: Windows DirectSound
Controller: ICH AC97

Network

Figure 1



Network

☑ Enable Network Adapter

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter #2

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

Promiscuous Mode: Allow All

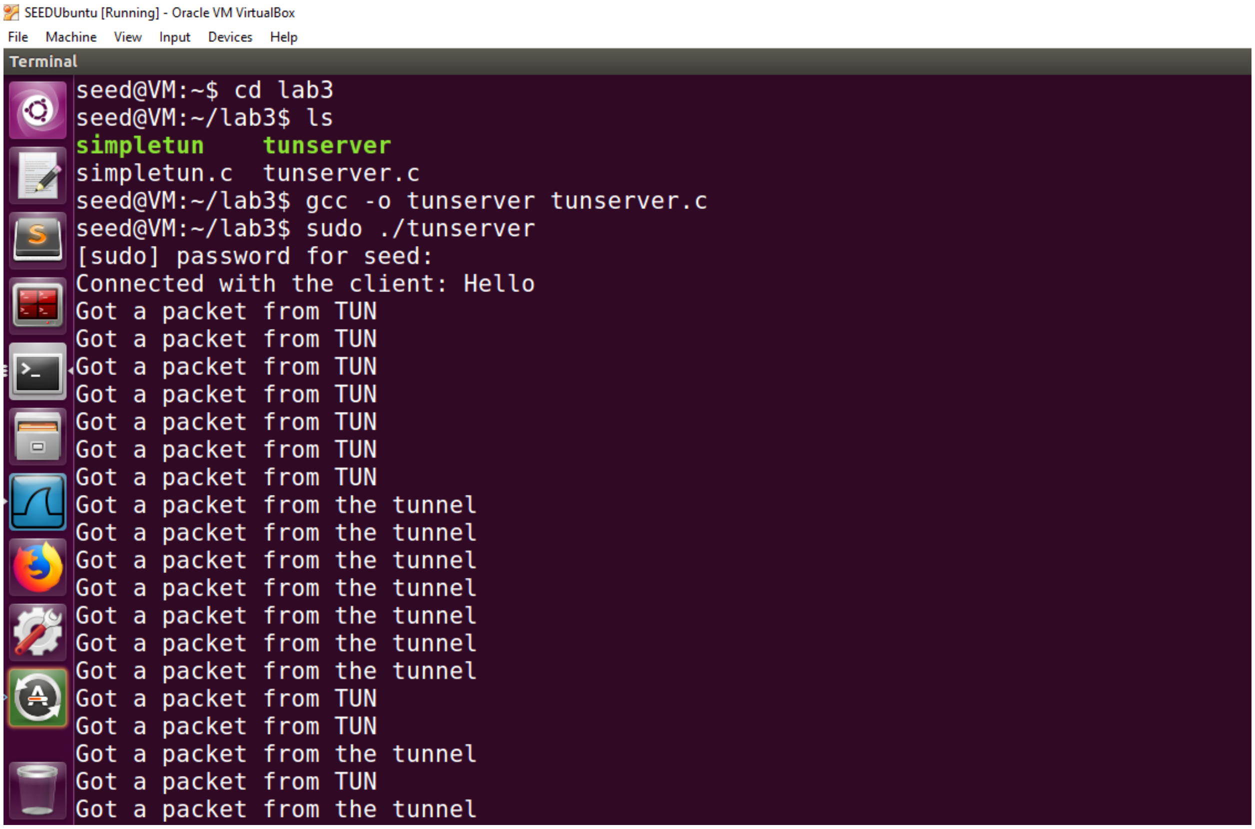
MAC Address: 080027DBADCC

☑ Cable Connected

Host Driver: Windows DirectSound
Controller: ICH AC97

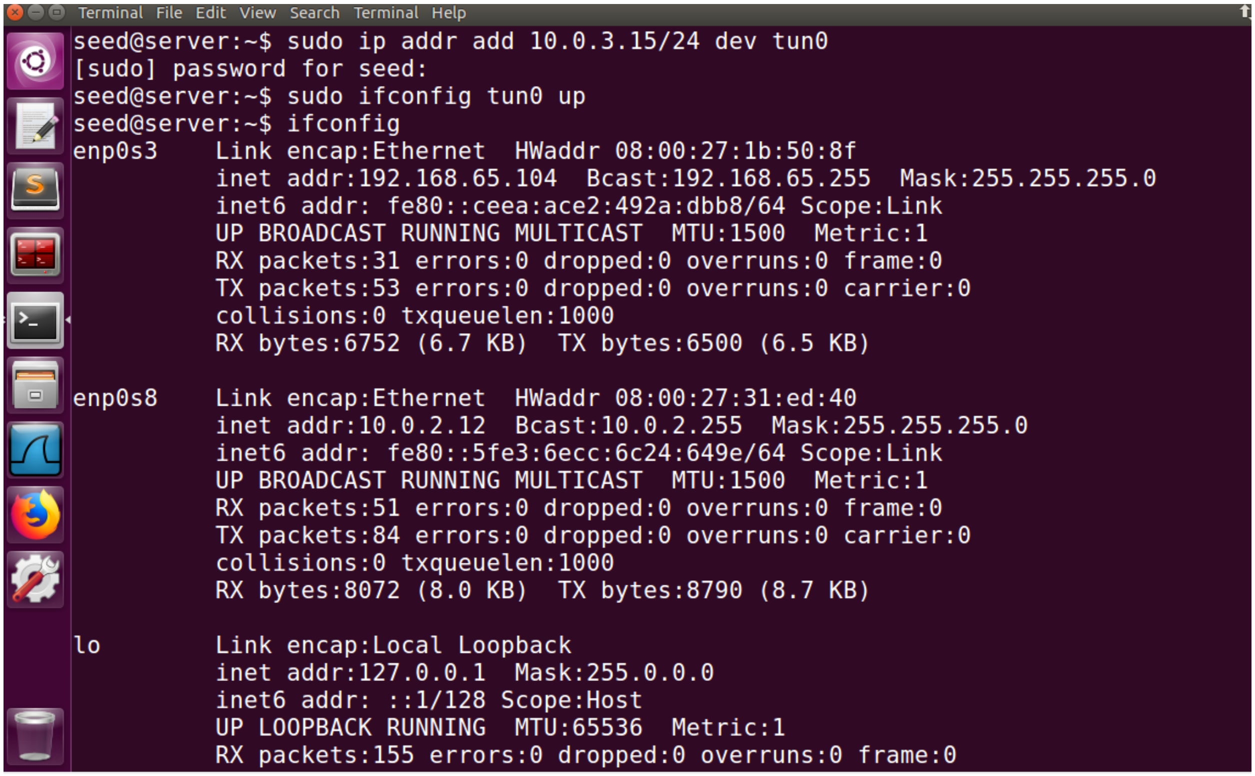
Figure 2

Observation: We enable a host-only network adapter in our VMs so that we can emulate internet that connects both the VMs.

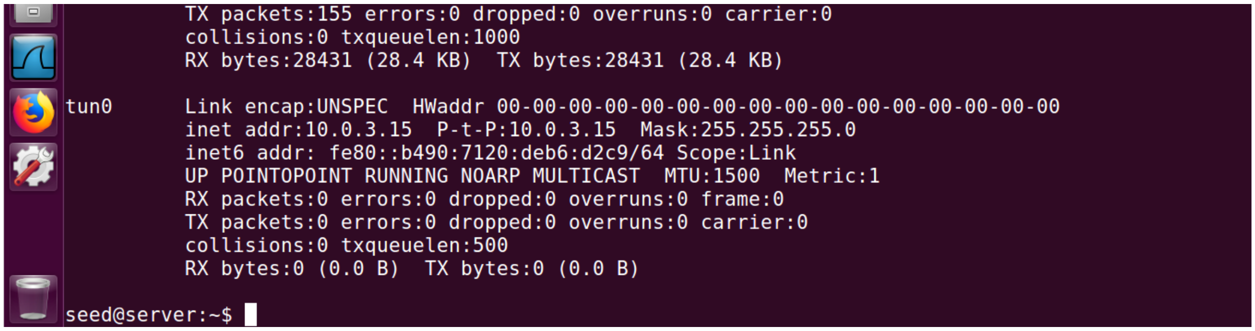

SEEDUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal

```
seed@VM:~$ cd lab3
seed@VM:~/lab3$ ls
simpletun    tunserver
simpletun.c  tunserver.c
seed@VM:~/lab3$ gcc -o tunserver tunserver.c
seed@VM:~/lab3$ sudo ./tunserver
[sudo] password for seed:
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
Got a packet from the tunnel
Got a packet from the tunnel
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 TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
```

Figure 3



```
seed@server:~$ sudo ip addr add 10.0.3.15/24 dev tun0
[sudo] password for seed:
seed@server:~$ sudo ifconfig tun0 up
seed@server:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:1b:50:8f
          inet addr:192.168.65.104  Bcast:192.168.65.255  Mask:255.255.255.0
          inet6 addr: fe80::ceea:ace2:492a:dbb8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31 errors:0 dropped:0 overruns:0 frame:0
          TX packets:53 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6752 (6.7 KB)  TX bytes:6500 (6.5 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:31:ed:40
          inet addr:10.0.2.12  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::5fe3:6ecc:6c24:649e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:51 errors:0 dropped:0 overruns:0 frame:0
          TX packets:84 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8072 (8.0 KB)  TX bytes:8790 (8.7 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:155 errors:0 dropped:0 overruns:0 frame:0
```

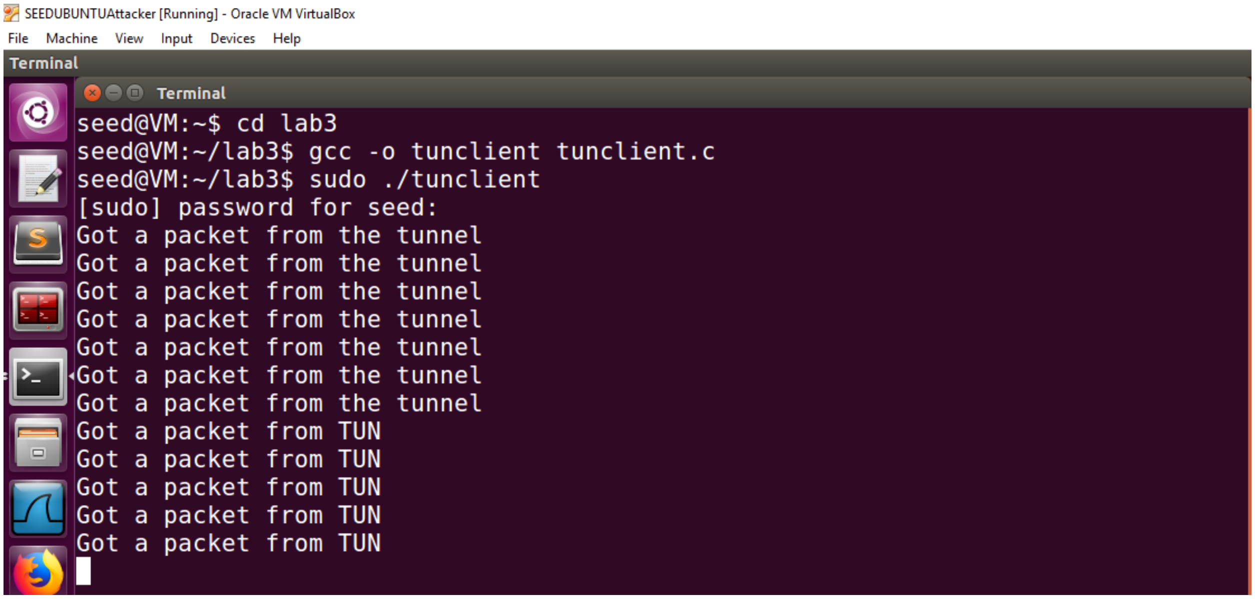


```
          TX packets:155 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:28431 (28.4 KB)  TX bytes:28431 (28.4 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.3.15  P-t-P:10.0.3.15  Mask:255.255.255.0
          inet6 addr: fe80::b490:7120:deb6:d2c9/64 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)

seed@server:~$
```

Figure 4

Observation: We execute the tunserver program on the server side of the tunnel. As seen in the above screenshot tun0 virtual network interface is created.

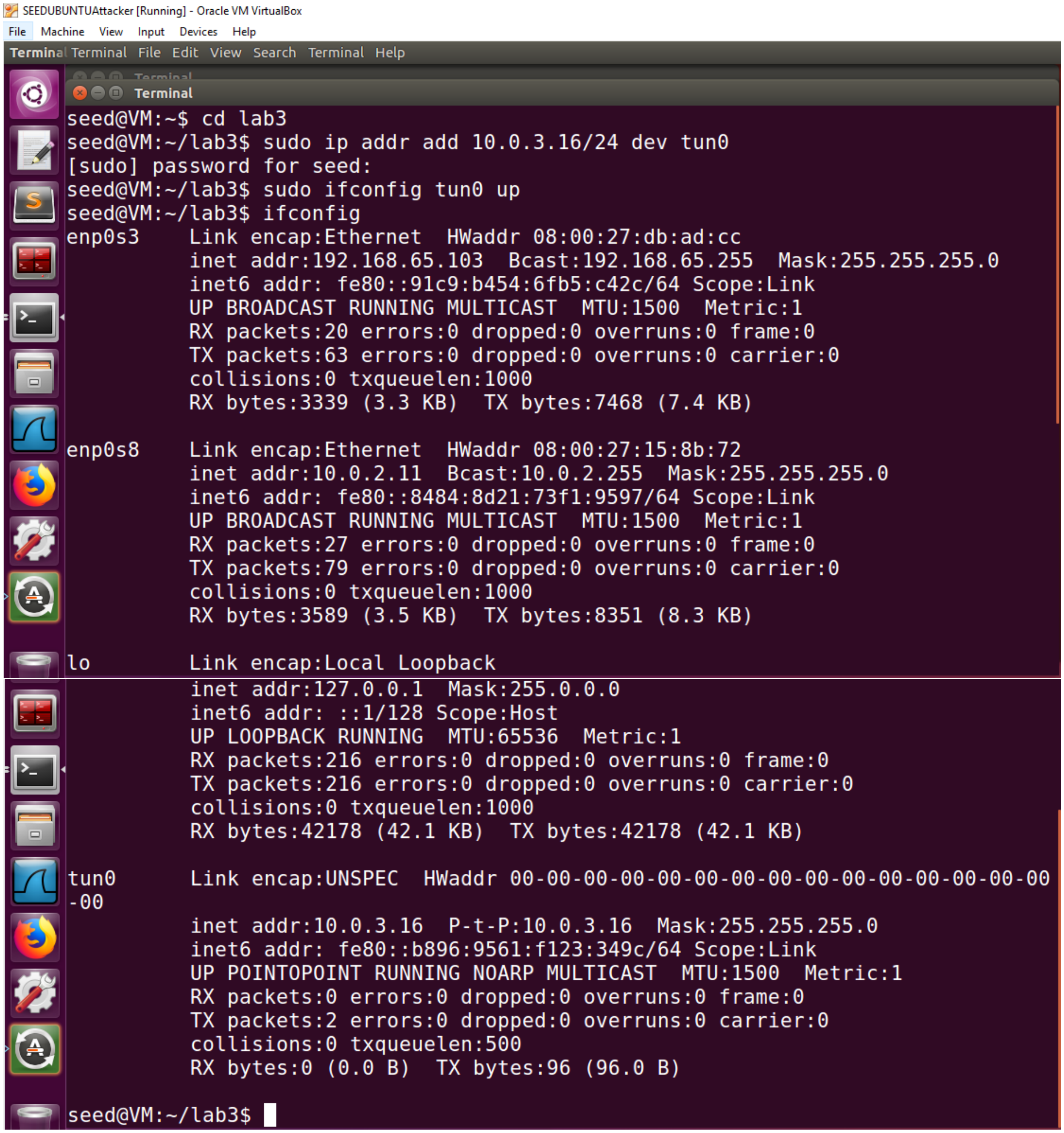
Explanation: Here tun0 is created and it is on the server side of the tunnel. After that we have to assign an IP address to the interface and then we have to bring up the interface.



SEEDUBUNTUAttacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
Terminal

```
seed@VM:~$ cd lab3
seed@VM:~/lab3$ gcc -o tunclient tunclient.c
seed@VM:~/lab3$ sudo ./tunclient
[sudo] password for seed:
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
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 TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
```

Figure 5

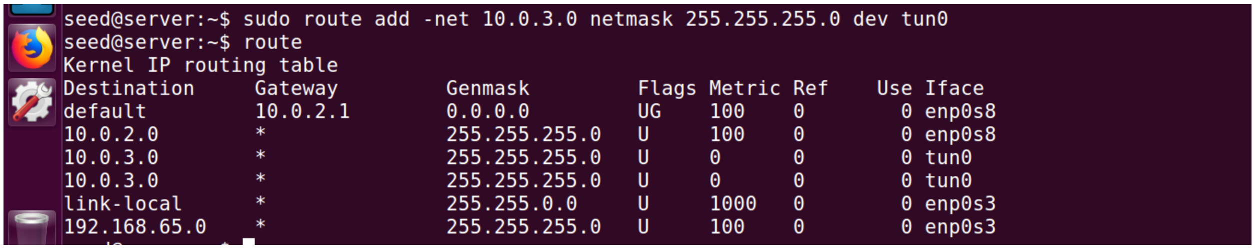

SEEDUBUNTUAttacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal Terminal File Edit View Search Terminal Help
Terminal

```
seed@VM:~$ cd lab3
seed@VM:~/lab3$ sudo ip addr add 10.0.3.16/24 dev tun0
[sudo] password for seed:
seed@VM:~/lab3$ sudo ifconfig tun0 up
seed@VM:~/lab3$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:db:ad:cc
          inet addr:192.168.65.103  Bcast:192.168.65.255  Mask:255.255.255.0
          inet6 addr: fe80::91c9:b454:6fb5:c42c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20 errors:0 dropped:0 overruns:0 frame:0
          TX packets:63 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3339 (3.3 KB)  TX bytes:7468 (7.4 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:15:8b:72
          inet addr:10.0.2.11  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::8484:8d21:73f1:9597/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:27 errors:0 dropped:0 overruns:0 frame:0
          TX packets:79 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3589 (3.5 KB)  TX bytes:8351 (8.3 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:216 errors:0 dropped:0 overruns:0 frame:0
          TX packets:216 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:42178 (42.1 KB)  TX bytes:42178 (42.1 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:10.0.3.16  P-t-P:10.0.3.16  Mask:255.255.255.0
          inet6 addr: fe80::b896:9561:f123:349c/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)

seed@VM:~/lab3$ 
```

Figure 6

Observation: We execute the tunclient program on the client side of the tunnel. As seen in the above screenshot tun0 virtual network interface is created.

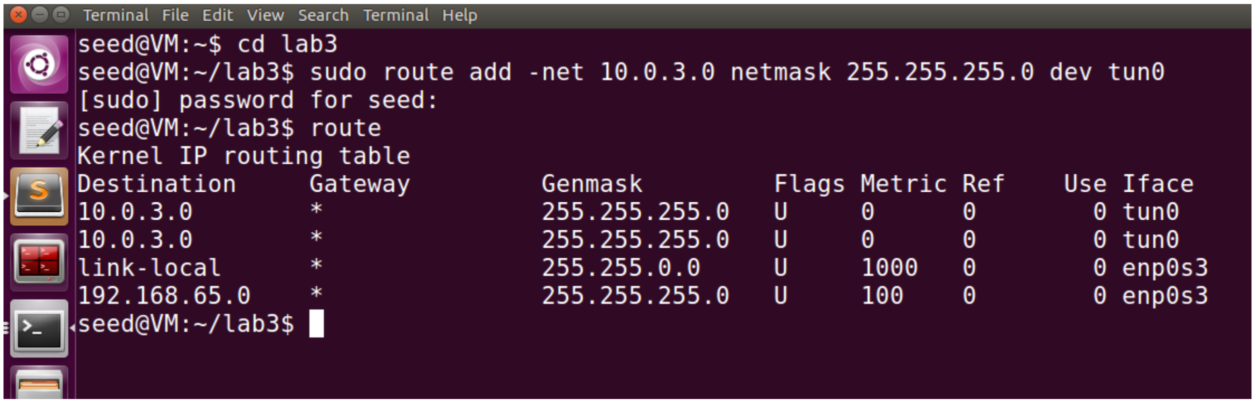
Explanation: Here tun0 is created and it is on the client side of the tunnel. After that we have to assign an IP address to the interface and then we have to bring up the interface.



```
seed@server:~$ sudo route add -net 10.0.3.0 netmask 255.255.255.0 dev tun0
seed@server:~$ 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 enp0s8
10.0.2.0        *               255.255.255.0   U     100    0        0 enp0s8
10.0.3.0        *               255.255.255.0   U     0      0        0 tun0
10.0.3.0        *               255.255.255.0   U     0      0        0 tun0
link-local      *               255.255.0.0     U     1000   0        0 enp0s3
192.168.65.0    *               255.255.255.0   U     100    0        0 enp0s3
```

Figure 7

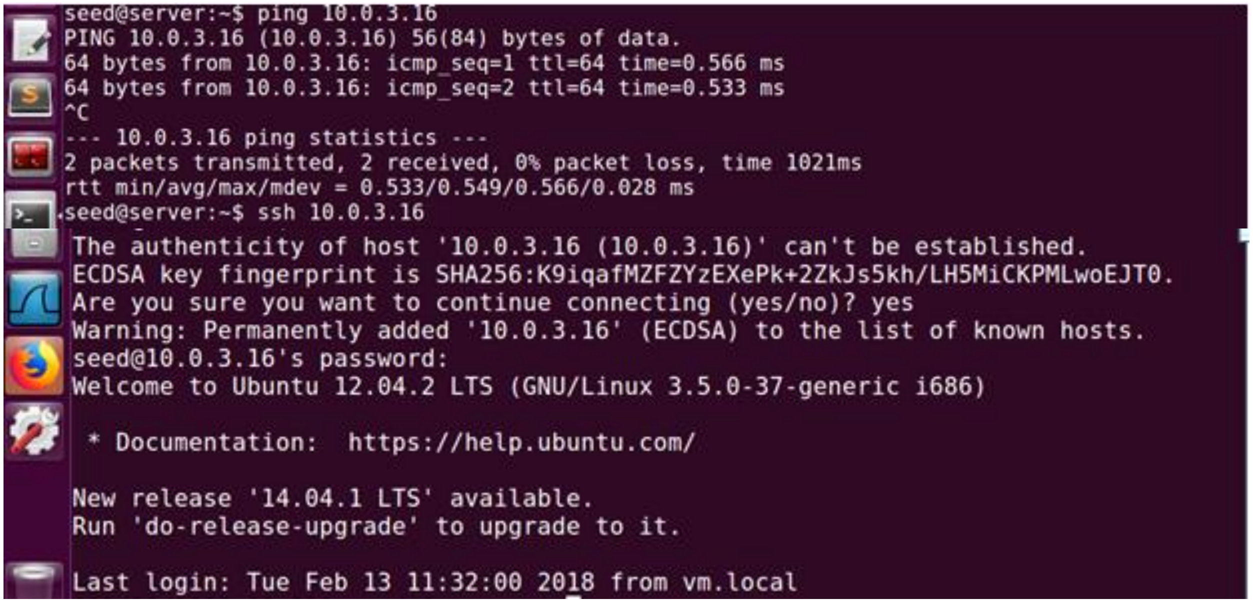
Observation: We add a routing path so that traffic can go through the tunnel and it directs all the packets for the 10.0.3.0/24 network through the interface tun0, from where the packet will be hauled through the tunnel. The screenshot shows that the route is added on the server side.



```
seed@VM:~$ cd lab3
seed@VM:~/lab3$ sudo route add -net 10.0.3.0 netmask 255.255.255.0 dev tun0
[sudo] password for seed:
seed@VM:~/lab3$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.3.0        *               255.255.255.0   U     0      0        0 tun0
10.0.3.0        *               255.255.255.0   U     0      0        0 tun0
link-local      *               255.255.0.0     U     1000   0        0 enp0s3
192.168.65.0    *               255.255.255.0   U     100    0        0 enp0s3
seed@VM:~/lab3$
```

Figure 8

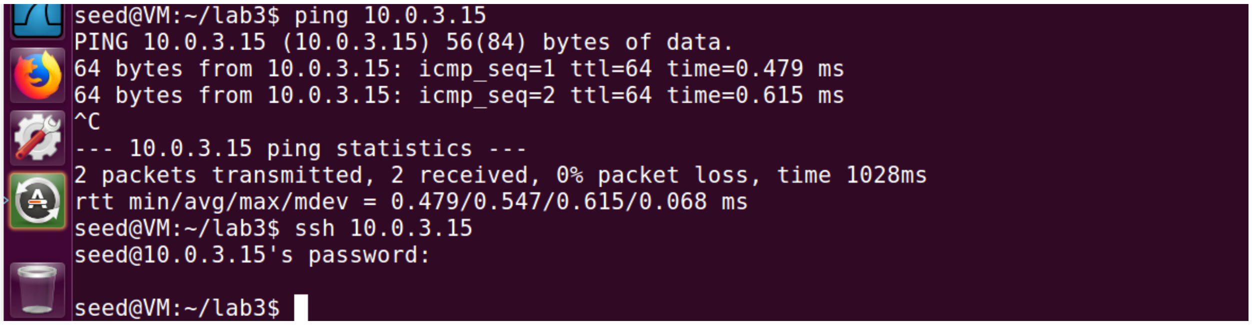
Observation: We add a routing path so that traffic can go through the tunnel and it directs all the packets for the 10.0.3.0/24 network through the interface tun0, from where the packet will be hauled through the tunnel. The screenshot shows that the route is added on the client side.



```
seed@server:~$ ping 10.0.3.16
PING 10.0.3.16 (10.0.3.16) 56(84) bytes of data.
64 bytes from 10.0.3.16: icmp_seq=1 ttl=64 time=0.566 ms
64 bytes from 10.0.3.16: icmp_seq=2 ttl=64 time=0.533 ms
^C
--- 10.0.3.16 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1021ms
rtt min/avg/max/mdev = 0.533/0.549/0.566/0.028 ms
seed@server:~$ ssh 10.0.3.16
The authenticity of host '10.0.3.16 (10.0.3.16)' can't be established.
ECDSA key fingerprint is SHA256:K9iqafMZFZYzEXePk+2ZkJs5kh/LH5MiCKPMLwoEJT0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.3.16' (ECDSA) to the list of known hosts.
seed@10.0.3.16's password:
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.5.0-37-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Feb 13 11:32:00 2018 from vm.local
```

Figure 9

Observation: After establishing the tunnel we can access 10.0.3.16 from 192.168.57.104. We then test using ping and establish a ssh connection.

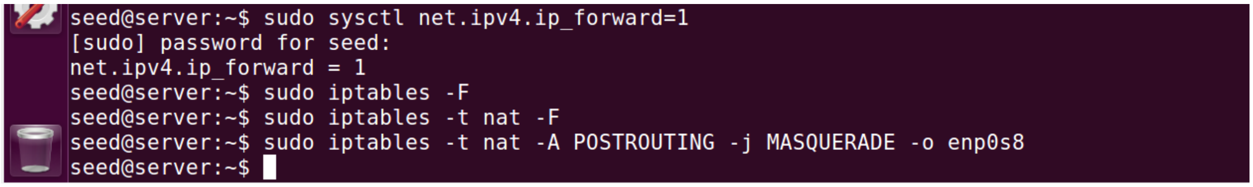


```
seed@VM:~/lab3$ ping 10.0.3.15
PING 10.0.3.15 (10.0.3.15) 56(84) bytes of data.
64 bytes from 10.0.3.15: icmp_seq=1 ttl=64 time=0.479 ms
64 bytes from 10.0.3.15: icmp_seq=2 ttl=64 time=0.615 ms
^C
--- 10.0.3.15 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1028ms
rtt min/avg/max/mdev = 0.479/0.547/0.615/0.068 ms
seed@VM:~/lab3$ ssh 10.0.3.15
seed@10.0.3.15's password:

seed@VM:~/lab3$
```

Figure 10

Observation: After establishing the tunnel we can access 10.0.3.15 from 192.168.57.103. We then test using ping and establish a ssh connection.

Task 2: Set up Host-to-Gateway Tunnel



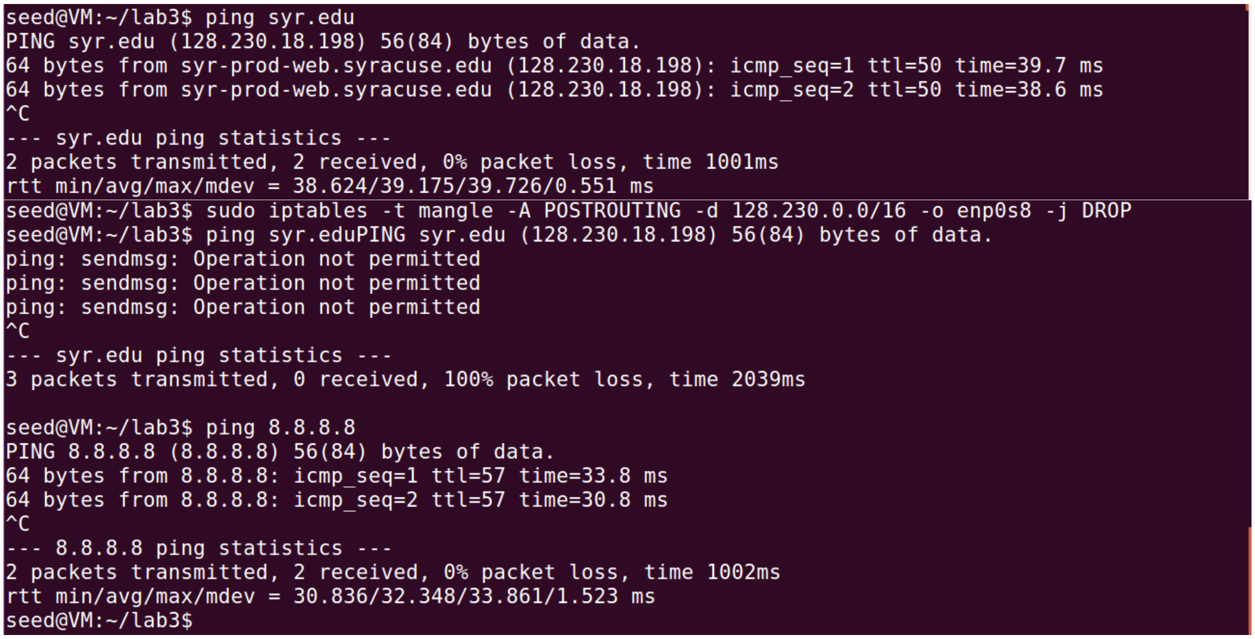
```
seed@server:~$ sudo sysctl net.ipv4.ip_forward=1
[sudo] password for seed:
net.ipv4.ip_forward = 1
seed@server:~$ sudo iptables -F
seed@server:~$ sudo iptables -t nat -F
seed@server:~$ sudo iptables -t nat -A POSTROUTING -j MASQUERADE -o enp0s8
seed@server:~$
```

Figure 11

Observation: The commands in the above screenshot sets up 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.

Explanation: The packets received at the server end is not meant to stay at the server, it must be forwarded. This is why we set up ip forwarding. After that we have go around the limitation of NAT by adding a new rule in the postrouting position to the nat network adapter connected to the VPN server.

Task 3: Set up Firewall



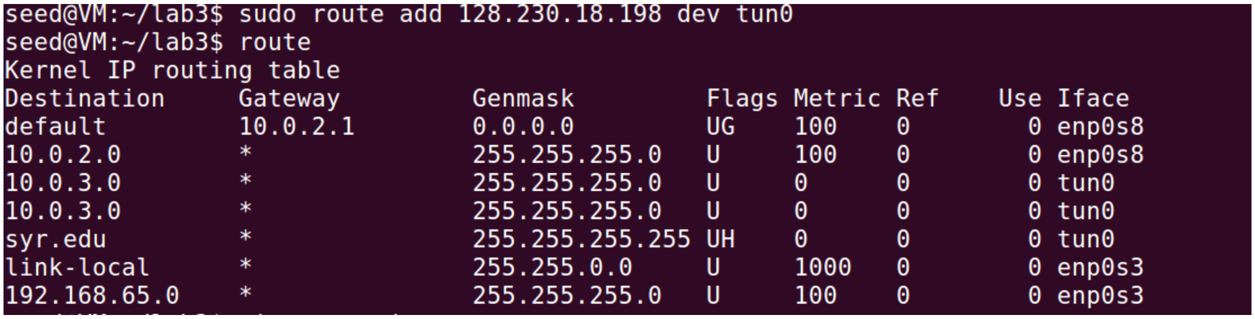
```
seed@VM:~/lab3$ ping syr.edu
PING syr.edu (128.230.18.198) 56(84) bytes of data.
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=1 ttl=50 time=39.7 ms
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=2 ttl=50 time=38.6 ms
^C
--- syr.edu ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 38.624/39.175/39.726/0.551 ms
seed@VM:~/lab3$ sudo iptables -t mangle -A POSTROUTING -d 128.230.0.0/16 -o enp0s8 -j DROP
seed@VM:~/lab3$ ping syr.eduPING syr.edu (128.230.18.198) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- syr.edu ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2039ms

seed@VM:~/lab3$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=57 time=33.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=57 time=30.8 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 30.836/32.348/33.861/1.523 ms
seed@VM:~/lab3$
```

Figure 12

Observation: We set up a firewall in the client machine to block all packets to 128.230.0.0/16. So when we ping to syr.edu, it will be blocked. When we ping to google, it is allowed as seen in the above screenshot.

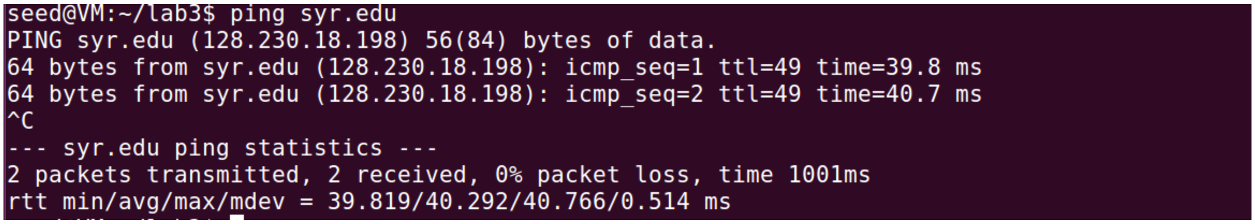
Explanation: We use iptables mangle to block all the traffic going to 128.230.0.0/16 from the client machine.

Task 4: Bypassing Firewall



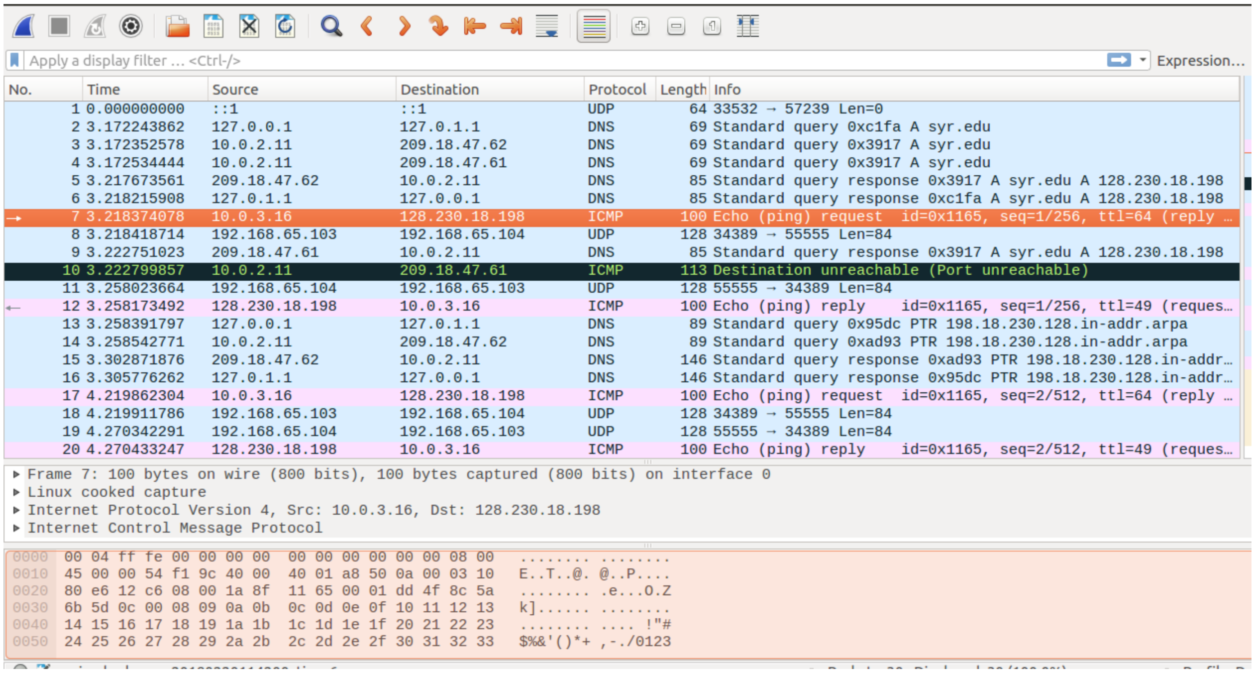
```
seed@VM:~/lab3$ sudo route add 128.230.18.198 dev tun0
seed@VM:~/lab3$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.2.1        0.0.0.0         UG    100    0        0 enp0s8
10.0.2.0        *               255.255.255.0   U     100    0        0 enp0s8
10.0.3.0        *               255.255.255.0   U     0      0        0 tun0
10.0.3.0        *               255.255.255.0   U     0      0        0 tun0
syr.edu         *               255.255.255.255 UH    0      0        0 tun0
link-local      *               255.255.0.0     U     1000   0        0 enp0s3
192.168.65.0    *               255.255.255.0   U     100    0        0 enp0s3
```

Figure 13



```
seed@VM:~/lab3$ ping syr.edu
PING syr.edu (128.230.18.198) 56(84) bytes of data.
64 bytes from syr.edu (128.230.18.198): icmp_seq=1 ttl=49 time=39.8 ms
64 bytes from syr.edu (128.230.18.198): icmp_seq=2 ttl=49 time=40.7 ms
^C
--- syr.edu ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 39.819/40.292/40.766/0.514 ms
```

Figure 14



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	::1	::1	UDP	64	33532 → 57239 Len=0
2	3.172243862	127.0.0.1	127.0.1.1	DNS	69	Standard query 0xc1fa A syr.edu
3	3.172352578	10.0.2.11	209.18.47.62	DNS	69	Standard query 0x3917 A syr.edu
4	3.172534444	10.0.2.11	209.18.47.61	DNS	69	Standard query 0x3917 A syr.edu
5	3.217673561	209.18.47.62	10.0.2.11	DNS	85	Standard query response 0x3917 A syr.edu A 128.230.18.198
6	3.218215908	127.0.1.1	127.0.0.1	DNS	85	Standard query response 0xc1fa A syr.edu A 128.230.18.198
7	3.218374078	10.0.3.16	128.230.18.198	ICMP	100	Echo (ping) request id=0x1165, seq=1/256, ttl=64 (reply …
8	3.218418714	192.168.65.103	192.168.65.104	UDP	128	34389 → 55555 Len=84
9	3.222751023	209.18.47.61	10.0.2.11	DNS	85	Standard query response 0x3917 A syr.edu A 128.230.18.198
10	3.222799857	10.0.2.11	209.18.47.61	ICMP	113	Destination unreachable (Port unreachable)
11	3.258023664	192.168.65.104	192.168.65.103	UDP	128	55555 → 34389 Len=84
12	3.258173492	128.230.18.198	10.0.3.16	ICMP	100	Echo (ping) reply id=0x1165, seq=1/256, ttl=49 (reques…
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	3.302871876	209.18.47.62	10.0.2.11	DNS	146	Standard query response 0xad93 PTR 198.18.230.128.in-addr…
16	3.305776262	127.0.1.1	127.0.0.1	DNS	146	Standard query response 0x95dc PTR 198.18.230.128.in-addr…
17	4.219862304	10.0.3.16	128.230.18.198	ICMP	100	Echo (ping) request id=0x1165, seq=2/512, ttl=64 (reply …
18	4.219911786	192.168.65.103	192.168.65.104	UDP	128	34389 → 55555 Len=84
19	4.270342291	192.168.65.104	192.168.65.103	UDP	128	55555 → 34389 Len=84
20	4.270433247	128.230.18.198	10.0.3.16	ICMP	100	Echo (ping) reply id=0x1165, seq=2/512, ttl=49 (reques…

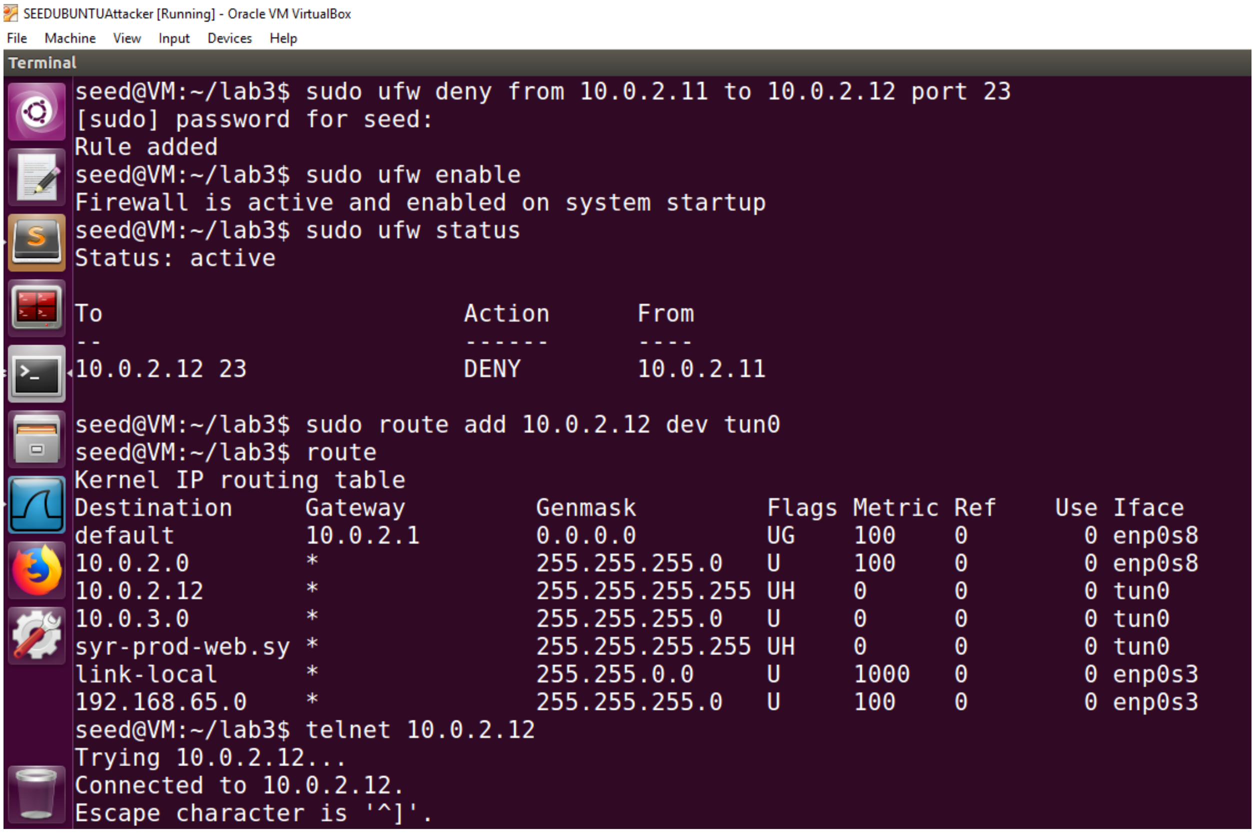
```
▶ Frame 7: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0
▶ Linux cooked capture
▶ Internet Protocol Version 4, Src: 10.0.3.16, Dst: 128.230.18.198
▶ Internet Control Message Protocol
```

```
0000  00 04 ff fe 00 00 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 54 f1 9c 40 00  40 01 a8 50 0a 00 03 10   E..T..@. @..P....
0020  80 e6 12 c6 08 00 1a 8f  11 65 00 01 dd 4f 8c 5a   ........ .e...O.Z
0030  6b 5d 0c 00 08 09 0a 0b  0c 0d 0e 0f 10 11 12 13   k]...... ........
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   $%&'()*+ ,-./0123
```

Figure 15

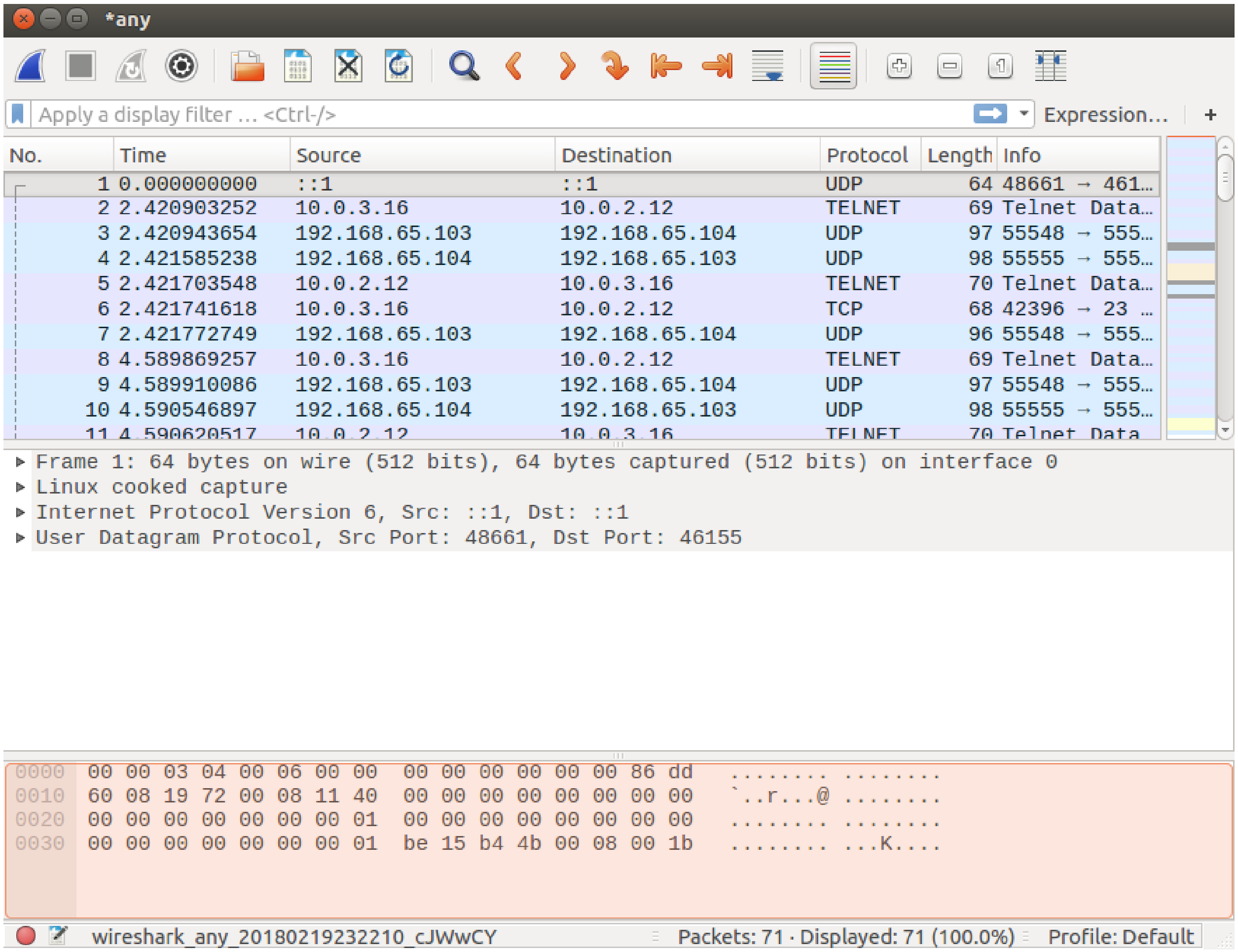
Observation: When we ping to 128.230.18.198, the ping is successful and we get a reply. The wireshark capture is evidence of this.

Explanation: The packets to 128.230.18.198 will go through the tun interface to the client side of the tunnel. The packets are encrypted there and then sent through the interface to the server side of the tunnel. The packets are then sent to the destination from the VPN server.


SEEDUBUNTUAttacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal

```
seed@VM:~/lab3$ sudo ufw deny from 10.0.2.11 to 10.0.2.12 port 23
[sudo] password for seed:
Rule added
seed@VM:~/lab3$ sudo ufw enable
Firewall is active and enabled on system startup
seed@VM:~/lab3$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
10.0.2.12 23               DENY        10.0.2.11

seed@VM:~/lab3$ sudo route add 10.0.2.12 dev tun0
seed@VM:~/lab3$ 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 enp0s8
10.0.2.0        *               255.255.255.0   U     100    0        0 enp0s8
10.0.2.12       *               255.255.255.255 UH    0      0        0 tun0
10.0.3.0        *               255.255.255.0   U     0      0        0 tun0
syr-prod-web.sy *               255.255.255.255 UH    0      0        0 tun0
link-local      *               255.255.0.0     U     1000   0        0 enp0s3
192.168.65.0    *               255.255.255.0   U     100    0        0 enp0s3
seed@VM:~/lab3$ telnet 10.0.2.12
Trying 10.0.2.12...
Connected to 10.0.2.12.
Escape character is '^]'.
```

Figure 16



```
No.  Time          Source           Destination      Protocol  Length Info
   1 0.000000000   ::1              ::1              UDP           64 48661 → 461…
   2 2.420903252   10.0.3.16        10.0.2.12        TELNET        69 Telnet Data…
   3 2.420943654   192.168.65.103   192.168.65.104   UDP           97 55548 → 555…
   4 2.421585238   192.168.65.104   192.168.65.103   UDP           98 55555 → 555…
   5 2.421703548   10.0.2.12        10.0.3.16        TELNET        70 Telnet Data…
   6 2.421741618   10.0.3.16        10.0.2.12        TCP           68 42396 → 23 …
   7 2.421772749   192.168.65.103   192.168.65.104   UDP           96 55548 → 555…
   8 4.589869257   10.0.3.16        10.0.2.12        TELNET        69 Telnet Data…
   9 4.589910086   192.168.65.103   192.168.65.104   UDP           97 55548 → 555…
  10 4.590546897   192.168.65.104   192.168.65.103   UDP           98 55555 → 555…
  11 4.590620517   10.0.2.12        10.0.3.16        TELNET        70 Telnet Data
```

```
▸ Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 6, Src: ::1, Dst: ::1
▸ User Datagram Protocol, Src Port: 48661, Dst Port: 46155
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 86 dd   ........ ........
0010  60 08 19 72 00 08 11 40  00 00 00 00 00 00 00 00   `..r...@ ........
0020  00 00 00 00 00 00 00 01  00 00 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 01  be 15 b4 4b 00 08 00 1b   ........ ...K....
```

```
wireshark_any_20180219232210_cJWwCY    Packets: 71 · Displayed: 71 (100.0%)    Profile: Default
```

Figure 17

Observation: We create a ufw rule to block telnet from 10.0.2.11 to 10.0.2.12. We then add a route so that all packets go through the tun interface when they have to establish a telnet between the systems. The telnet is successful and we get a reply. The wireshark capture is evidence of this.

Explanation: The packets to 10.0.2.12 will go through the tun interface to the client side of the tunnel. The packets are encrypted there and then sent through the interface to the server side of the tunnel. The packets are then sent to the destination from the VPN server.