Lab 3
Gunnar Yonker

Part 1: Tasks 1,3, and 4

Task 1: Using Firewall(ufw)

A: 10.0.2.15

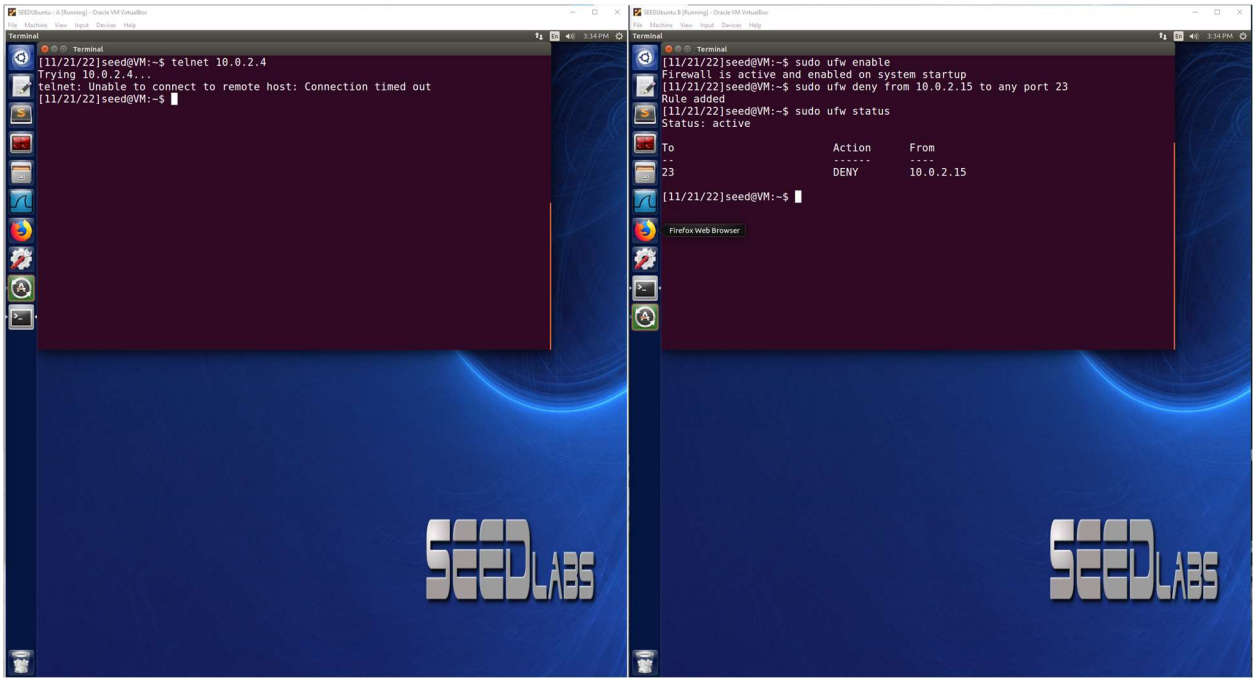
B: 10.0.2.4

C: 10.0.2.5

- Prevent A from doing telnet to Machine B

Firewall rule set on B: sudo ufw deny from 10.0.2.15 to any port 23

A is no longer able to telnet to Machine B

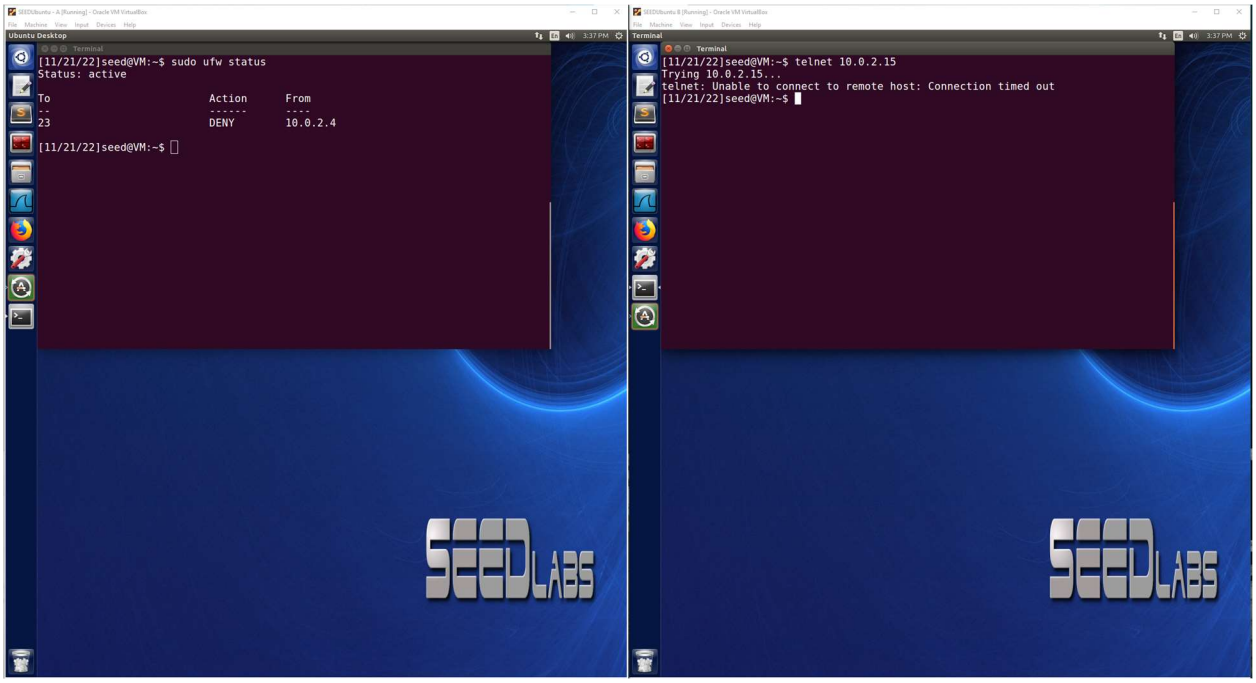


Lab 3

- Prevent B from doing telnet to Machine A

Firewall rule set on A: sudo ufw deny from 10.0.2.4 to any port 23

B is no longer able to telnet to Machine A

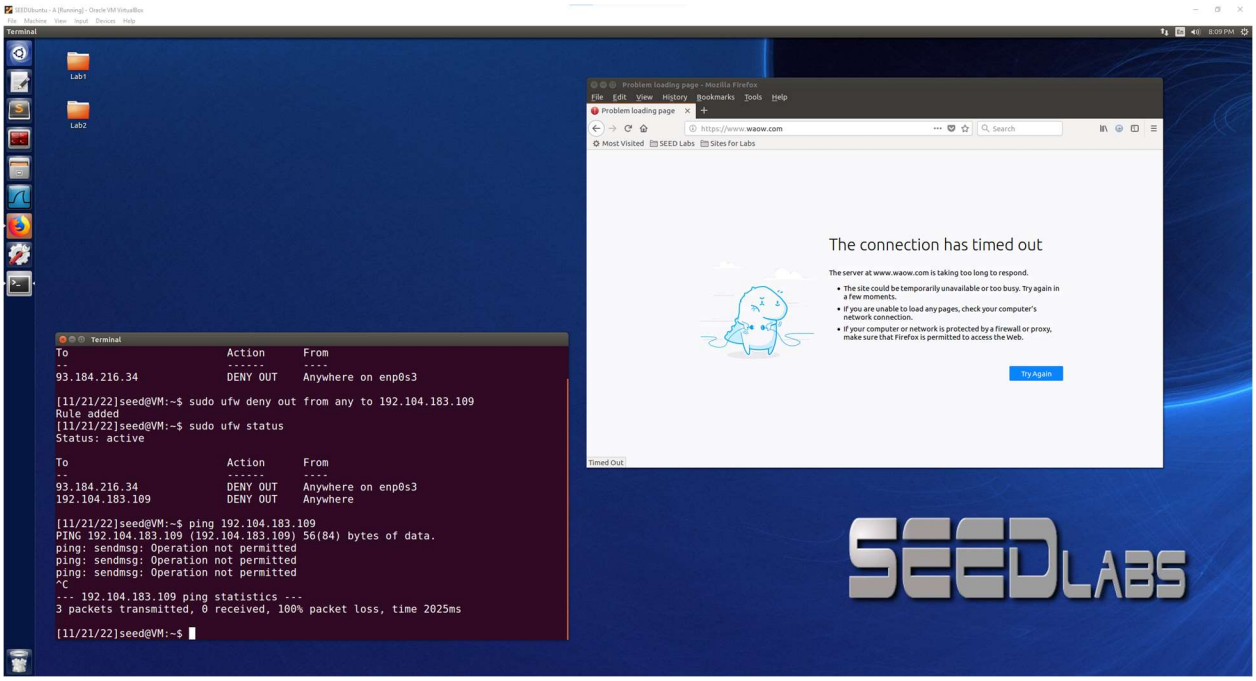


Lab 3

- Prevent A from visiting an external web site.

Blocking www.waow.com IP: 192.104.183.109

ufw command: sudo ufw deny out from any to 192.104.183.109



Lab 3

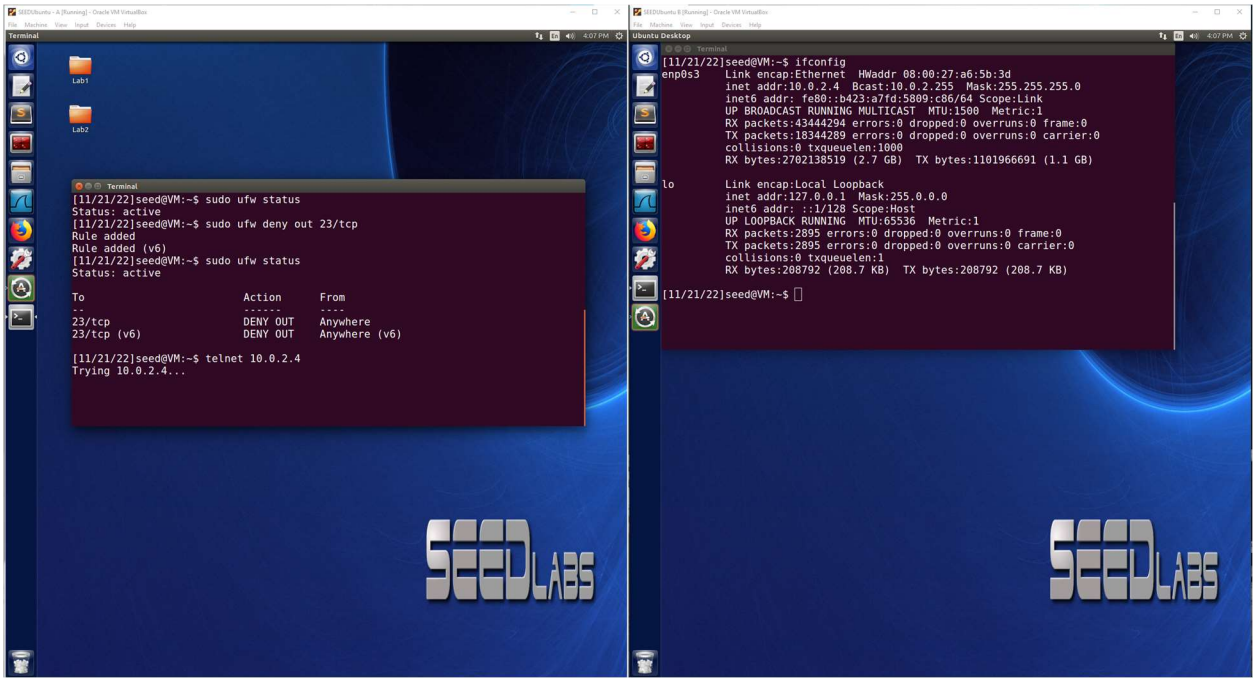
Task 3: Evading Egress Filtering

A: 10.0.2.15(inside firewall, also running firewall)

B: 10.0.2.4(outside firewall)

- Block all outgoing traffic to external telnet servers

sudo ufw deny out 23/tcp

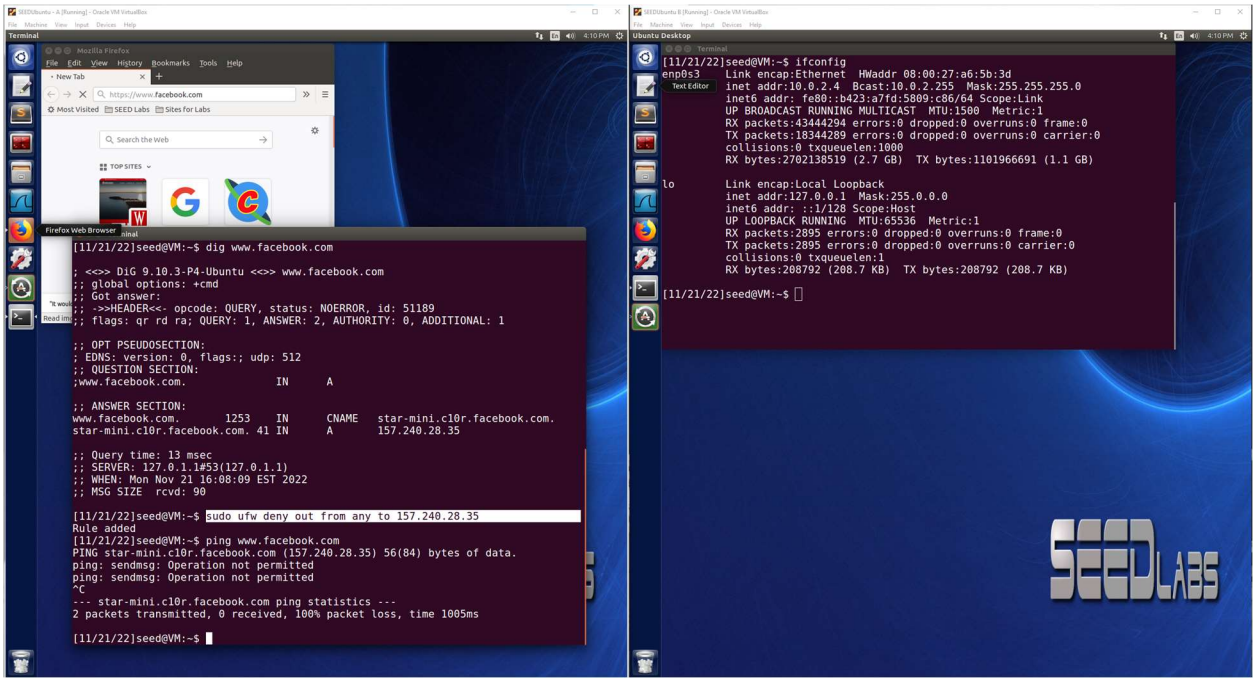


Lab 3

- Block all the outgoing traffic to www.facebook.com(157.240.28.35)

sudo ufw deny out from any to 157.240.28.35

Ping successfully blocked and unable to connect to www.facebook.com on web browser



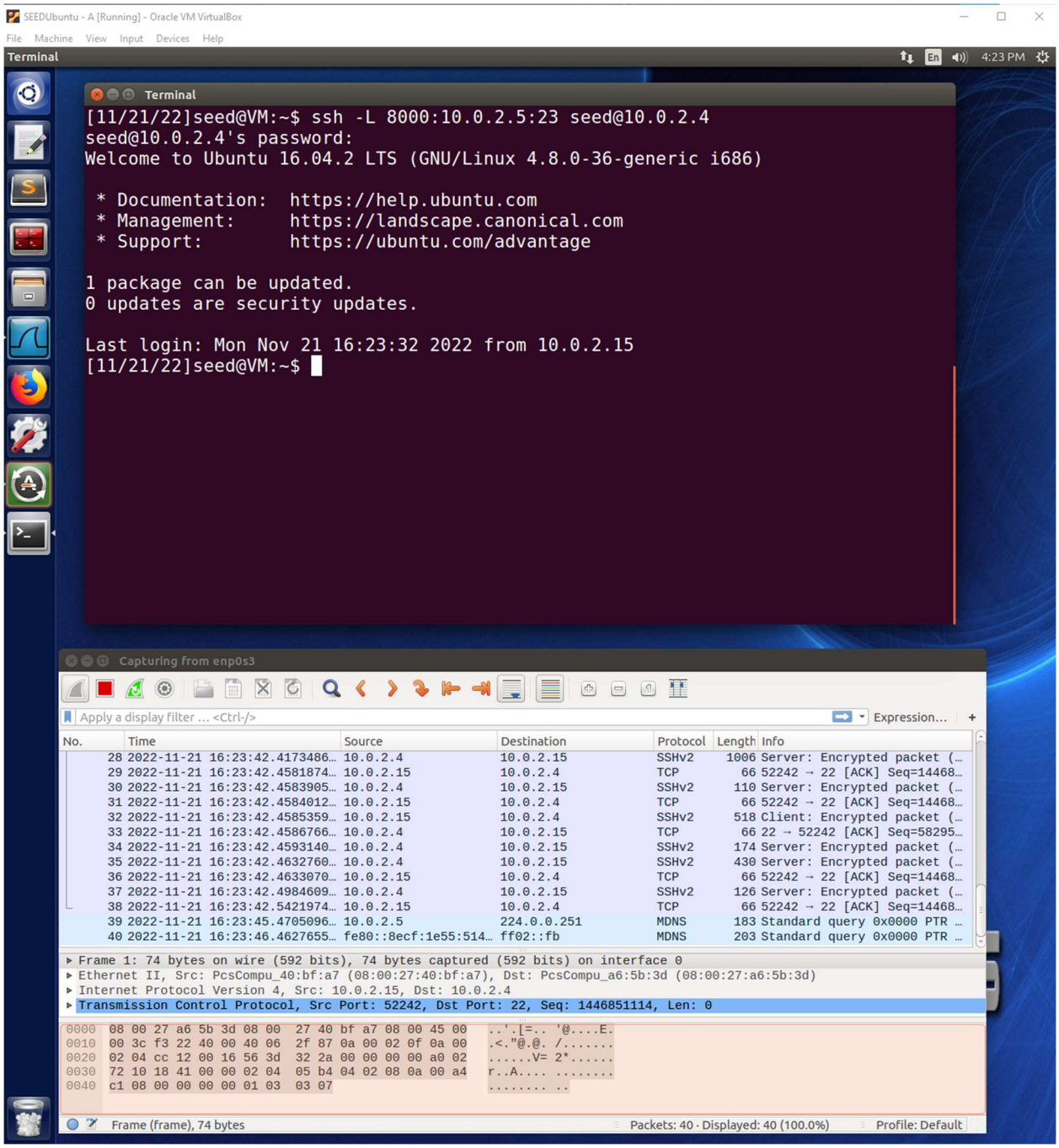
Lab 3

Task 3.a: Telnet to Machine C through the Firewall

A: 10.0.2.15(inside firewall, also running firewall)

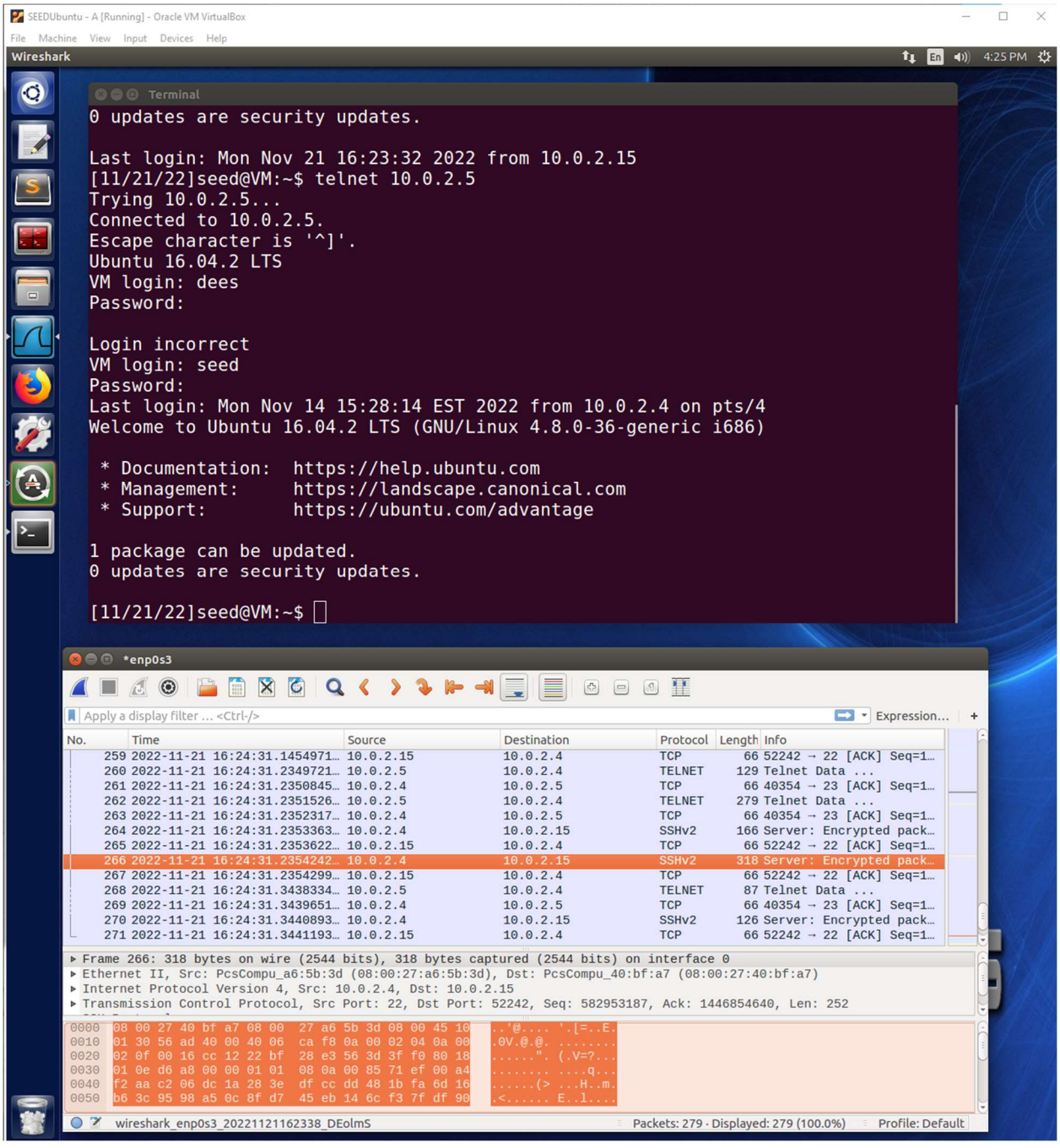
B: 10.0.2.4(outside firewall)

C: 10.0.2.5(outside firewall)



In this first image the Wireshark window shows that an ssh tunnel was opened up between Machine A and Machine B using encrypted packets. This means that the tunneling was successful between Machine

A and Machine B using port 8000 for the host and port 22 for Machine B which is not blocked by the firewall. Now when the packet is sent from Machine B(outside of the firewall) to telnet to Machine C through port 23, it will not be blocked by the firewall and Machine A is able to telnet to Machine C bypassing the firewall.

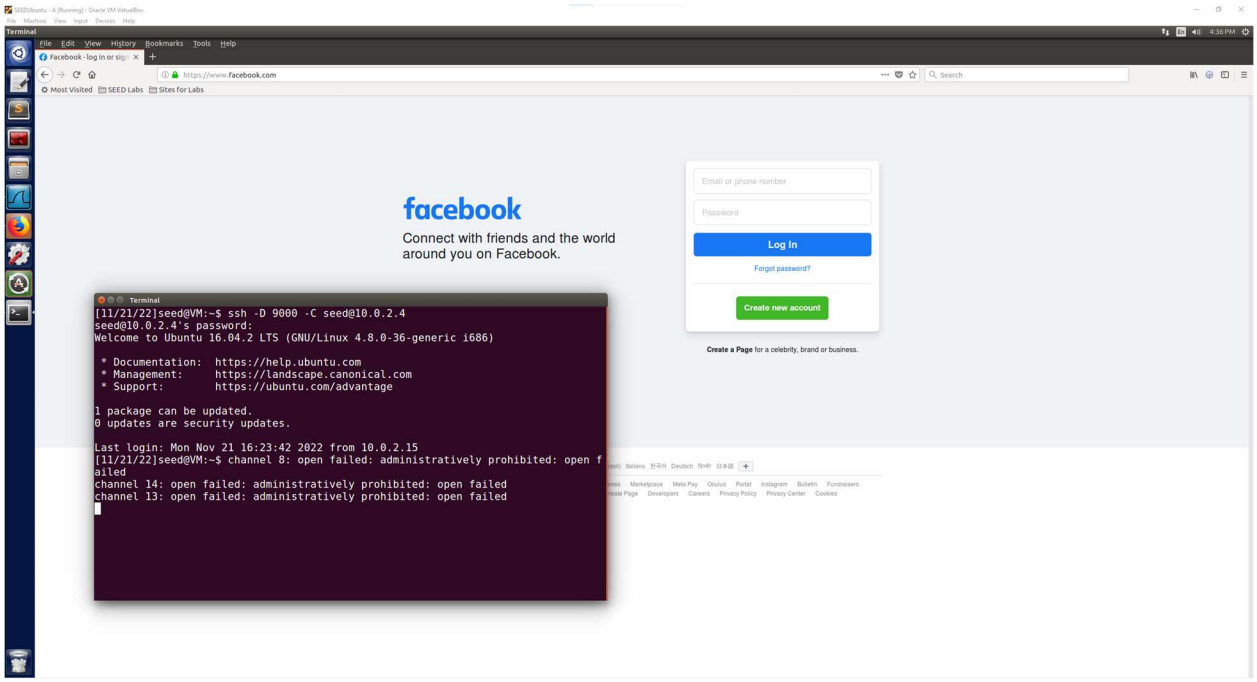


The connection is shown in this above image with Machine A being able to form a successful telnet connection to Machine C, by use of the ssh tunnel. Wireshark shows the telnet commands between Machine B and C which are outside of the firewall, but on machine A by use of the ssh tunnel sending

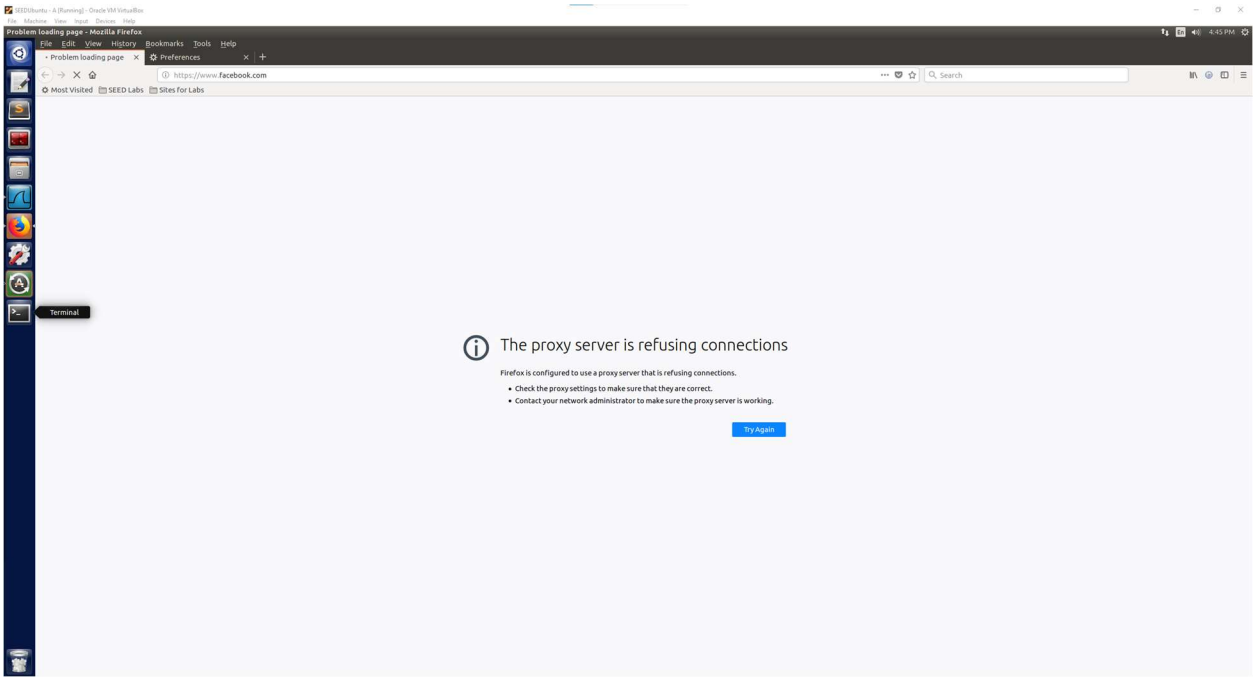
encrypted communication to B using port 22(not blocked by firewall) we can successfully telnet from A to C because the packets are forwarded from A to C through the ssh connection with B. This was able to be done because the packets sent from A to B are not blocked by the firewall on port 22, and then B which is outside of the firewall can forward the packets to port 23 on C and allow the telnet connection to take place.

Task 3.b: Connect to Facebook using SSH Tunnel

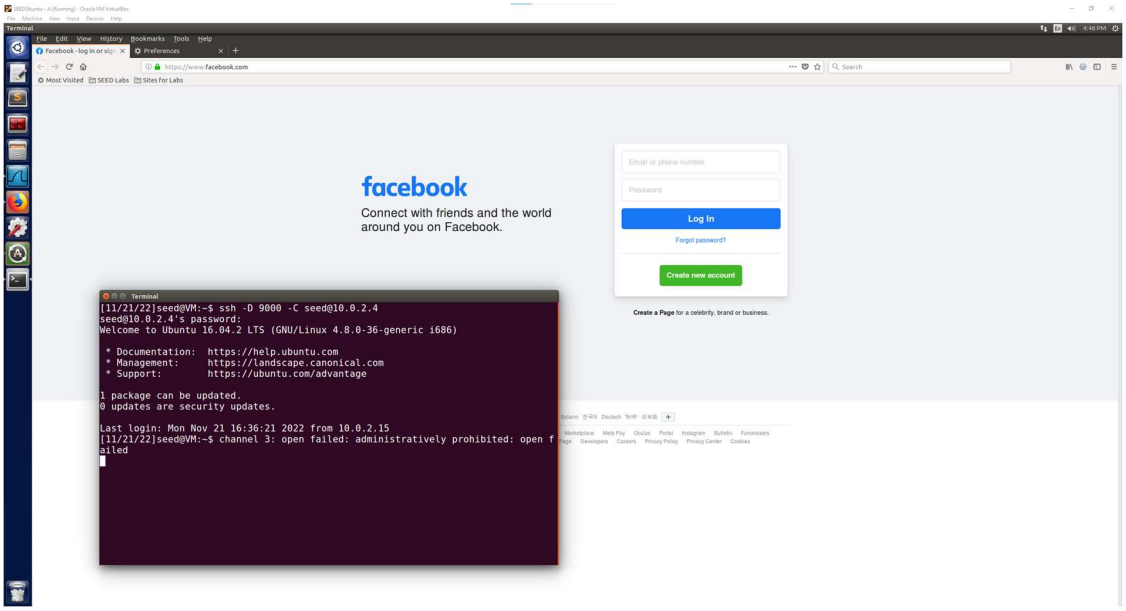
(1) After establishing the ssh tunnel to Machine B from Machine A and then configuring the network proxy in Firefox, the user on Machine A is then able to access Facebook successfully as seen below. This is because when A goes to connect to the web server through Firefox, the traffic first goes through the ssh tunnel to Machine B(outside the firewall) and then is able to connect to the webserver bypassing the firewall. The proxy is needed so that Firefox knows to connect to localhost:9000 which is the ssh tunnel before connecting to the web server.



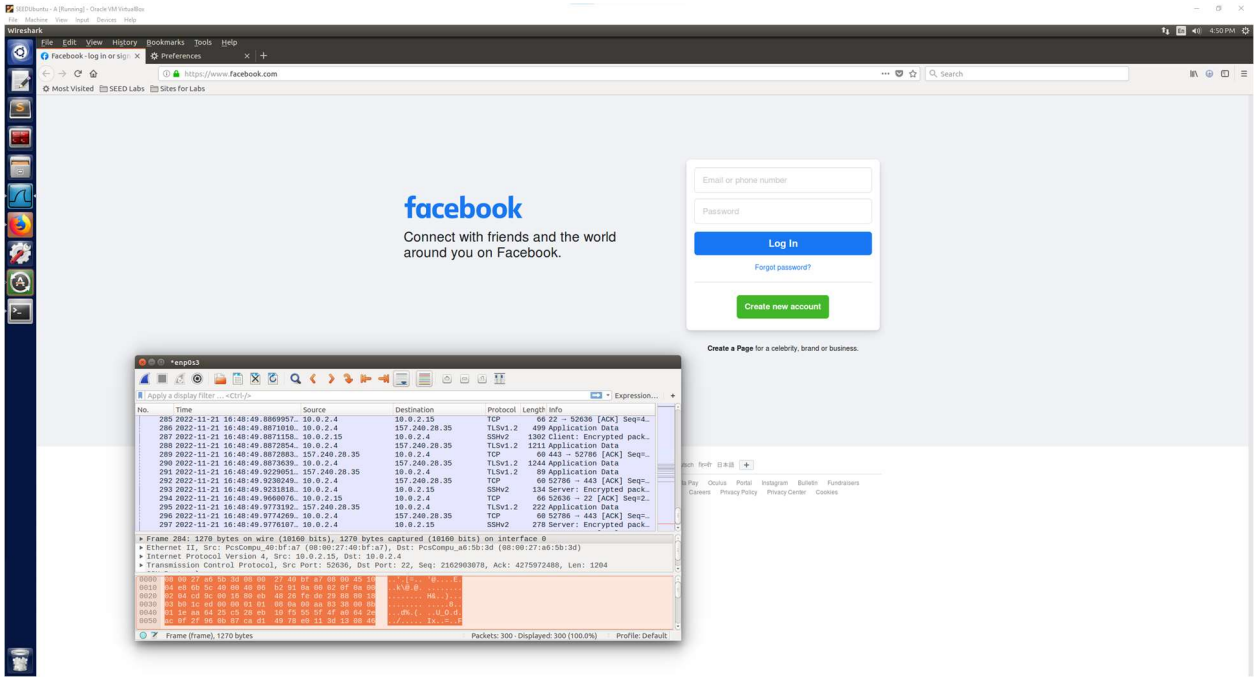
(2) With the SSH tunnel broken and the Firefox cache cleared, upon refreshing the page, the page says that the proxy server is refusing connections. This is because the proxy server is still setup in the Firefox proxy settings, but we no longer have the SSH tunnel connected so there is no proxy to connect to before connecting to Facebook. Without clearing the network settings, no connection would be successful regardless of the firewall rules because the intended proxy server is no longer connected. If you change to no proxy on Firefox, then Facebook will no longer load either due to the firewall blocking the connection and pings will also fail.



(3) With the SSH tunnel established again and the network proxy settings in place on Firefox, the Facebook page is again successfully shown for Machine A and the firewall is bypassed using the SSH tunnel that is establish to Machine B.



(4) When using Wireshark to observe what is happening with the given packets, we can see that the SSH tunnel is established between Machine A and Machine B. Now when a TCP packet is sent from Machine A trying to connect to Facebook(157.240.28.35) that packet is first passed on to Machine B through the SSH tunnel. In the image below an example of this is packet 287 where A is sending a packet to Machine B using the SSH tunnel. Then we can see that Machine B forwards that packet to the Facebook IP address and successfully communicates packets back and forth with Facebook. Then in packet 293, we can see that Machine B communicates back to Machine A. This is how the information is transferred between Machine A and Facebook, using the SSH tunnel to Machine B. All of the packets are first sent to B from A using SSH and then B sends those packets to their destination and communicates the response back to A through the SSH tunnel.



Lab 3

Task 4: Evading Ingress Filtering

A: 10.0.2.15(internal, running web server)

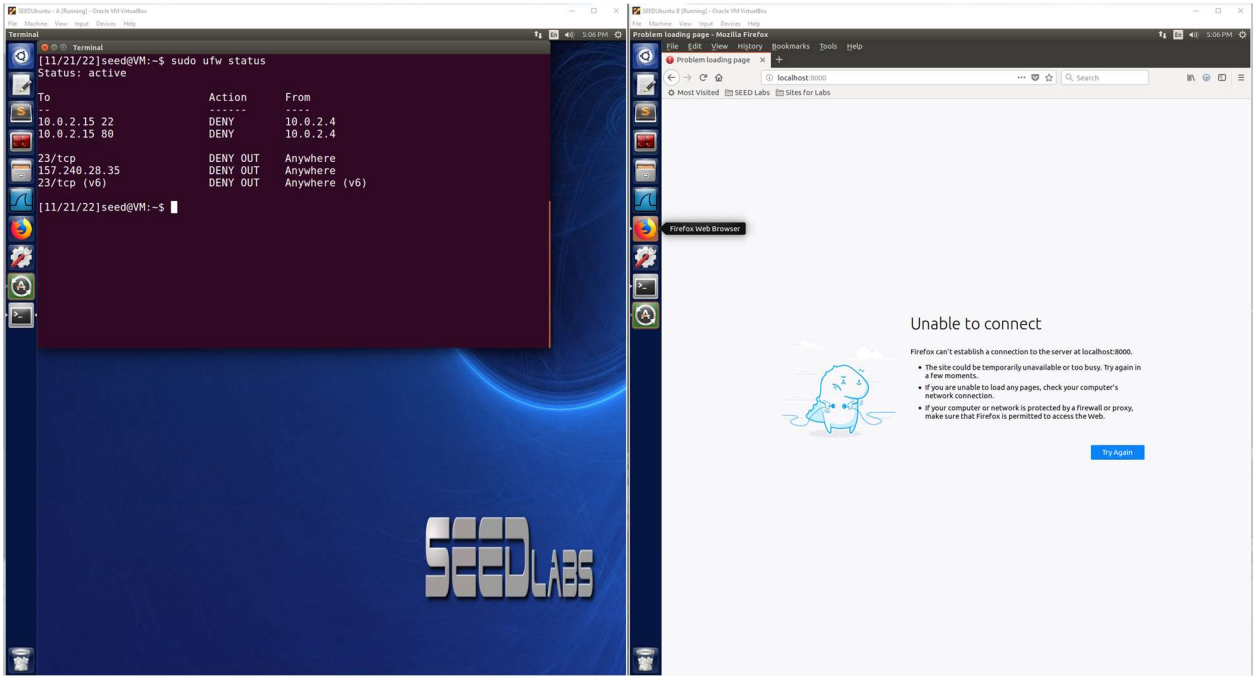
B: 10.0.2.4(outside machine at home)

Blocking port 80 and port 22 from B accessing A

sudo ufw deny in from 10.0.2.4 to 10.0.2.15 port 22

sudo ufw deny in from 10.0.2.4 to 10.0.2.15 port 80

Before setting up the reverse SSH tunnel on Machine A, on Machine B if we try to access localhost:8000 we get the image below which shows that we are unable to access the protected web server(apache2 for this example).

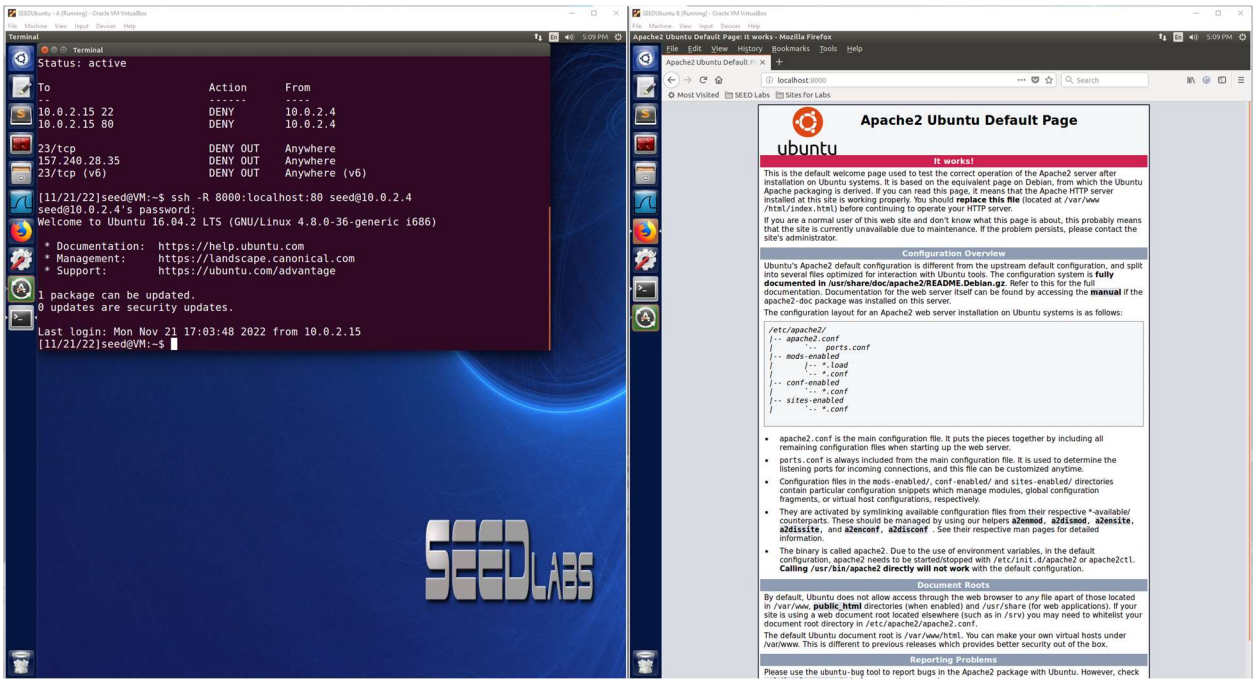


Lab 3

After establishing the reverse SSH tunnel on Machine A:

ssh -R 8000:localhost:80 seed@10.0.2.4

Now we can head to Machine B, our at home machine and we can check to make sure that the reverse SSH tunnel was successfully established by again trying to access the protected web server that is on Machine A by going to localhost:8000. We are successfully able to access the protected web server(apache2 webpage in this case) using the reverse SSH tunneling method that we set up on Machine A. This means that we are still able to access the protected web server on A from our home Machine B using the reverse SSH tunneling command.



Lab 3

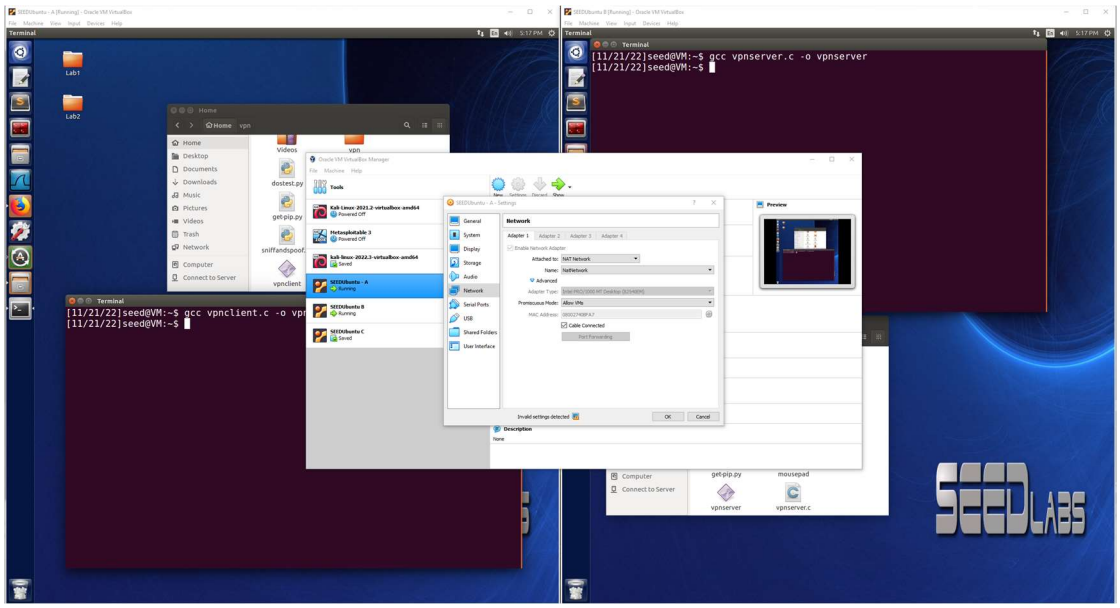
Part 2: Firewall Evasion using VPN – All Tasks

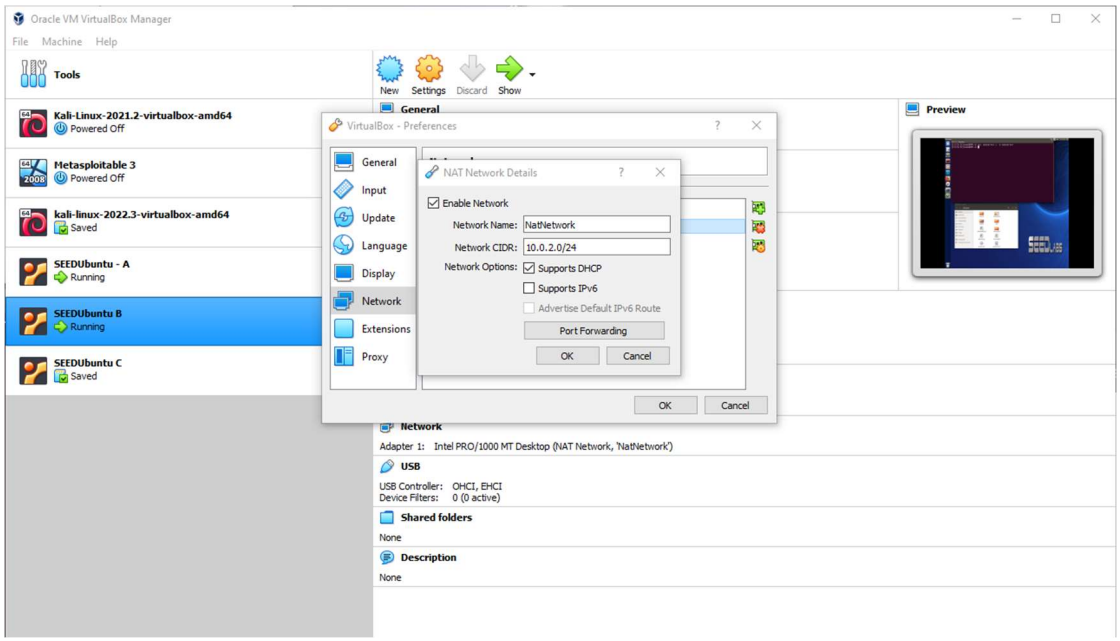
Task 1: VM Setup

VM1: 10.0.2.15(inside firewall, vpnclient)

VM2: 10.0.2.4(outside firewall, vpnserver)

Nat Network Adapter is setup for these two VMs from the original SEED set up document we received at the beginning of the class. The given settings are below for the NatNetwork that both VMs are setup with.





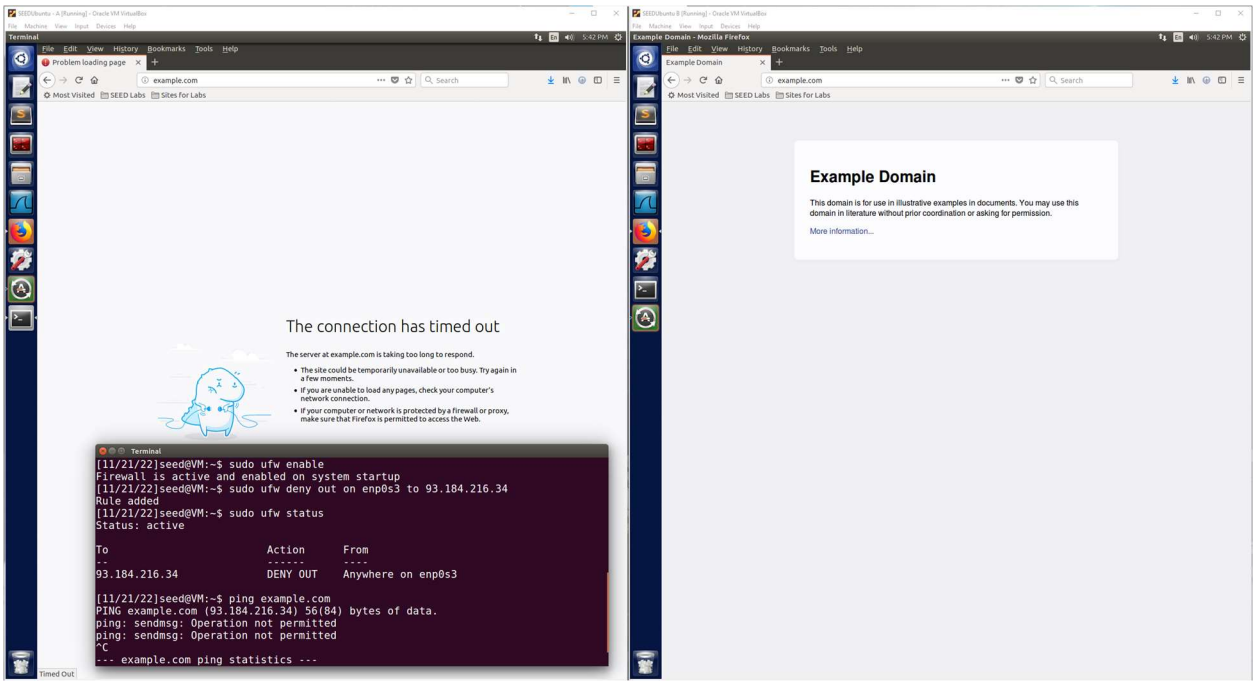
Lab 3

Task 2: Set up Firewall

Website that is blocked is example.com IP Address: 93.184.216.34

VM2(right) is able to access example.com with no issues(no firewall present)

VM1(left) is unable to access example.com with the firewall active, blocking any traffic to the target IP address as shown with the rule table of the firewall below.

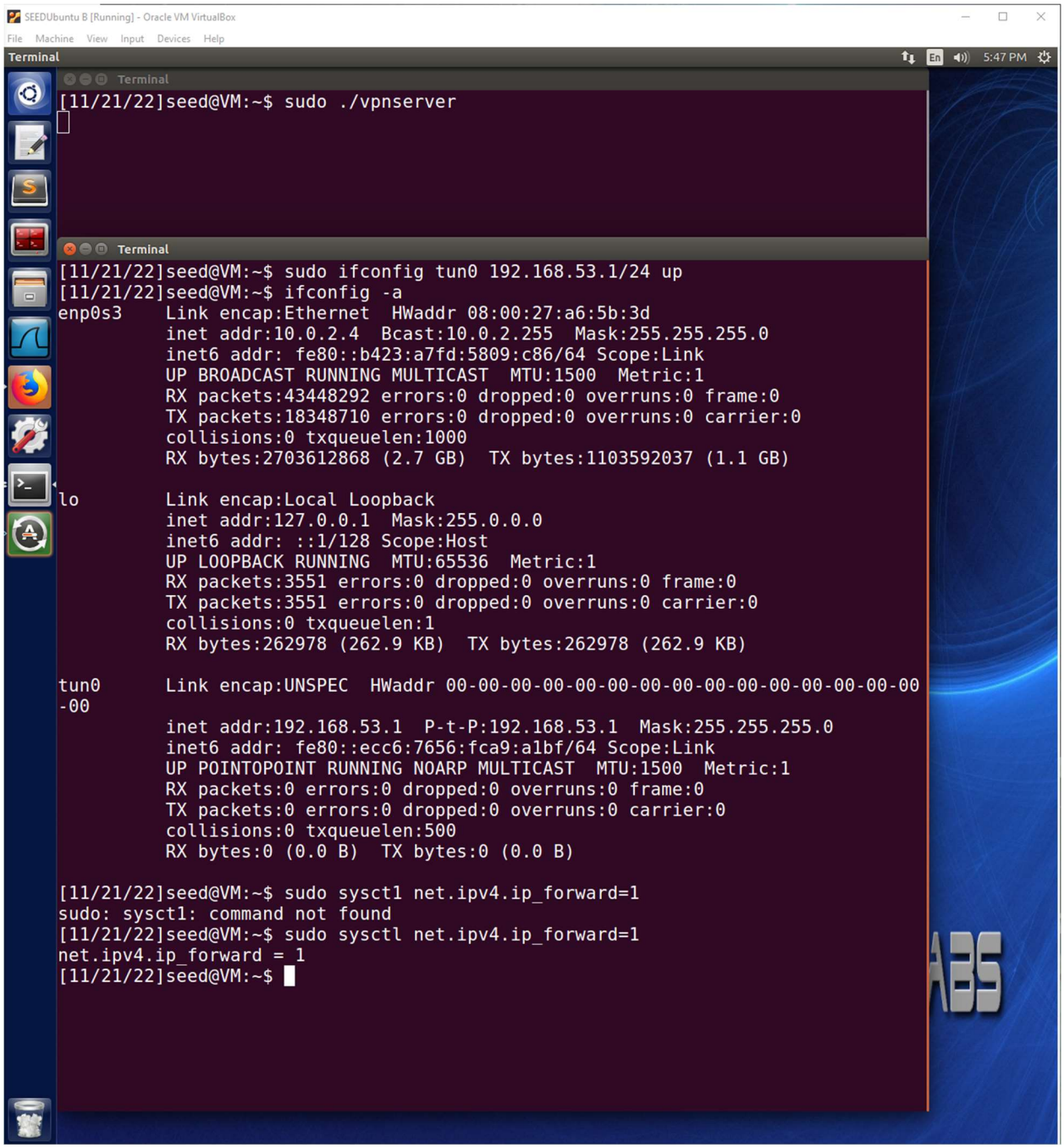


The firewall is set up to block any out from anywhere on enp0s3 to 93.184.216.34 which is the IP address of example.com. As shown in the above image for VM1 with the firewall active it is working, and the IP address is no longer reachable through Firefox and pings are blocked. For VM2, no firewall, the IP address is still reachable. This demonstrates that the firewall is working.

Task 3: Bypassing Firewall using VPN

Step 1: Run VPN Server

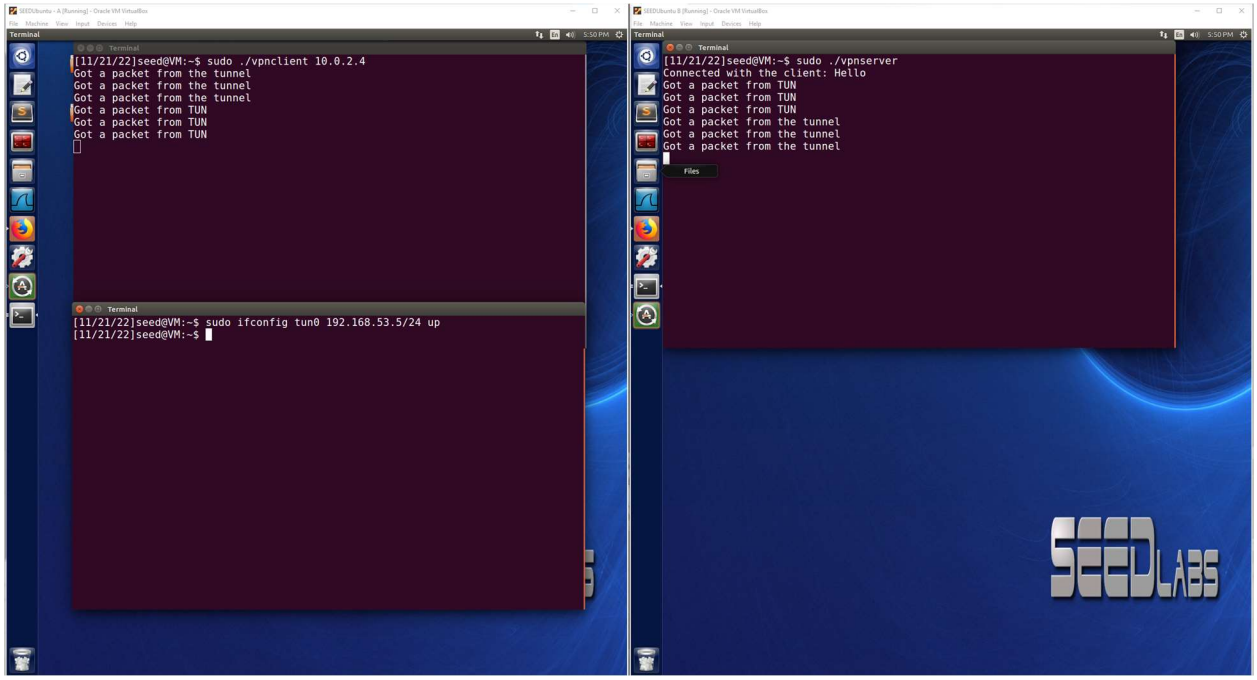
VM2: 10.0.2.4 – 198.168.53.1/24



Lab 3

Step 2: Run VPN Client

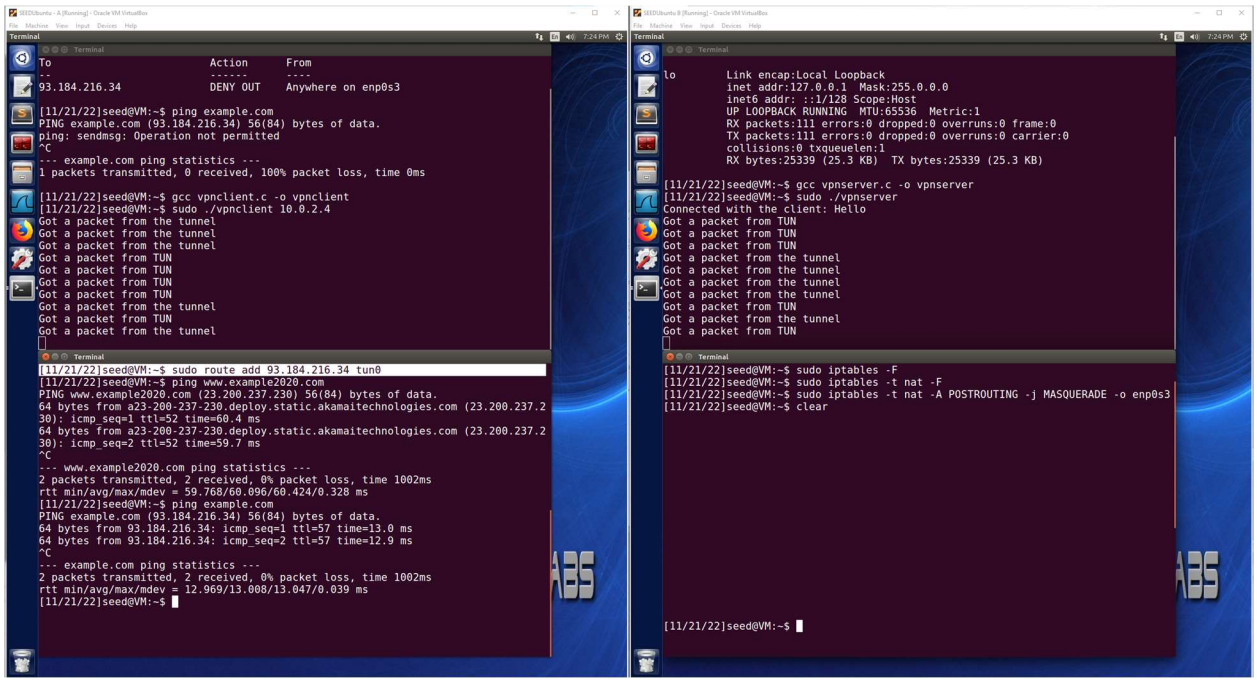
VM1: 10.0.2.15 – 192.168.53.5/24



Step 3: Set Up Routing on Client and Server VMs

All traffic coming from or going to 93.184.216.34(example.com) needs to be routed to the VPN tunnel connection as shown below.

sudo route add 93.184.216.34 tun0



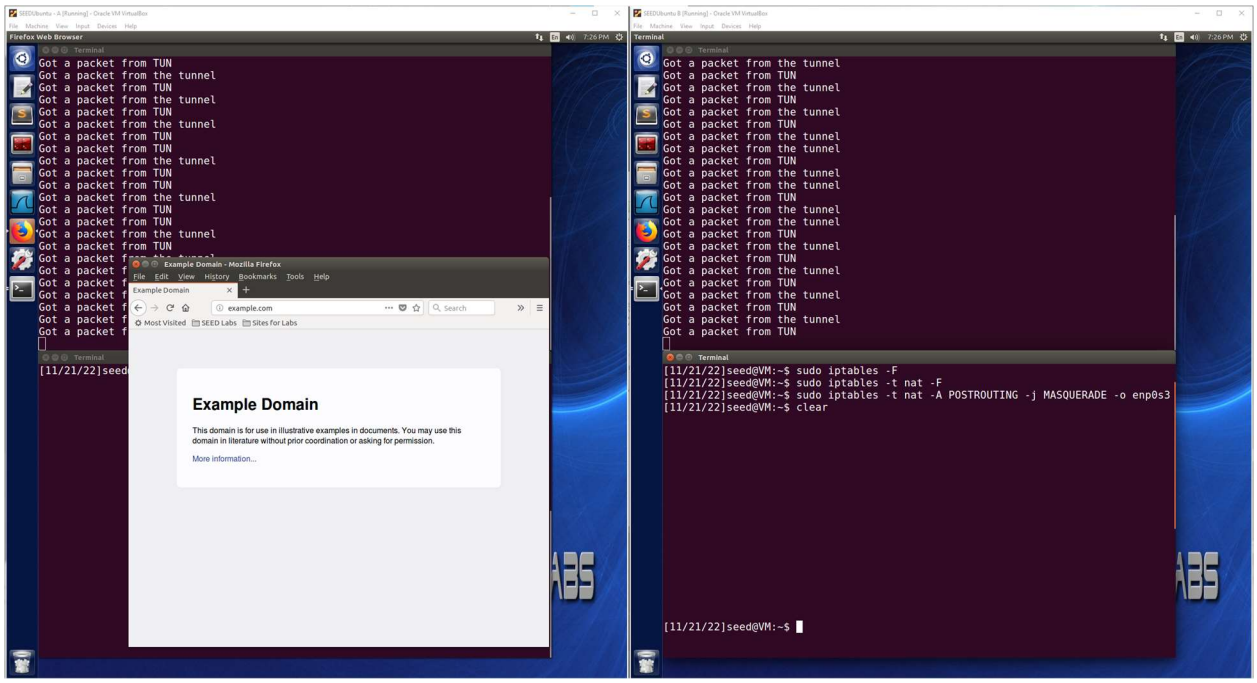
Step 4: Set Up NAT on Server VM

VM1: 10.0.2.15 – Pictured on the Left – Client VPN

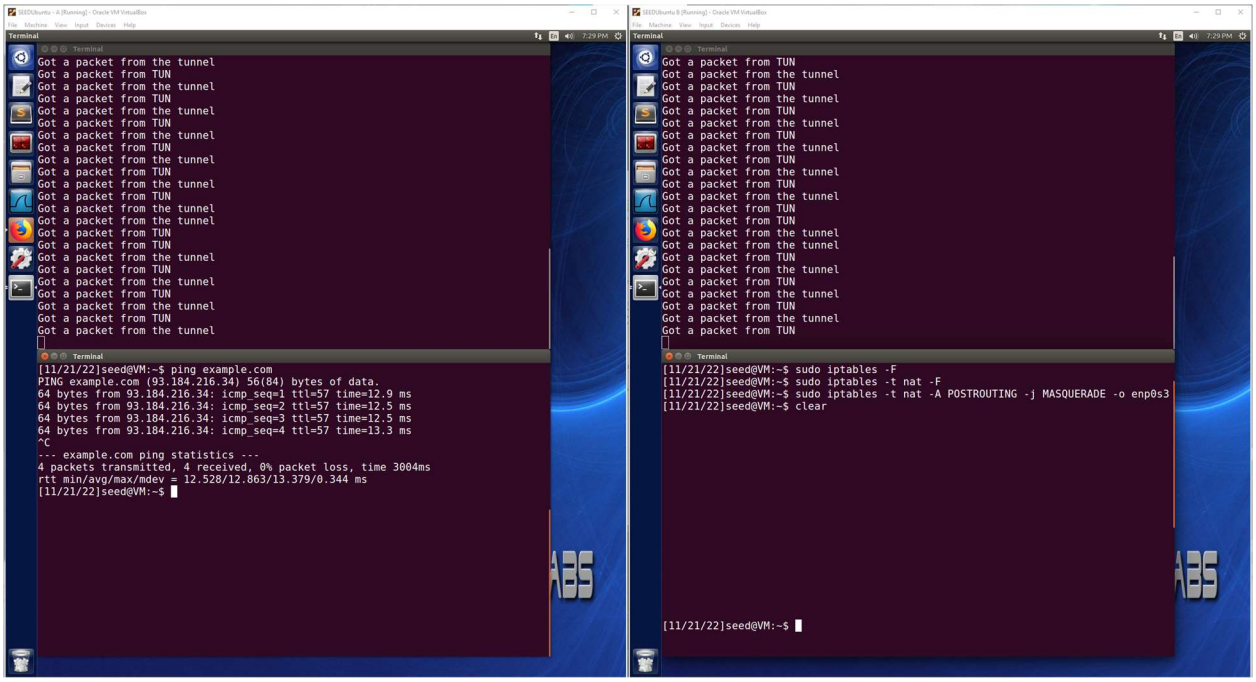
VM2: 10.0.2.4 – Pictured on the Right – Server VPN

The packet will be sent to the VPN server first from the destination because the destination is responding to who it saw on its end that the destination received the packet from which would be the VPN server. The VPN server would then receive the response from the destination and forward that response through the VPN tunnel to the VPN Client(VM1) who originated the packet request to the destination and is trying to bypass the firewall using the VPN Server(VM2).

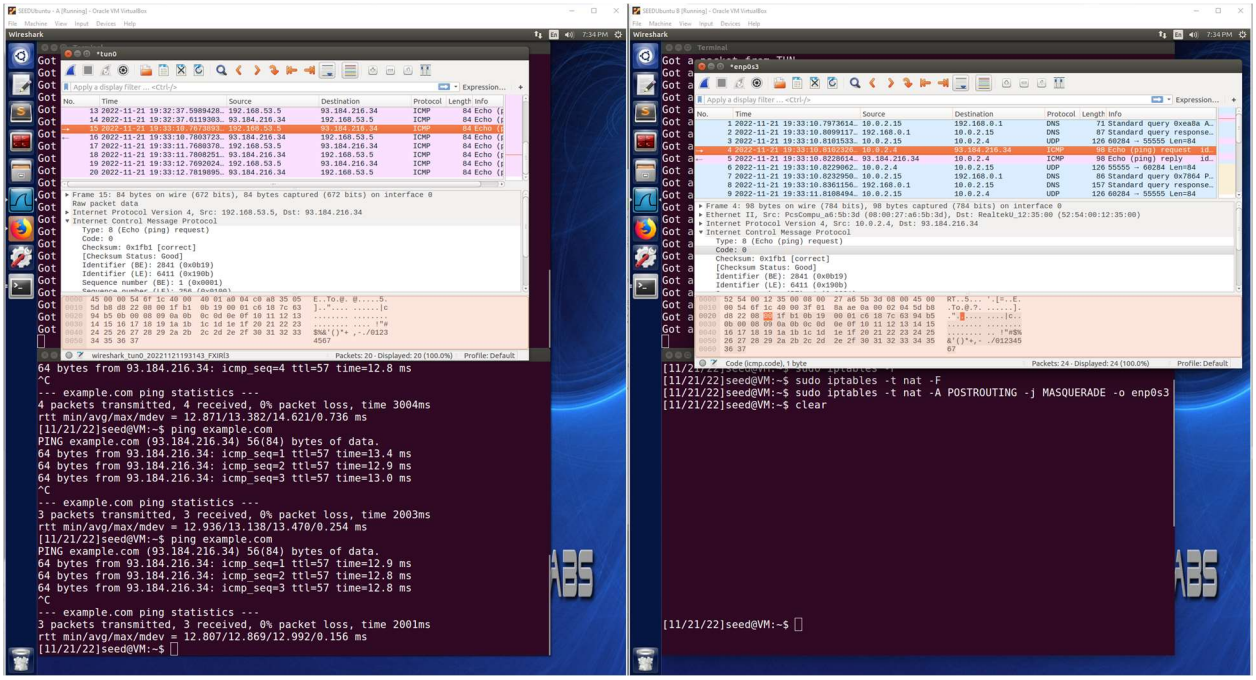
Firewall is successfully bypassed on the Client VM via the VPN, below will be the provided evidence that it has been successfully bypassed.



Now the Client VM is able to use Firefox to navigate to example.com(93.184.216.34) and the correct page is now displayed as the Client VM's connection is being tunneled through the Server VM VPN and is able to bypass the firewall using this method to access the blocked web page.



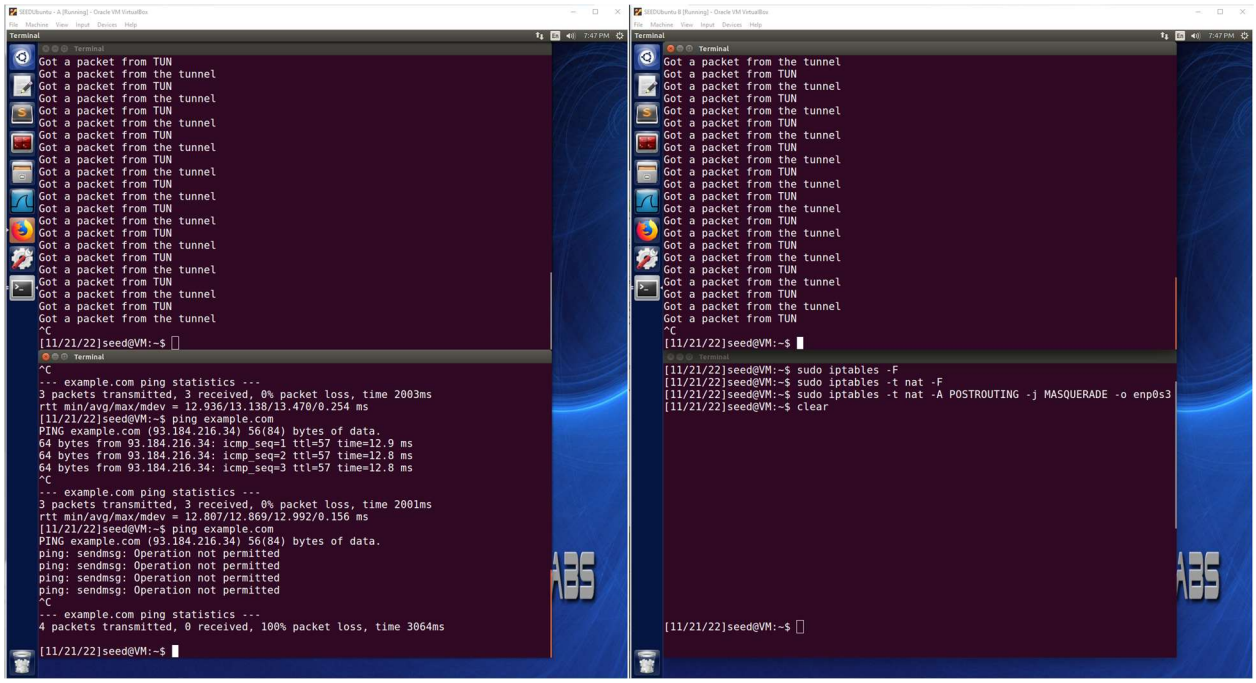
Another piece of evidence that the firewall has been bypassed is that previous the ping command was blocked by the firewall when trying to ping example.com. Now with the VPN tunnel established, the Client VM as seen in the image above can successfully ping example.com and receive a response. This is because the packet was routed through the VPN tunnel then to the destination, and the response was sent to the Server VM, but then sent back to the Client VM through the VPN tunnel. This can also be observed in Wireshark as more evidence that the firewall was now bypassed.



In Wireshark the ping ICMP packet can be observed on the Client VM being sent out to the destination and then the response being received from the destination. The source is the IP address assigned with

the VPN(192.168.53.5) instead of the Client VM's original IP address(10.0.2.15) which is evidence that the VPN is working and bypassing the firewall. On the Server VM, we can also see on Wireshark when the packet is forwarded to the destination, the Server VM's IP address is 10.0.2.4 as the source and then sending the ping request to the destination which is forwarding the packet from the Client VM. When the response is received it is then forwarded back through the VPN tunnel to the Client VM. This is what allows the Client VM to bypass the firewall to ping and navigate to the blocked website.

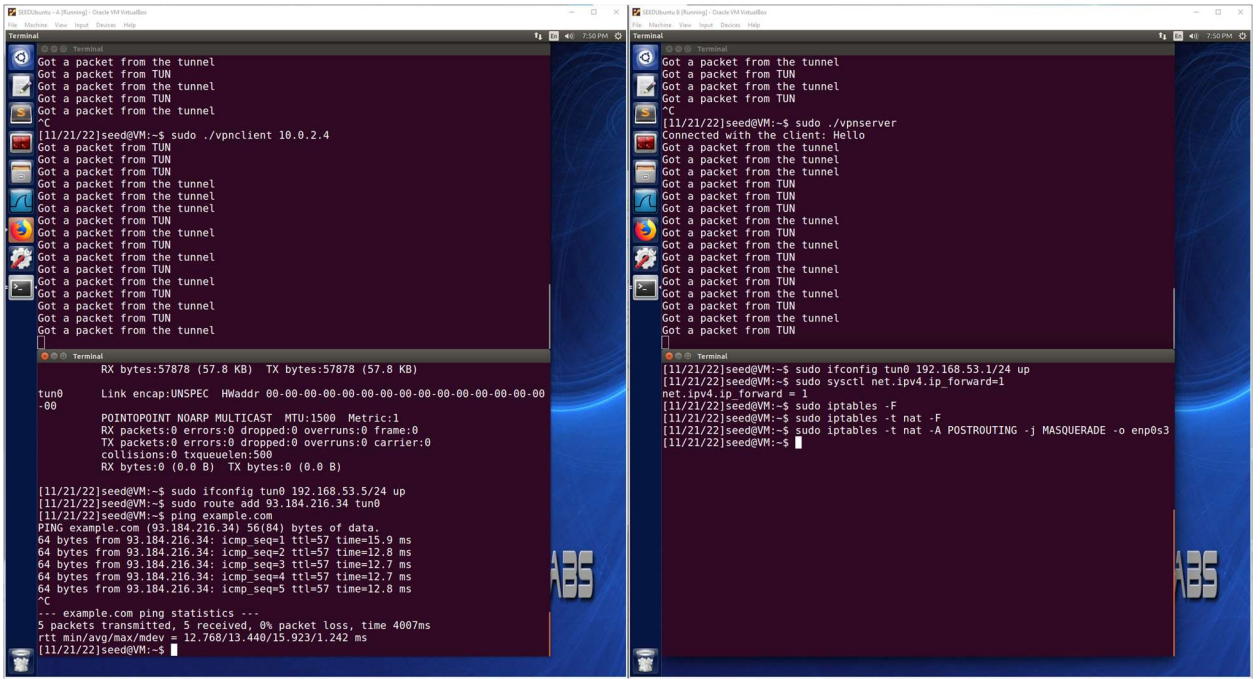
As a final piece of evidence that the firewall was functioning properly for the tests by blocking the intended website and that it was being bypassed using the VPN connection. As seen below when the VPN tunneling is broken, the Client VM is no longer able to ping the website as it is blocked by the firewall.



When the VPN tunnel is established and set up properly, the Client VM is once again able to ping the blocked website by using the VPN tunnel to bypass the firewall as seen below.

This is evidence that the Client VM is able to bypass the firewall to reach the blocked website via the VPN tunnel in different ways such as through a web browser or through a ping command as seen above.

Lab 3


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
[11/21/22]seed@VM:~$ sudo ./vpnclient 10.0.2.4
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
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
RX bytes:57878 (57.8 KB) TX bytes:57878 (57.8 KB)
tun0 Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
POINTOPOINT 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)
[11/21/22]seed@VM:~$ sudo ifconfig tun0 192.168.53.5/24 up
[11/21/22]seed@VM:~$ sudo route add 93.184.216.34 tun0
[11/21/22]seed@VM:~$ ping example.com
PING example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34: icmp_seq=1 ttl=57 time=15.9 ms
64 bytes from 93.184.216.34: icmp_seq=2 ttl=57 time=12.8 ms
64 bytes from 93.184.216.34: icmp_seq=3 ttl=57 time=12.7 ms
64 bytes from 93.184.216.34: icmp_seq=4 ttl=57 time=12.7 ms
64 bytes from 93.184.216.34: icmp_seq=5 ttl=57 time=12.8 ms
^C
--- example.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 12.768/13.440/15.923/1.242 ms
[11/21/22]seed@VM:~$
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
^C
[11/21/22]seed@VM:~$ sudo ./vpnserver
Connected with the client: Hello
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 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
[11/21/22]seed@VM:~$ sudo ifconfig tun0 192.168.53.1/24 up
[11/21/22]seed@VM:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[11/21/22]seed@VM:~$ sudo iptables -F
[11/21/22]seed@VM:~$ sudo iptables -t nat -F
[11/21/22]seed@VM:~$ sudo iptables -t nat -A POSTROUTING -j MASQUERADE -o enp0s3
[11/21/22]seed@VM:~$