CNS LAB

LAB - 4

Linux Firewall Exploration Lab

NAME : SIRI S

SEMESTER : 5

SECTION :H

SRN : PES1UG19CS485

Lab Setup:

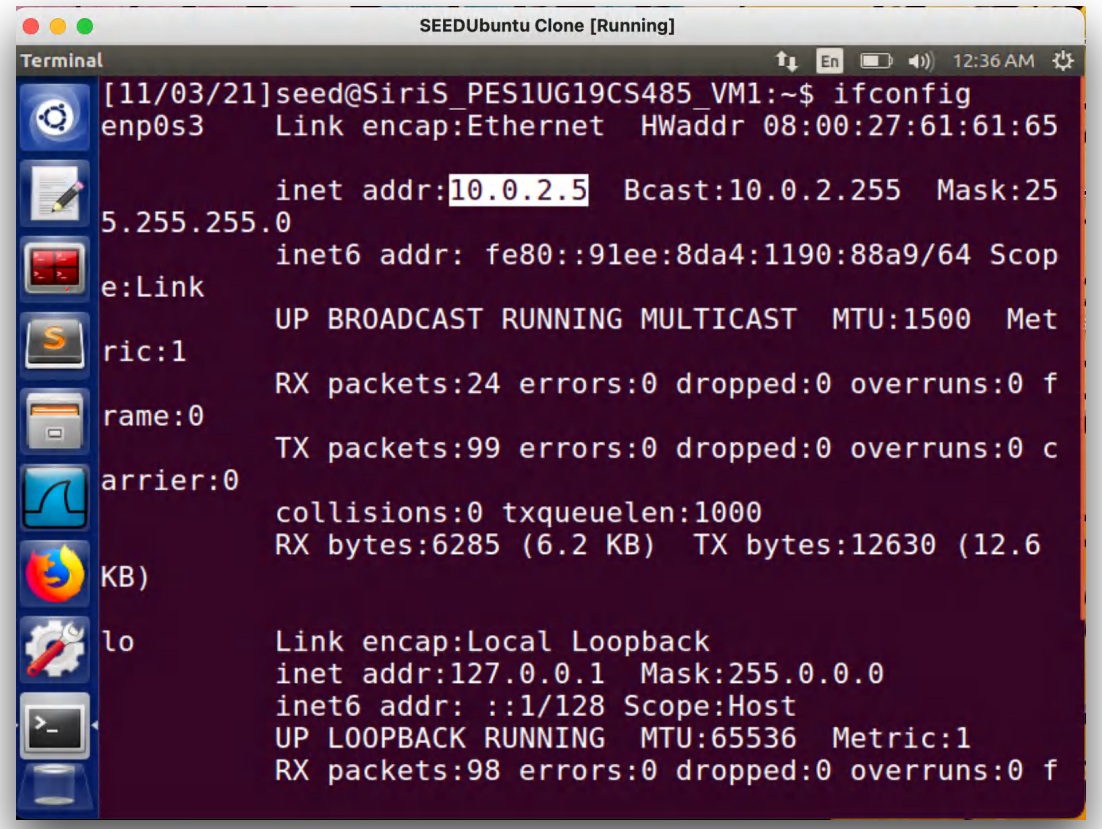
VM 2: 10. 0. 2. 4

VM 1: 10. 0. 2. 5

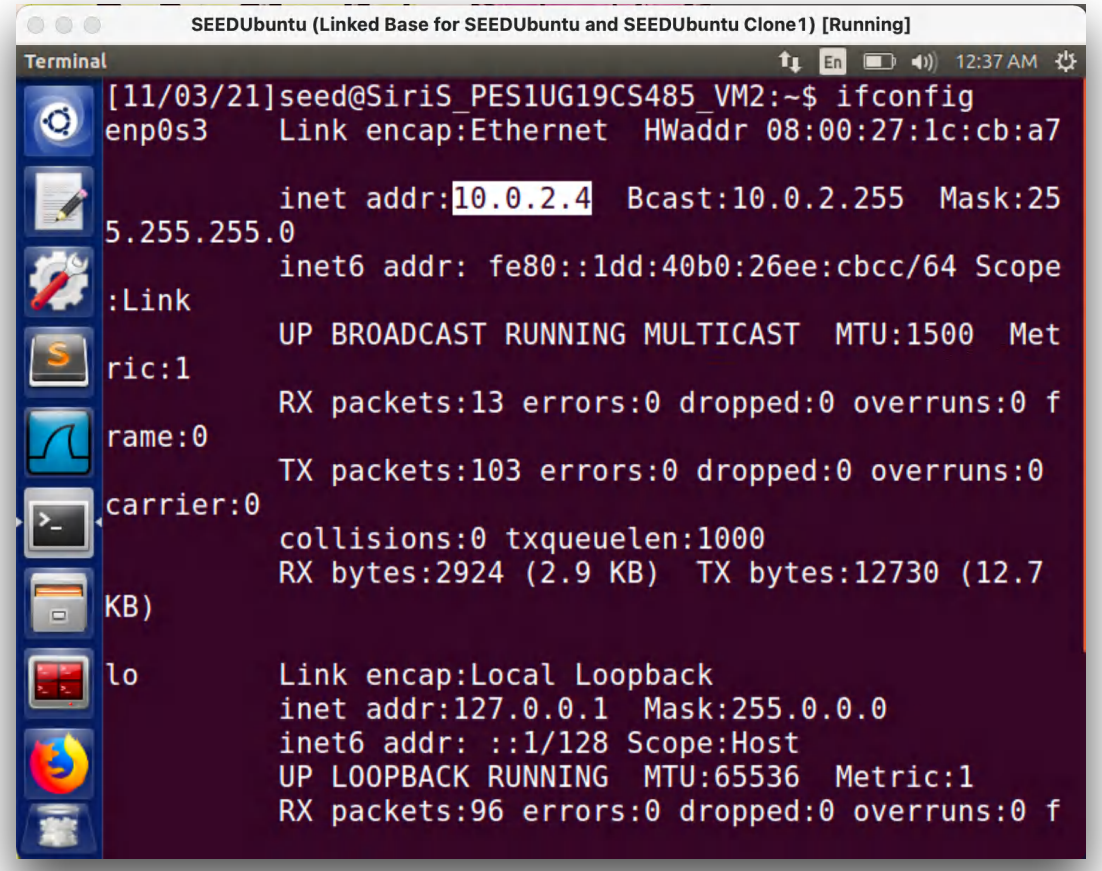
VM 3: 10. 0. 2. 6

IP Addresses

VM 1:

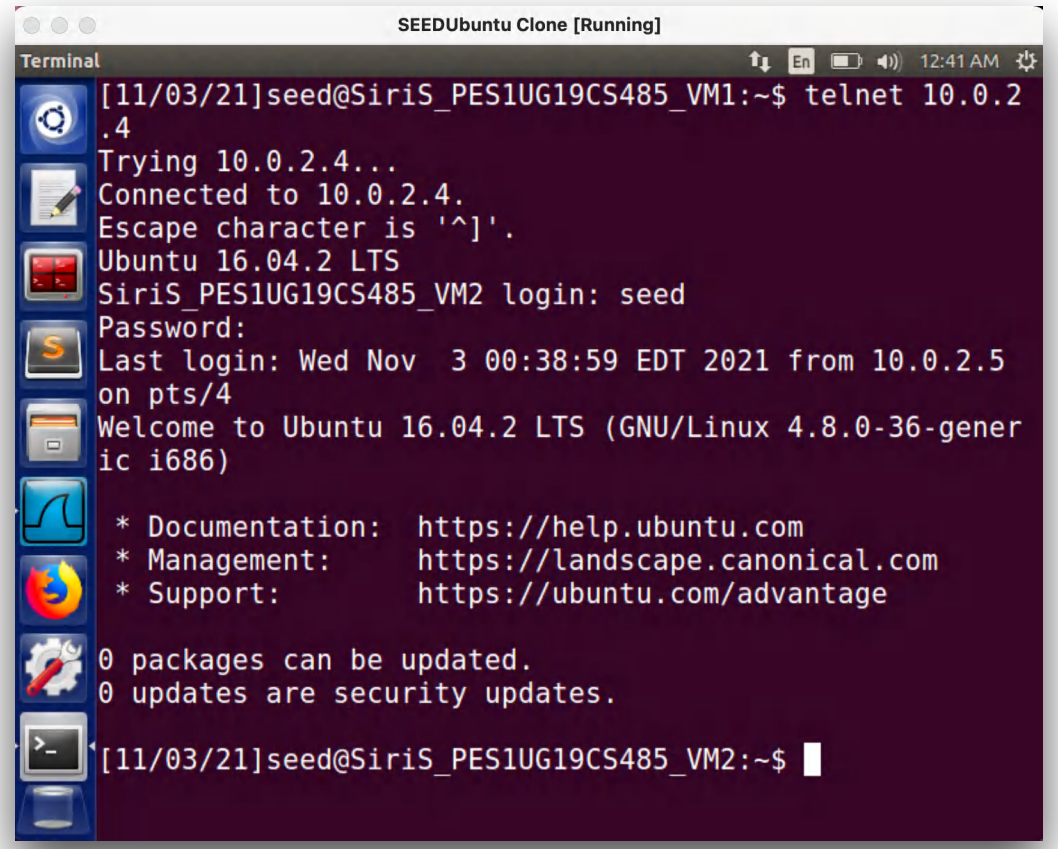


VM 2:

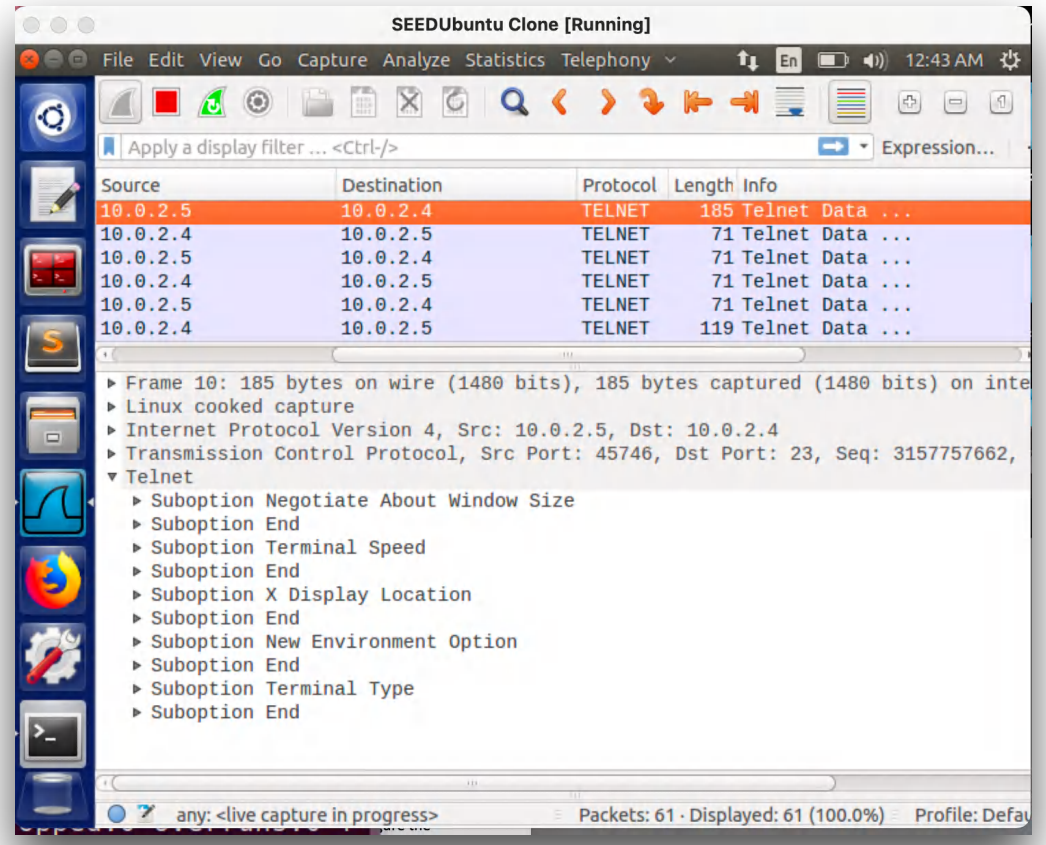


Task 1: Using Firewall

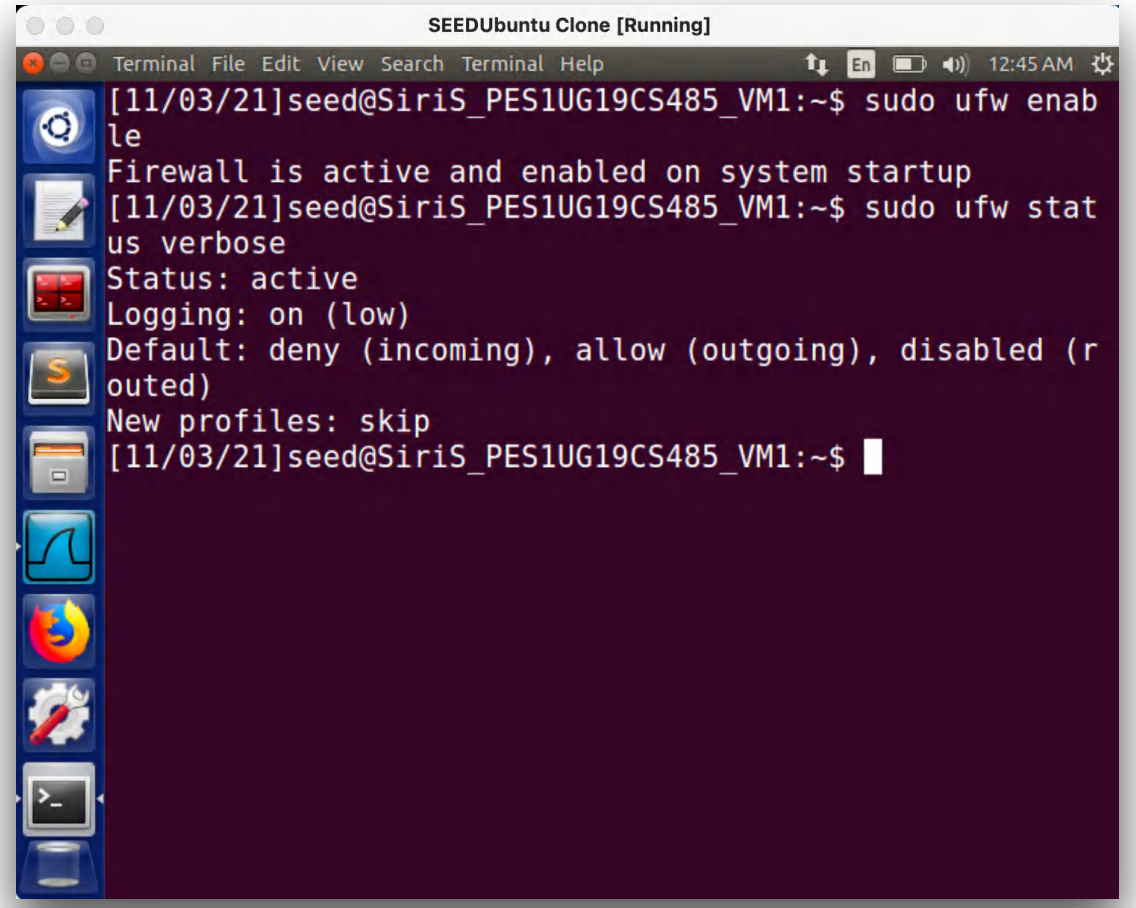
To show that we are able to establish a telnet connection from VM 1 to VM 2:



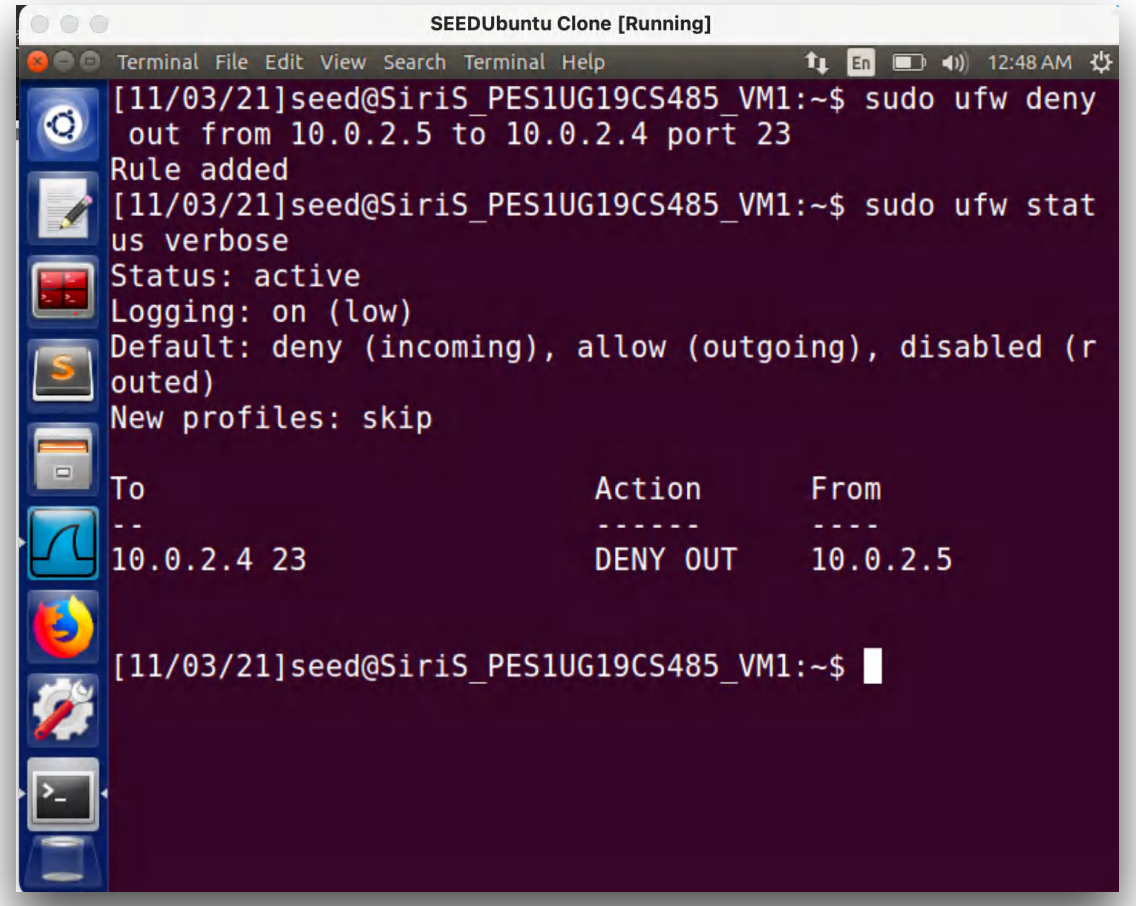
Wireshark screenshot:



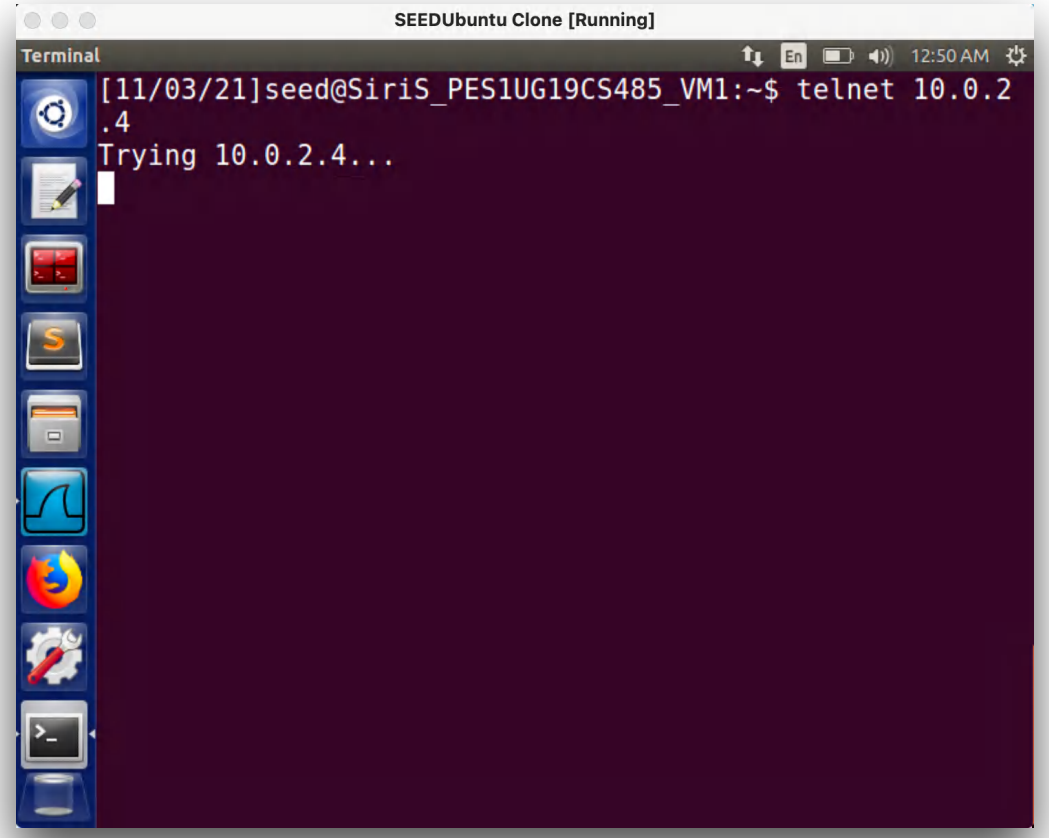
Configuring a firewall using ***ufw*** to prevent VM1 from being able to telnet to VM2:



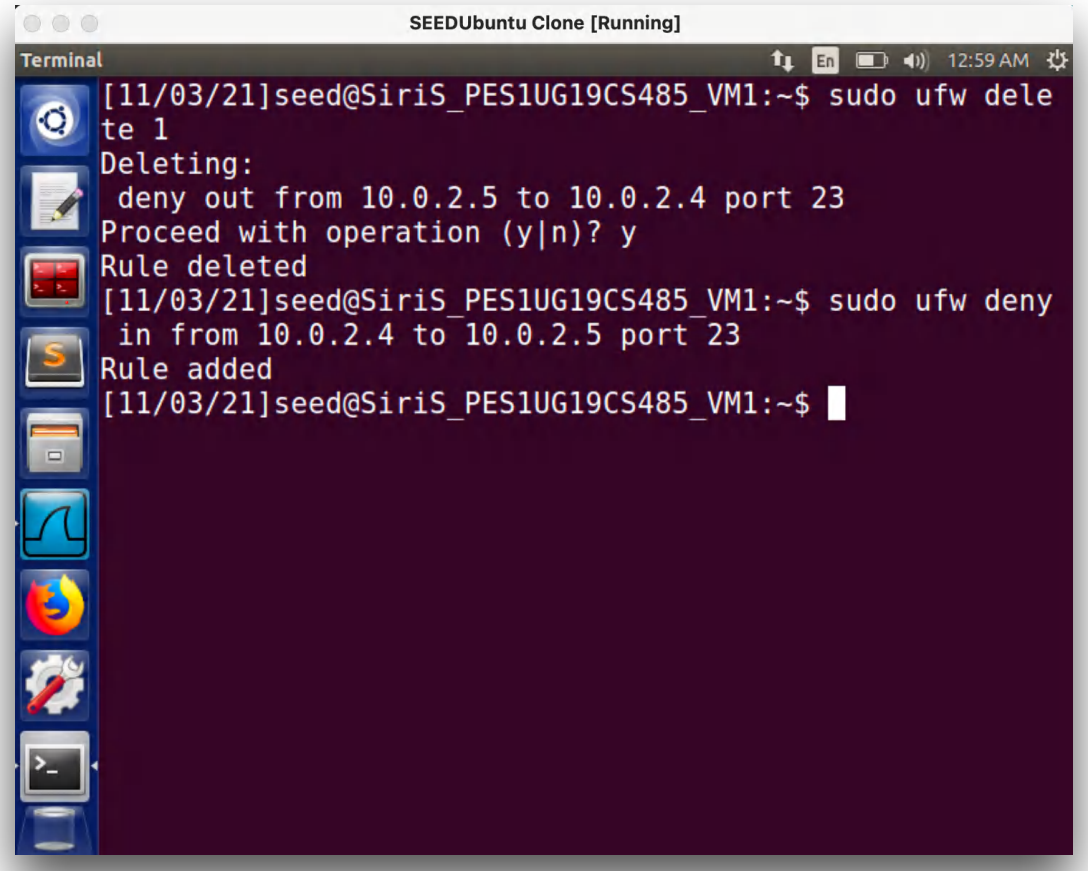
Configuring a firewall on VM1 to deny telnet (port 23) to VM2:



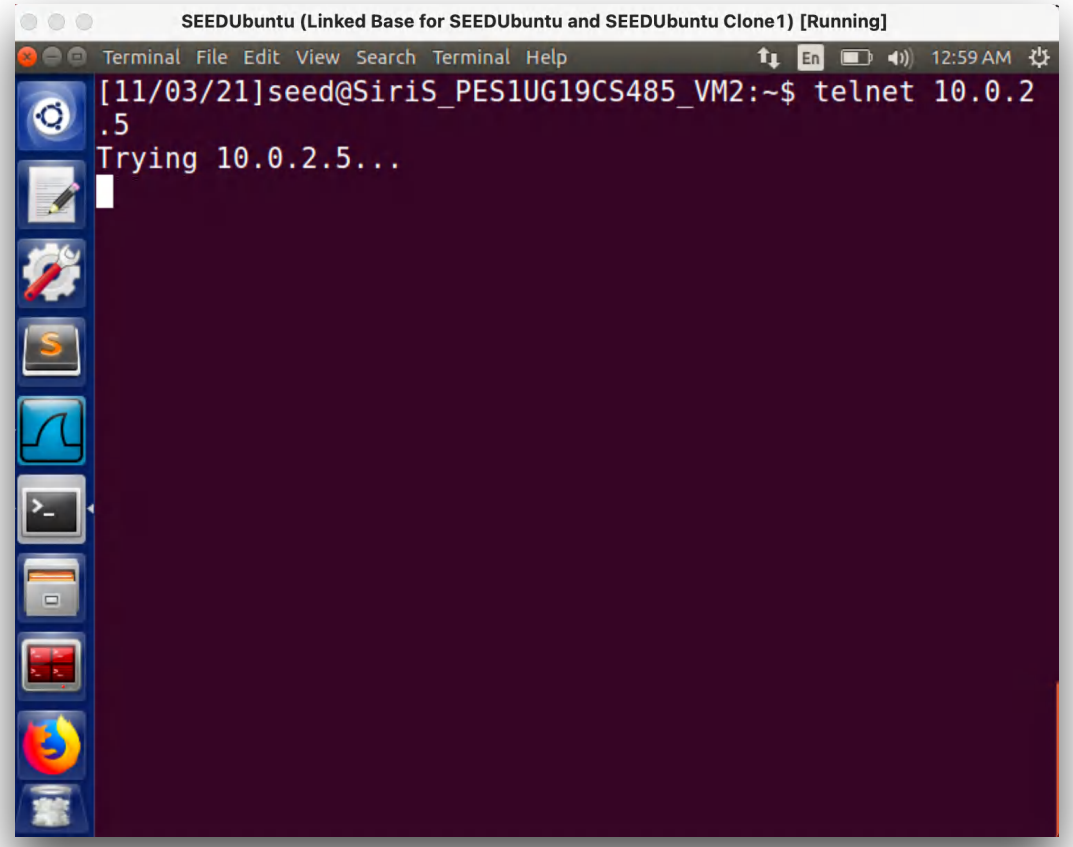
To show that because of the firewall that's enabled, telnet is denied from VM1 to VM2:



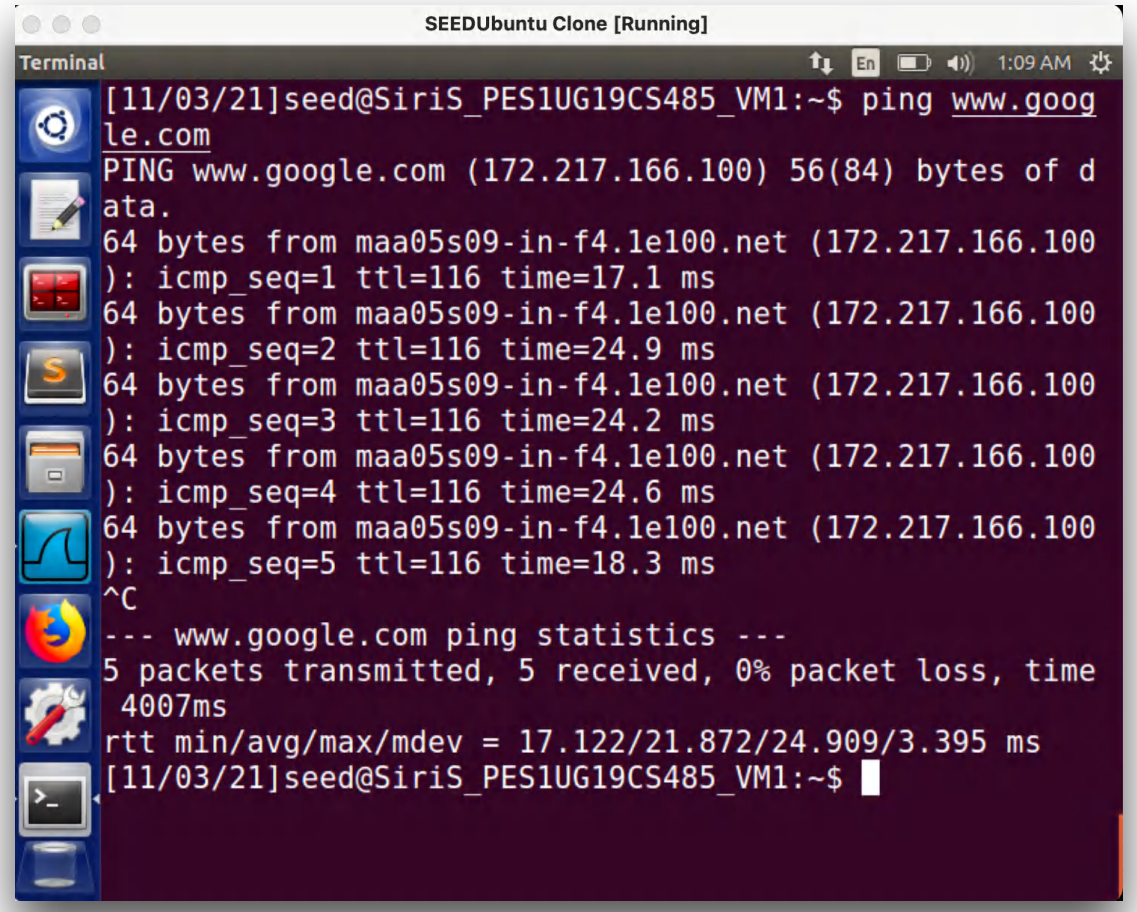
We next delete the firewall rule in VM1. We add a rule to prevent VM2 from being able to do telnet to VM1:



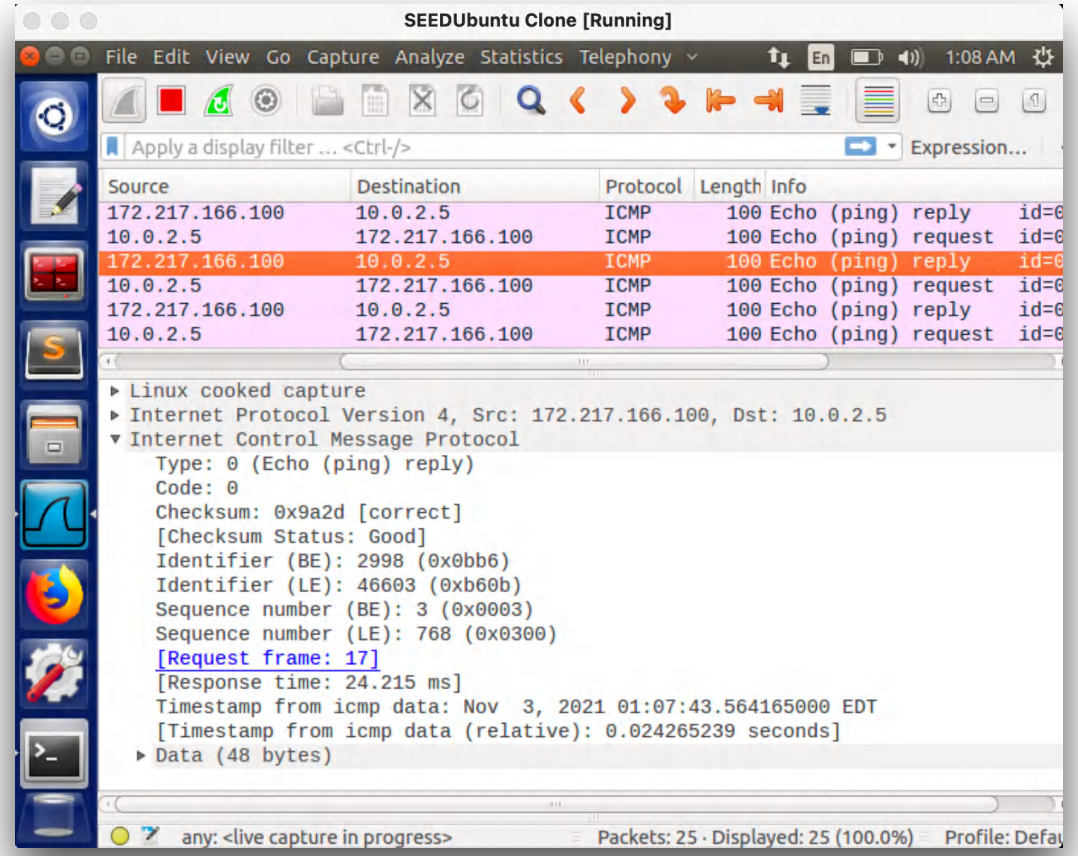
Now, telnet is blocked from VM2 to VM1:



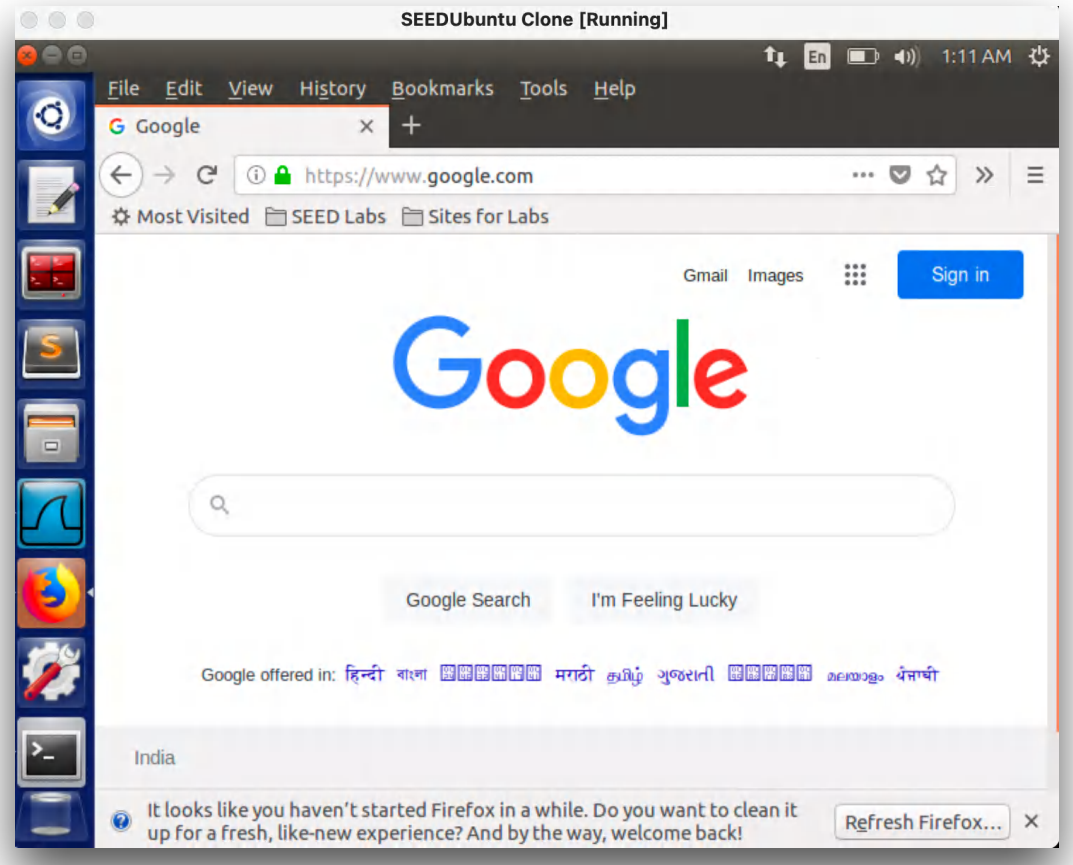
We next need to block VM1 from visiting a website. ‘www.google.com’:



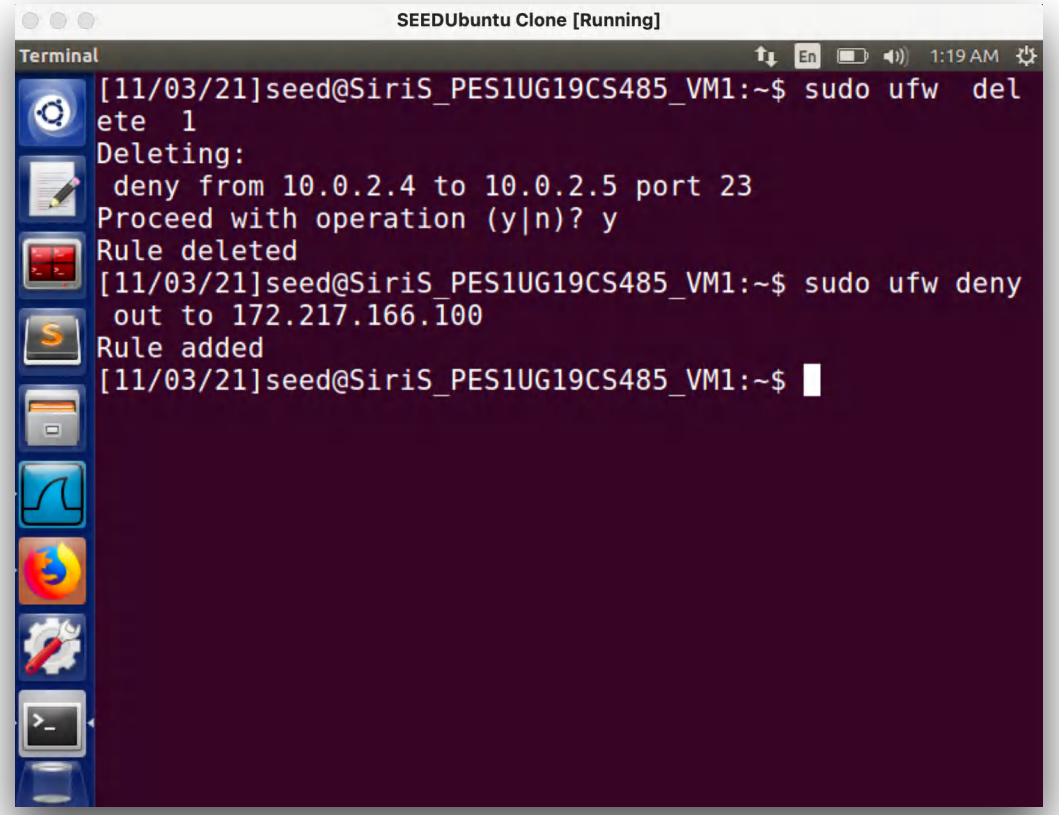
Wireshark screenshot:



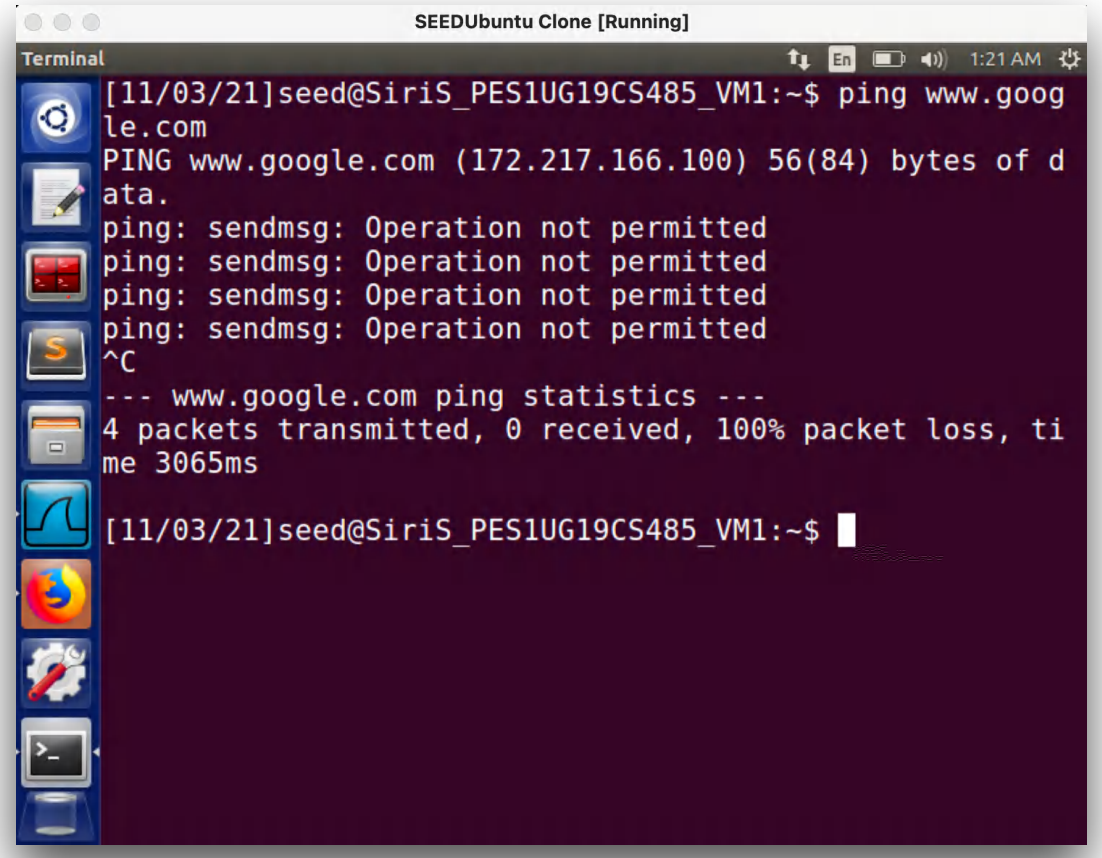
Testing whether the website is accessible to the browser:



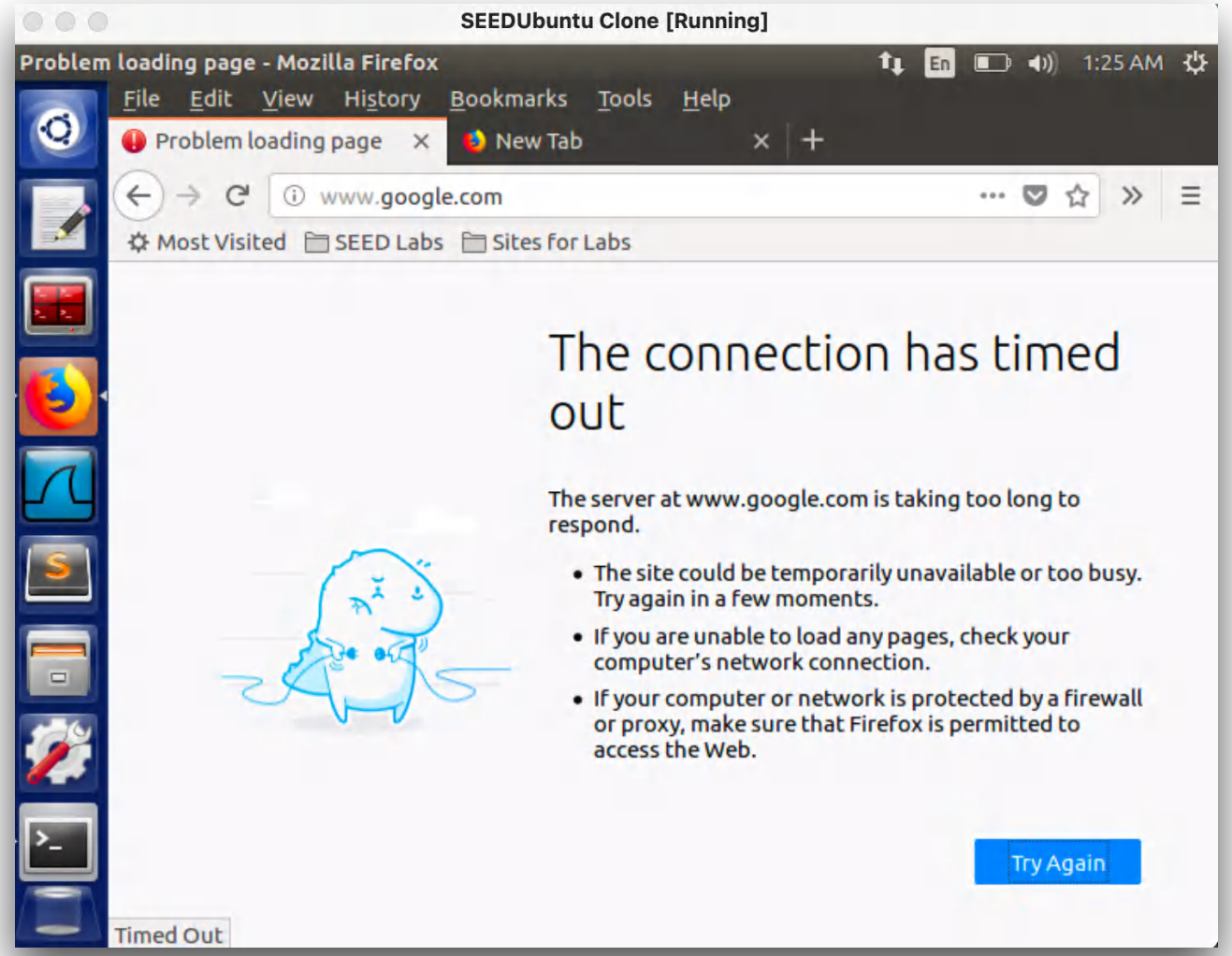
We then clear the browser cache by deleting Cache from the browser history. We next add the firewall rule to prevent VM1 from accessing the IP address for **www.google.com** (172.217.166.100)



To show that firewall has blocked www.google.com:



This can be verified by revisiting www.google.com in the Firefox browser. The page is not loaded.

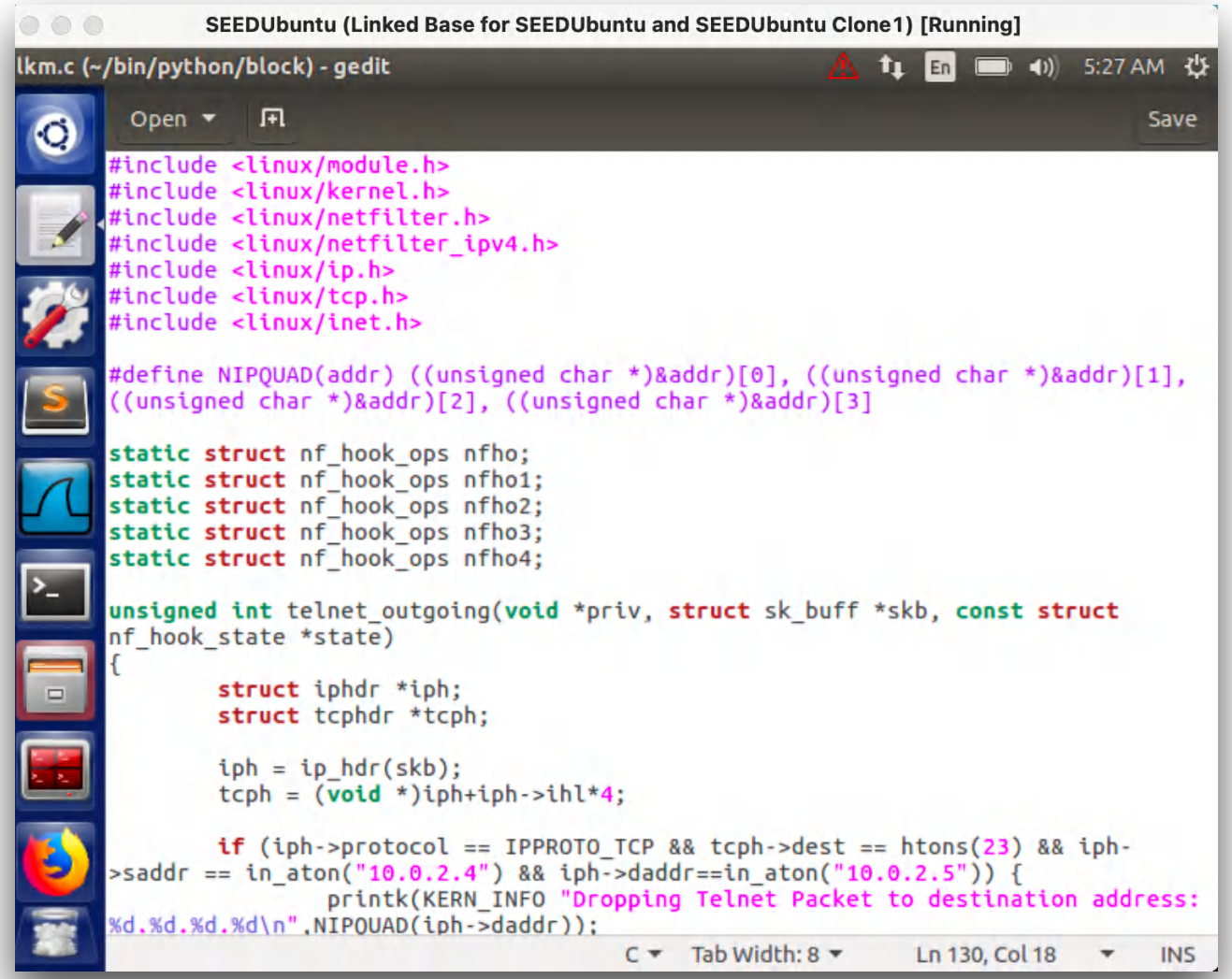


Task 2: How Firewall Works

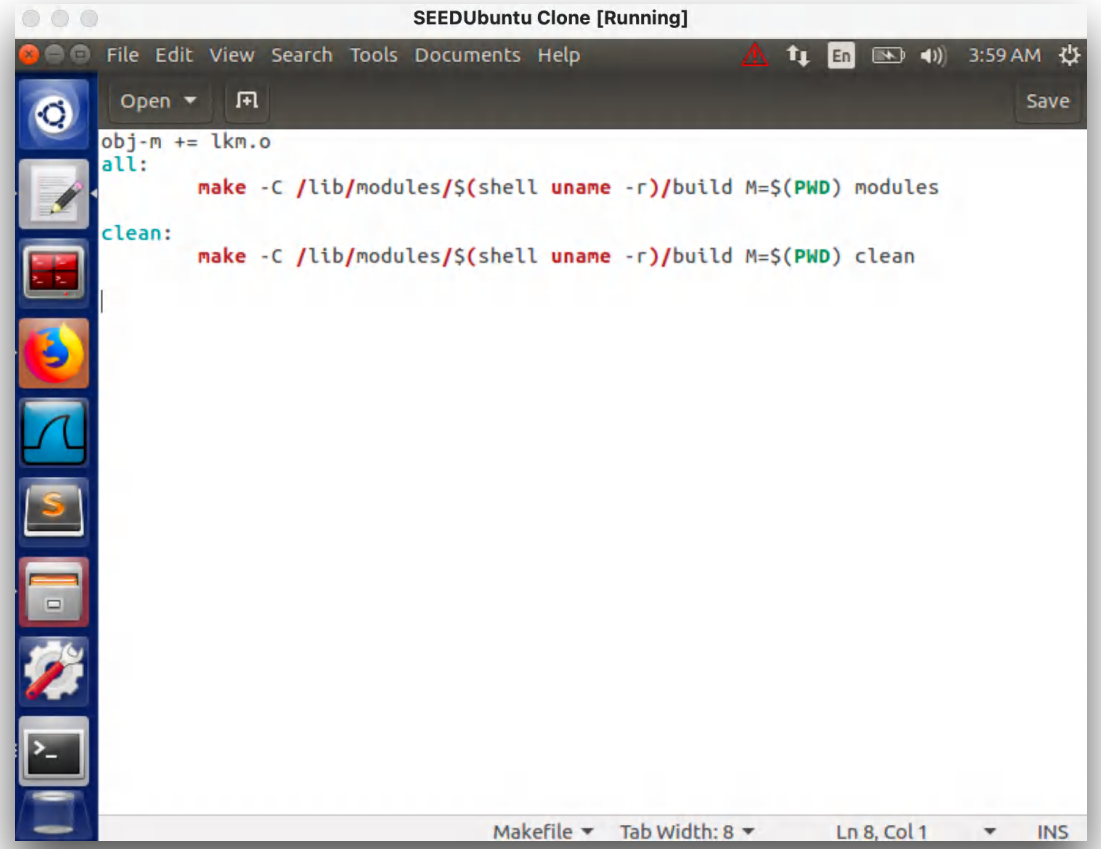
In this task, we will develop a firewall using netfilter and LKM. We implement five rules in this firewall:

- Block telnet from VM1 to VM2
- Block telnet from VM2 to VM1
- Block external website access from VM1
- Block ssh from VM1 to VM2
- Block ssh from VM2 to VM1

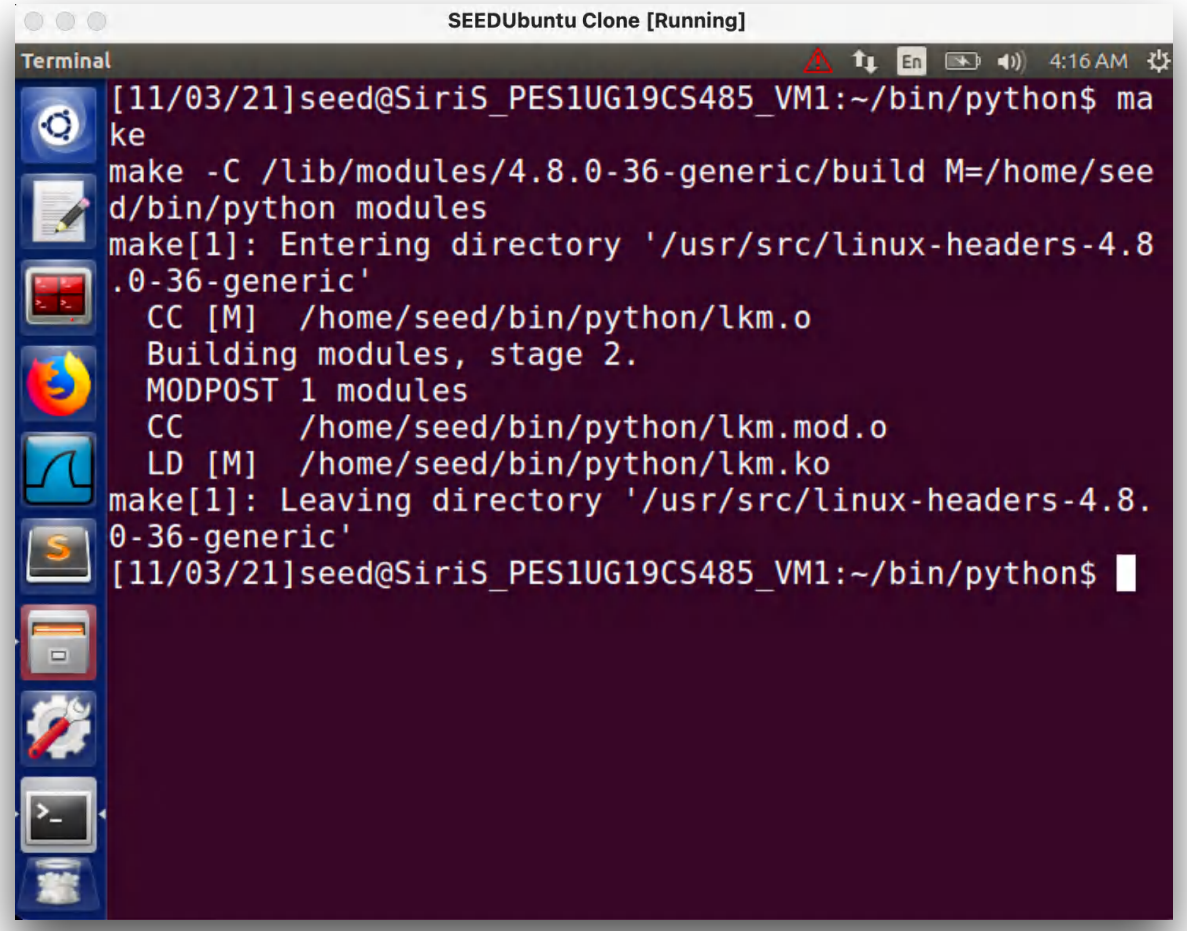
Code for the firewall:



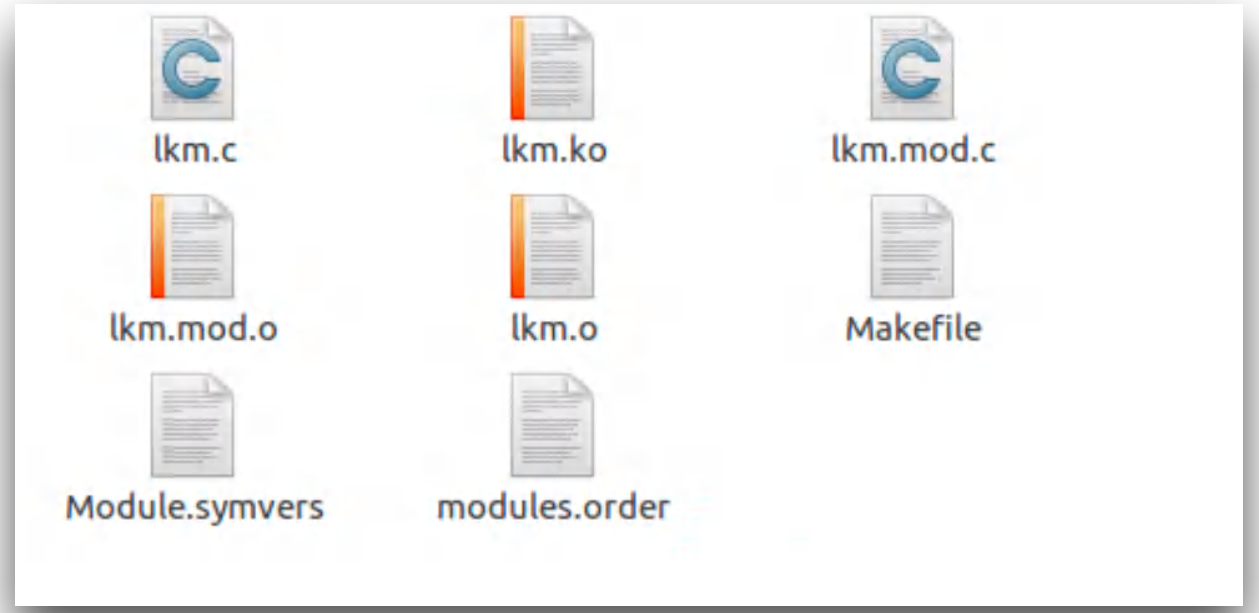
Makefile used:



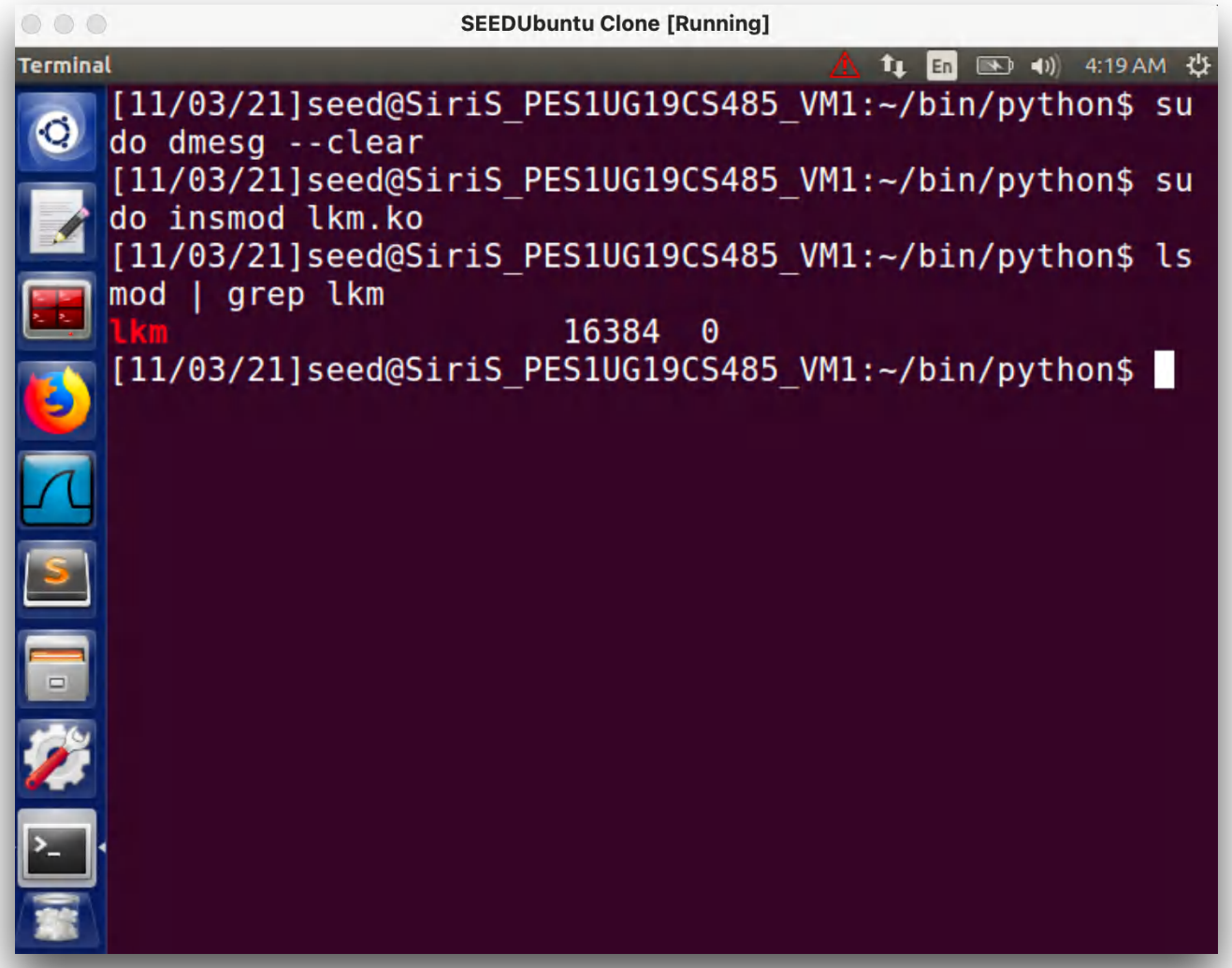
Upon running these two files:



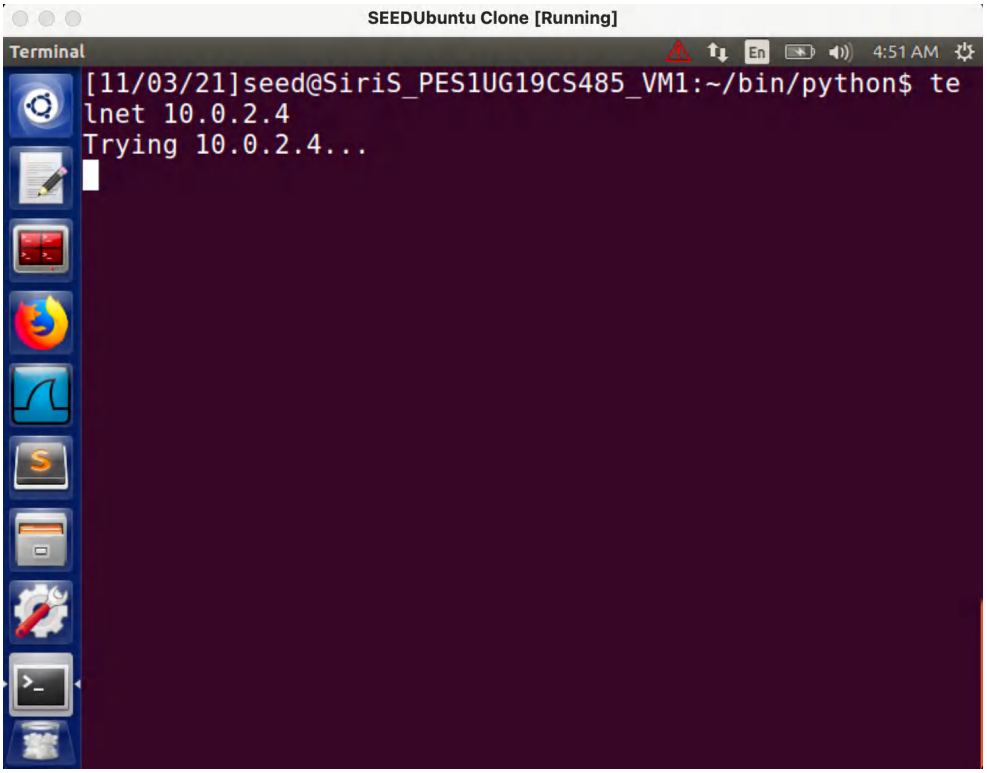
On running make command, all the different files get created and is stored in the same directory.



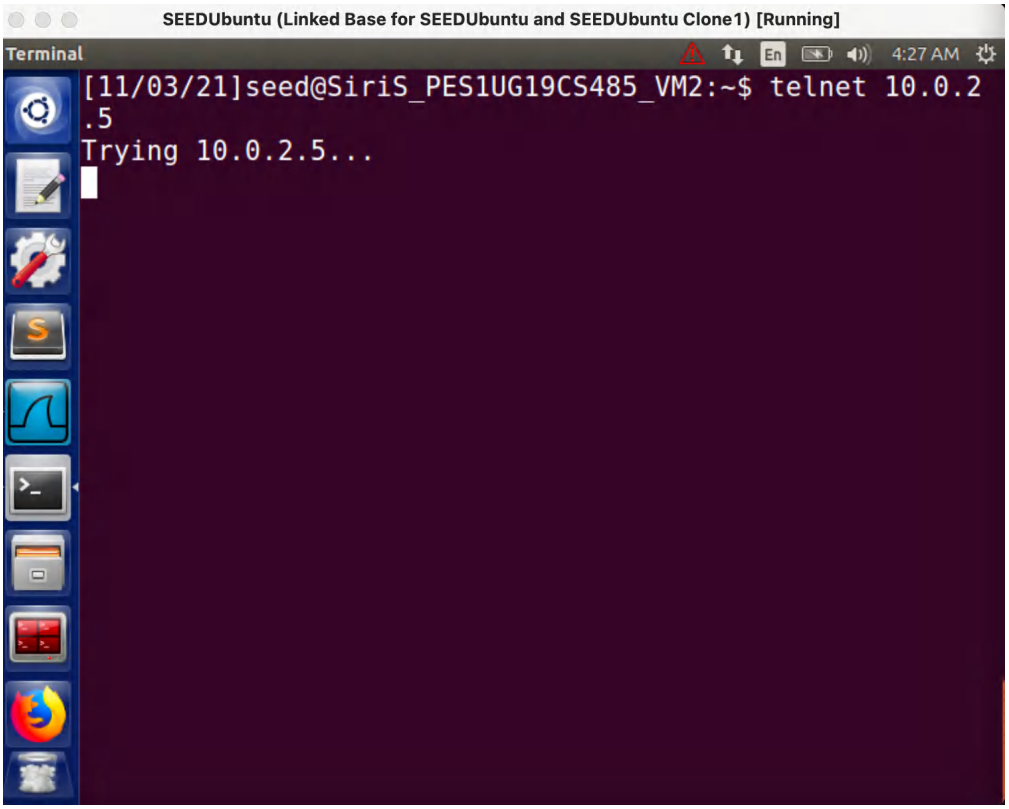
Now we need to insert the kernel module (task2.ko) using insmod :



Step 1: Testing whether telnet from VM1 to VM2 is blocked

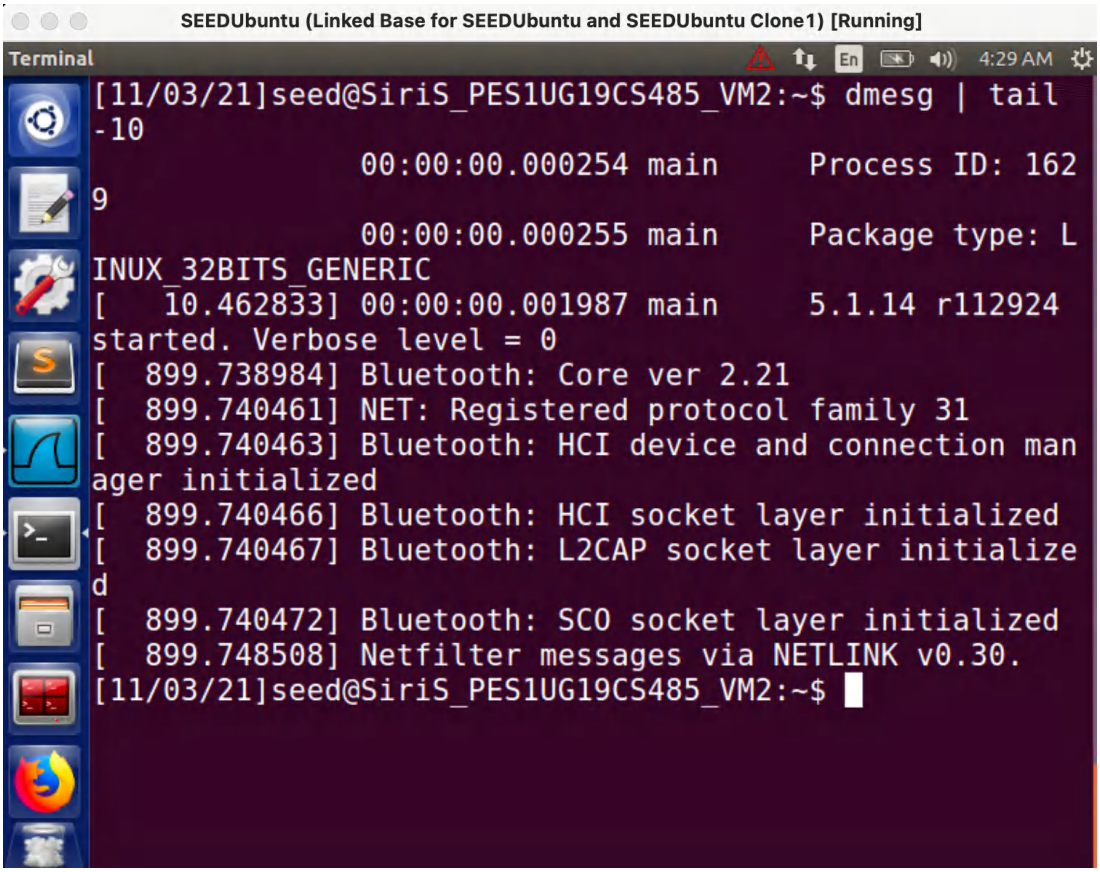


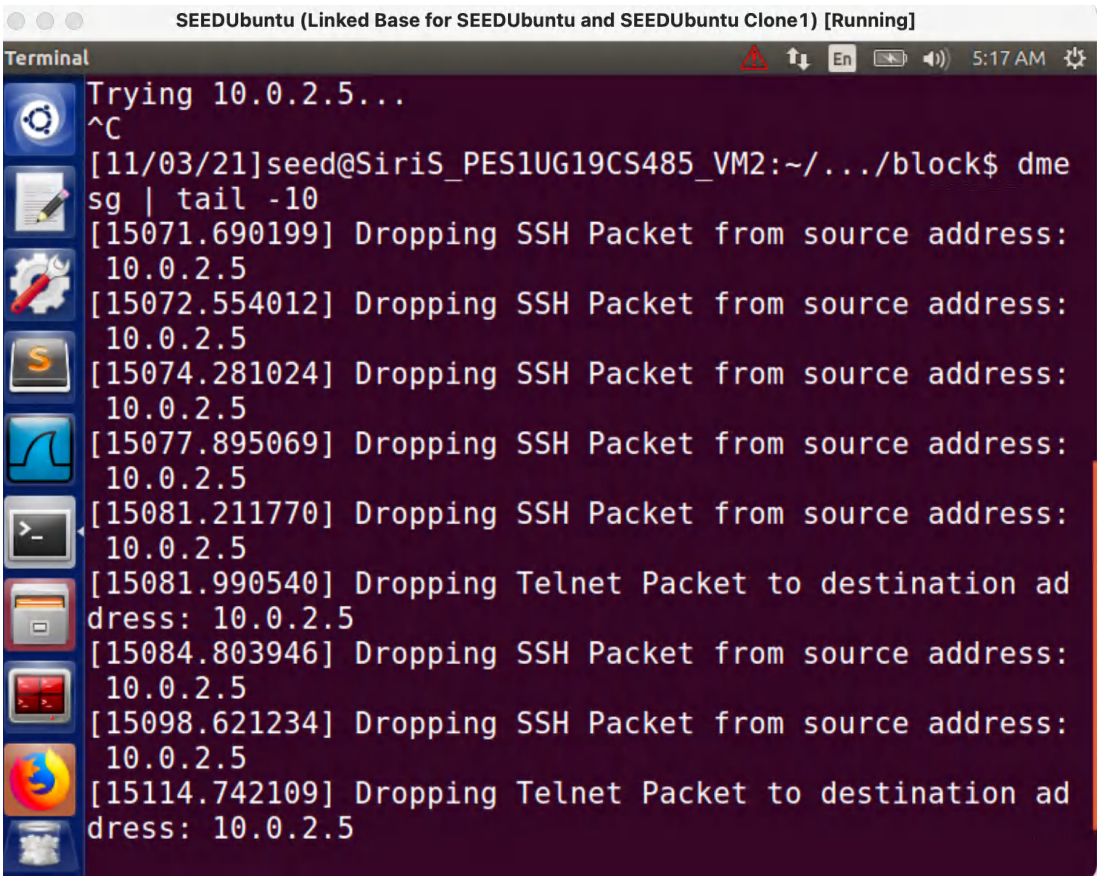
Step 2: Testing whether telnet from VM2 to VM1 is blocked



We can see that telnet is blocked from both ways.

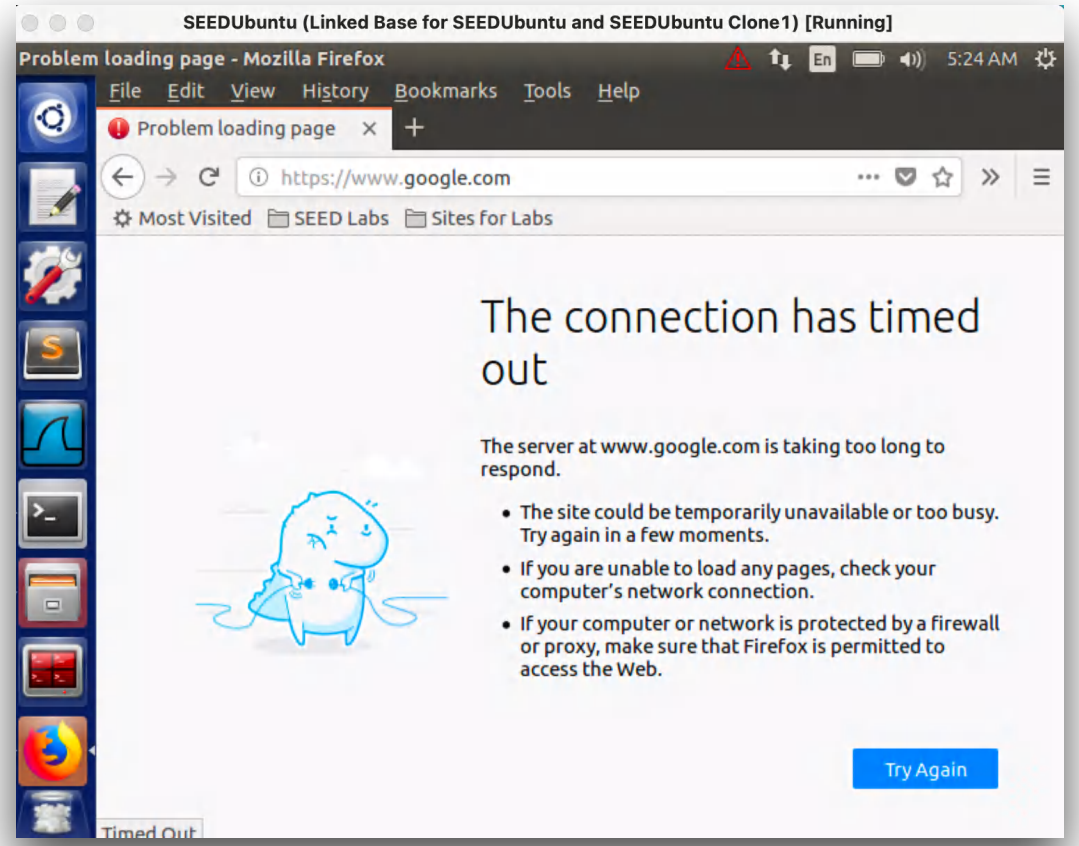
Command that is used to show the dropping of packets because of the firewall:



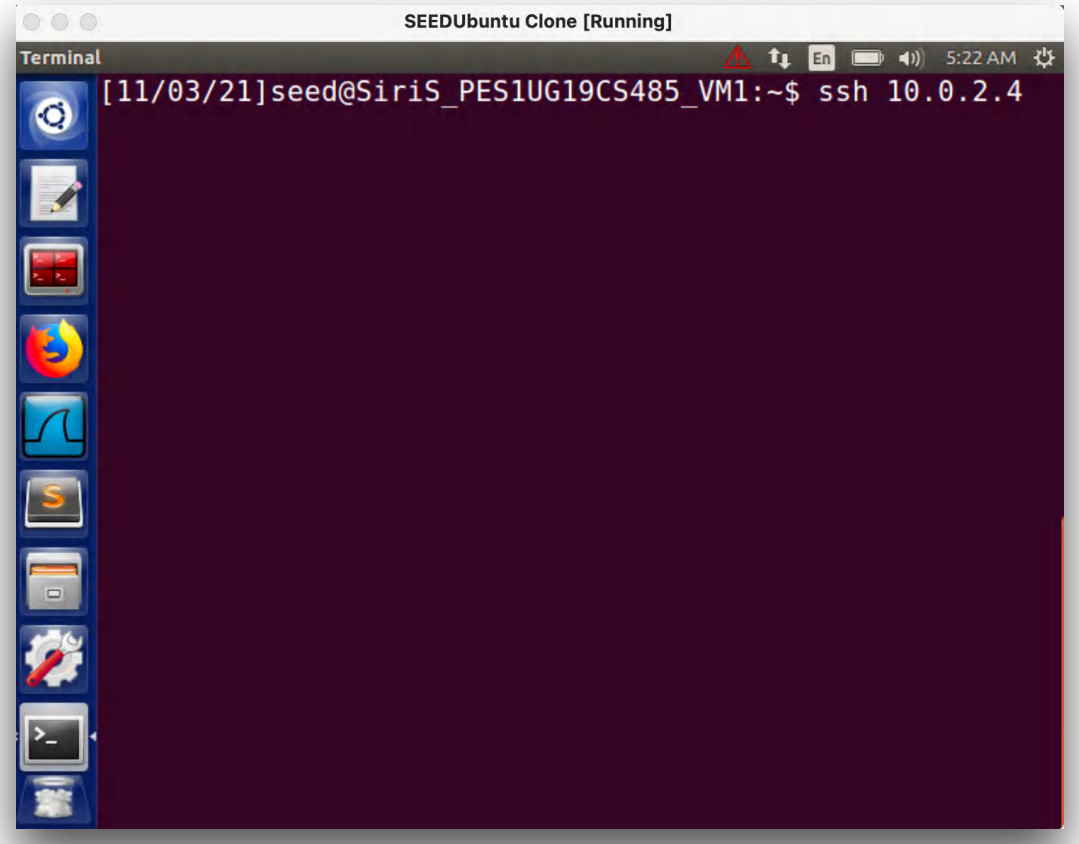


Step 3: Testing whether external website is blocked from VM1.

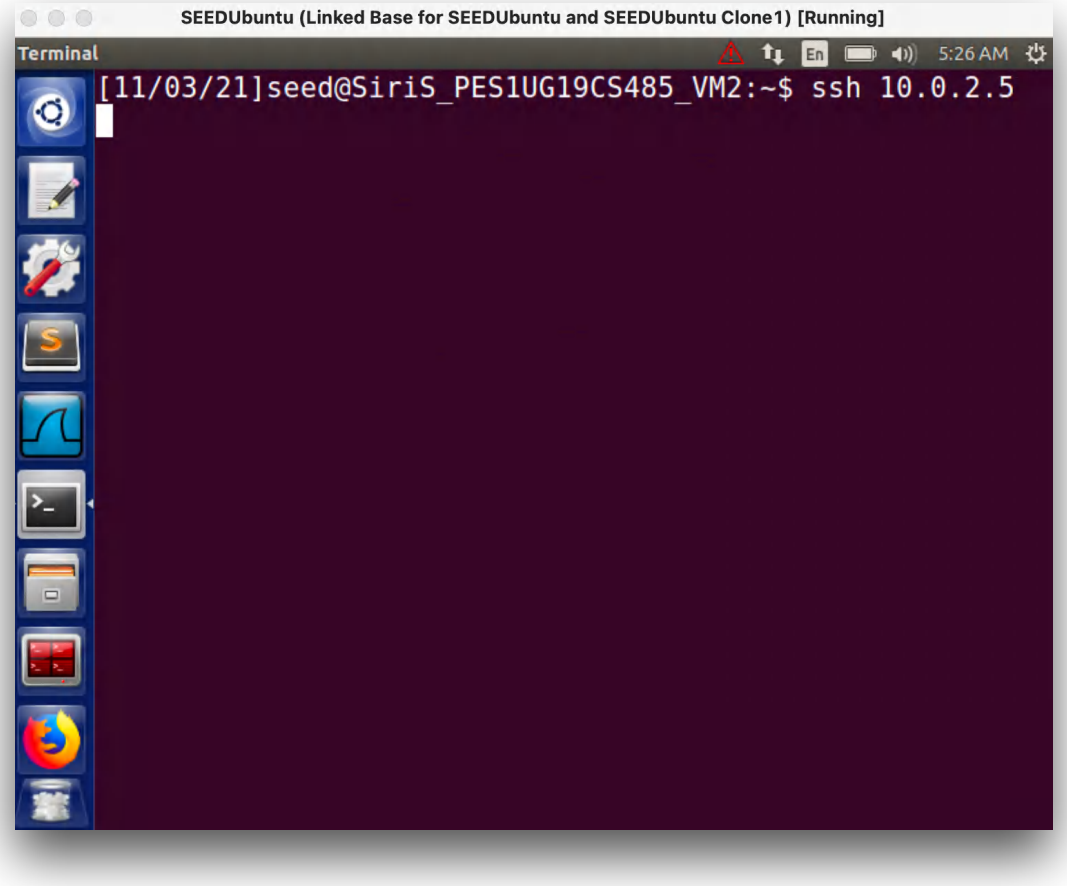
We can see that www.google.com does not load in the browser



Step 4: Testing whether *ssh* from VM1 to VM2 is blocked



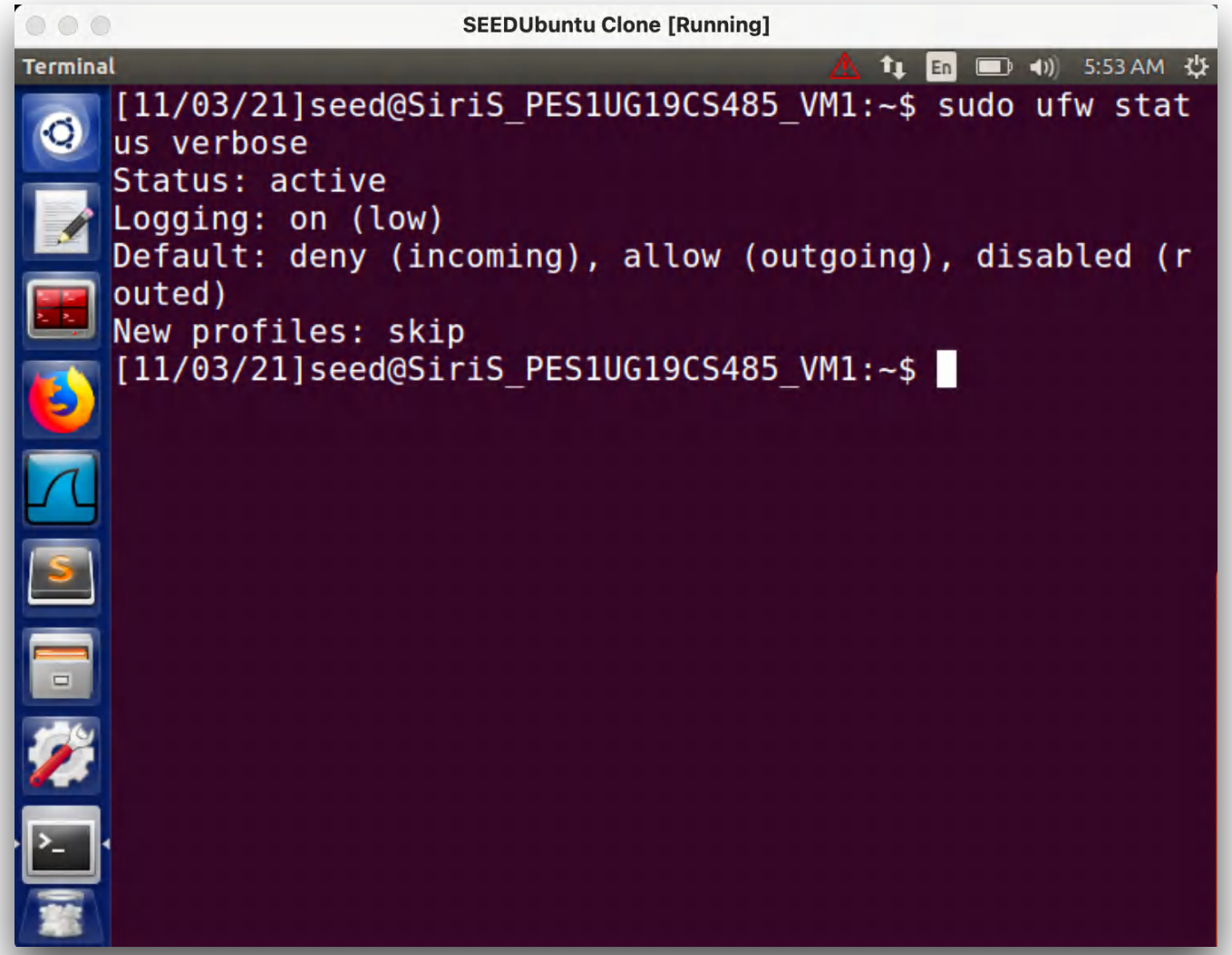
Step 5: Testing whether telnet from VM2 to VM1 is blocked



We can see that ssh is blocked both ways.

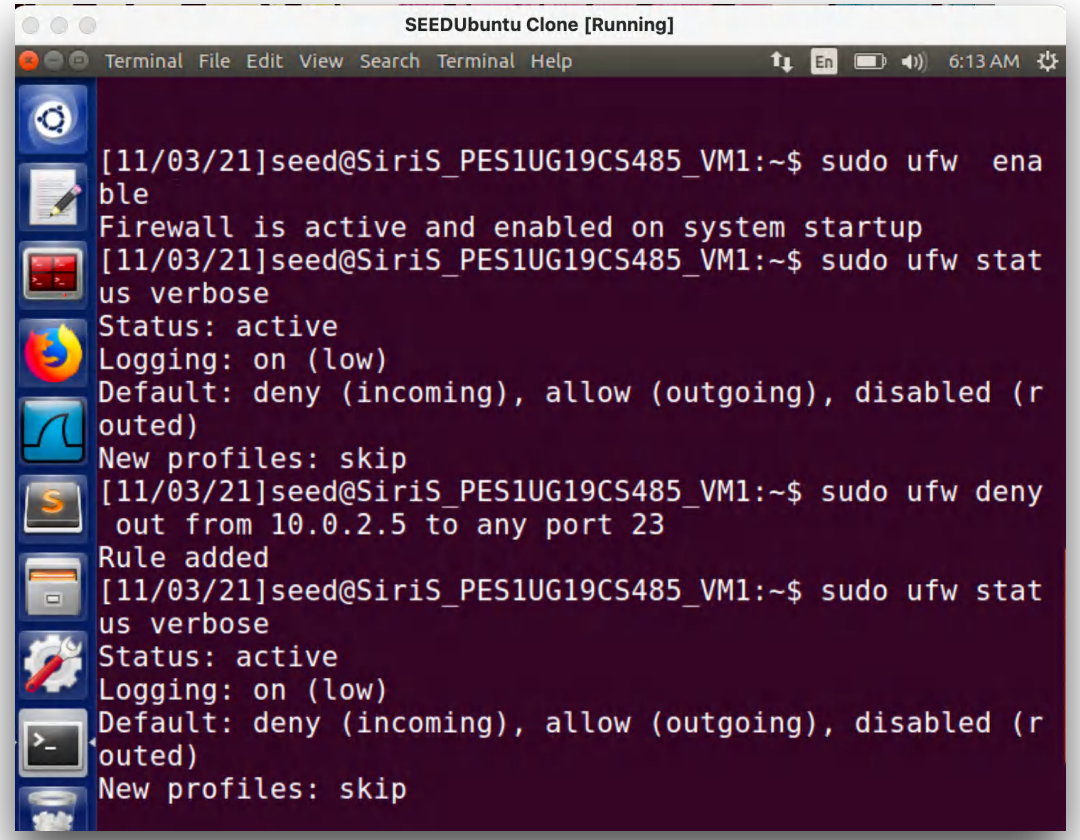
Task 3: Evading Egress Filtering

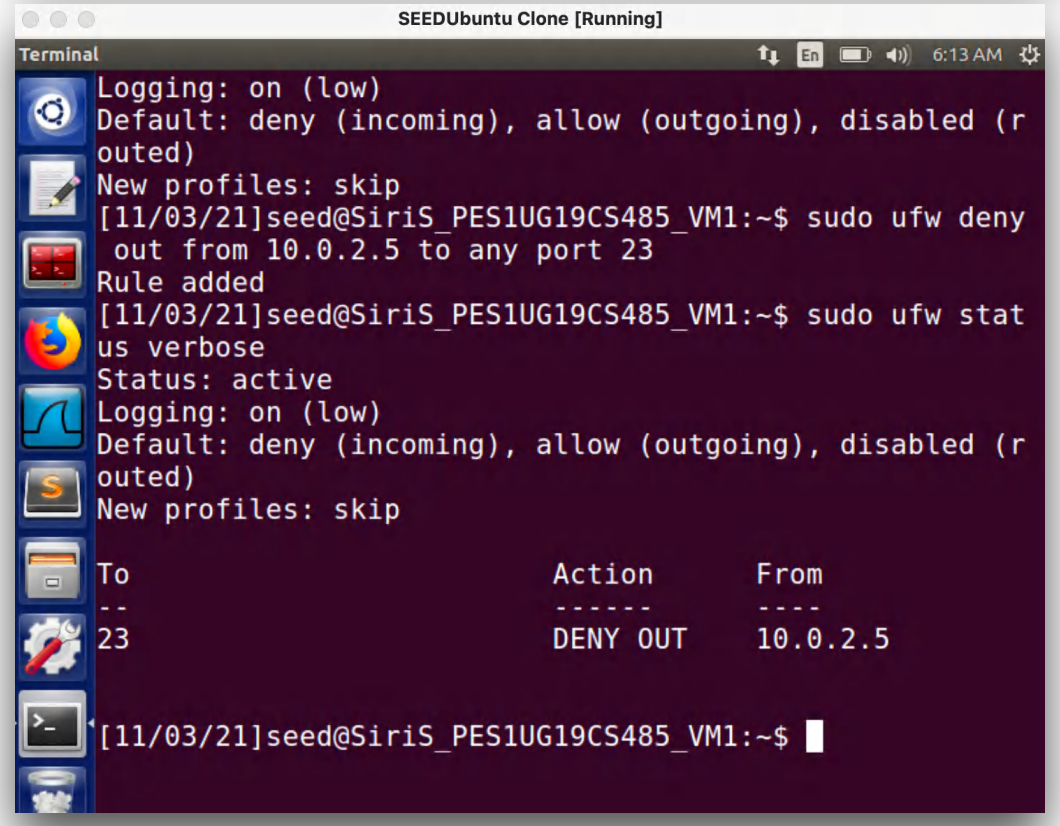
Deleting all the firewall rules from the previous task, we can see that the status is empty:



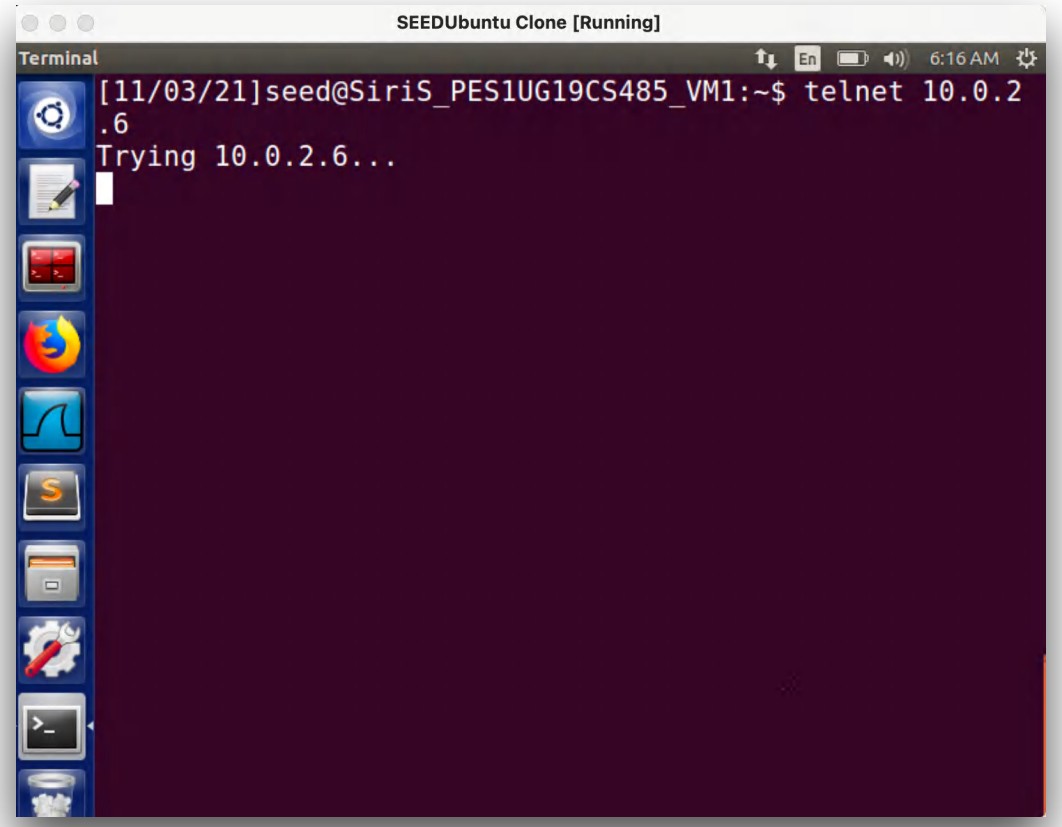
Task 3. a: Telnet to Machine B through the firewall

We will first block VM1 from being able to telnet to any other machine.

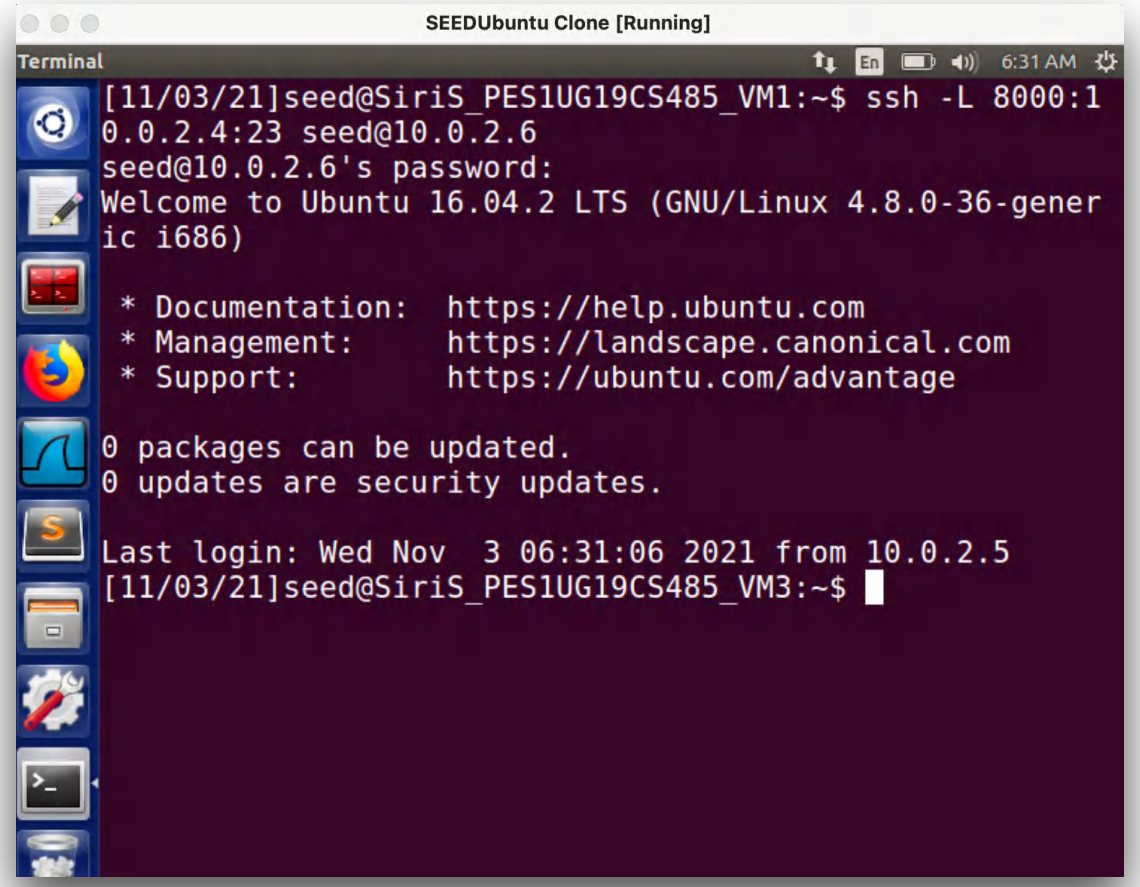




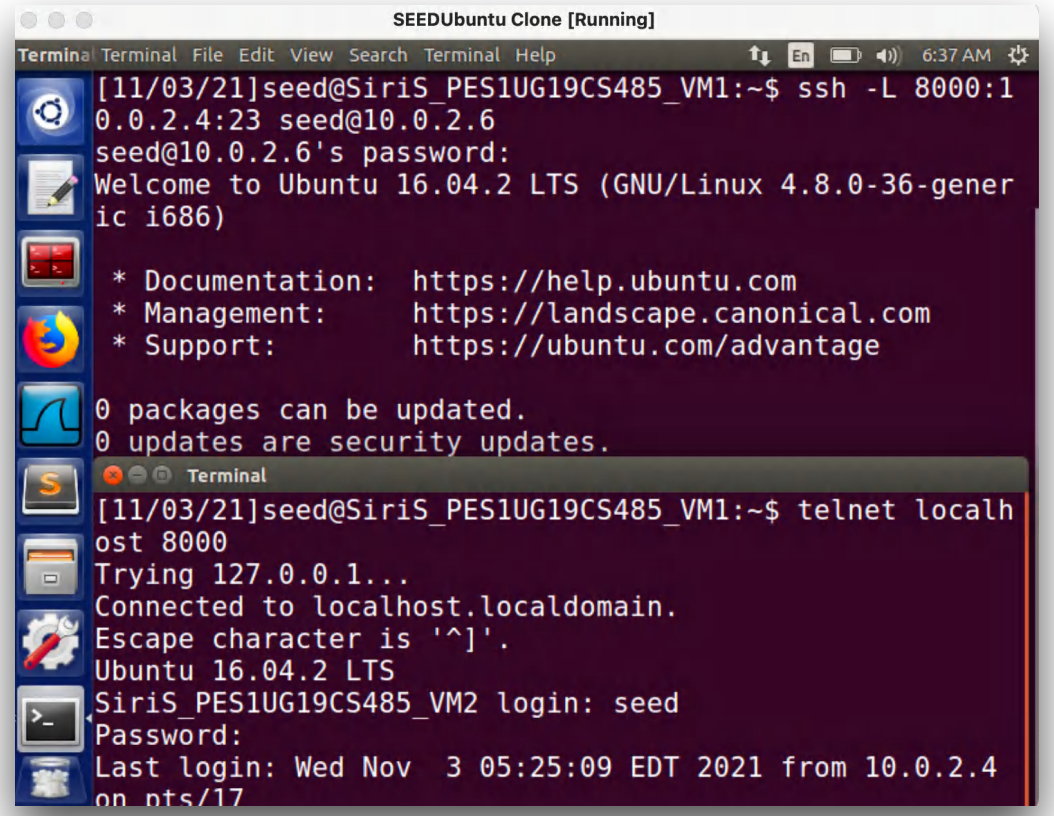
Now we check if we're able to telnet to VM3 from VM1:



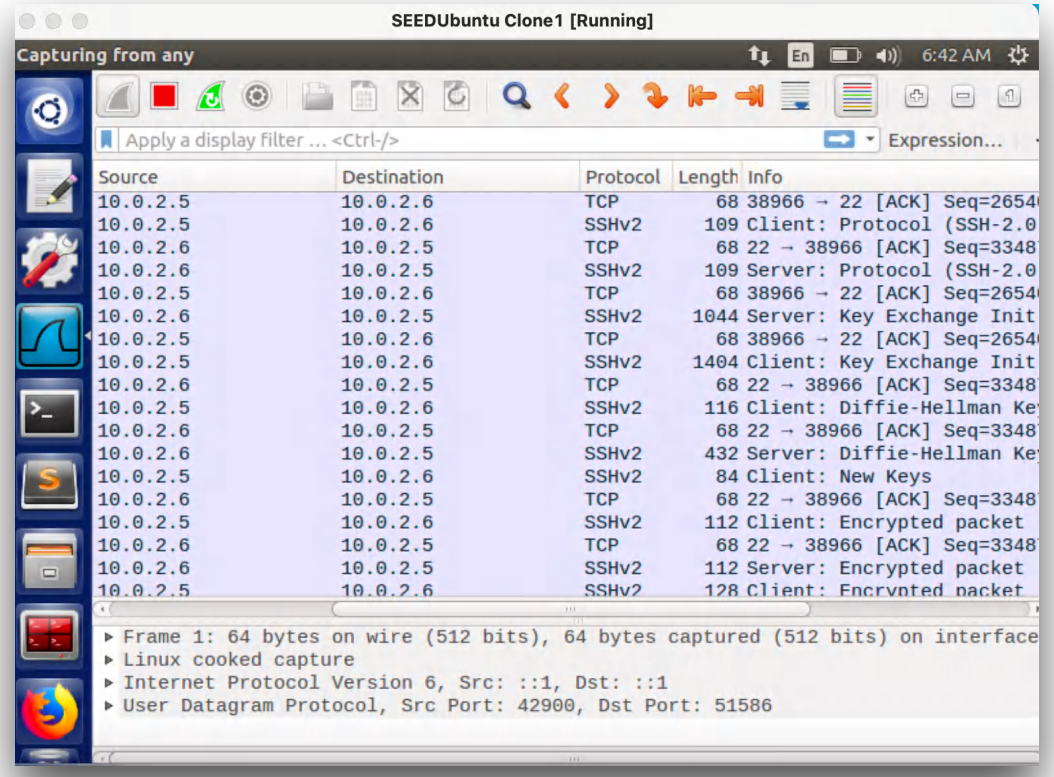
It doesn't work so we establish an ssh tunnel between VM1 and VM2 to allow VM1 to telnet to VM3 via VM2 :



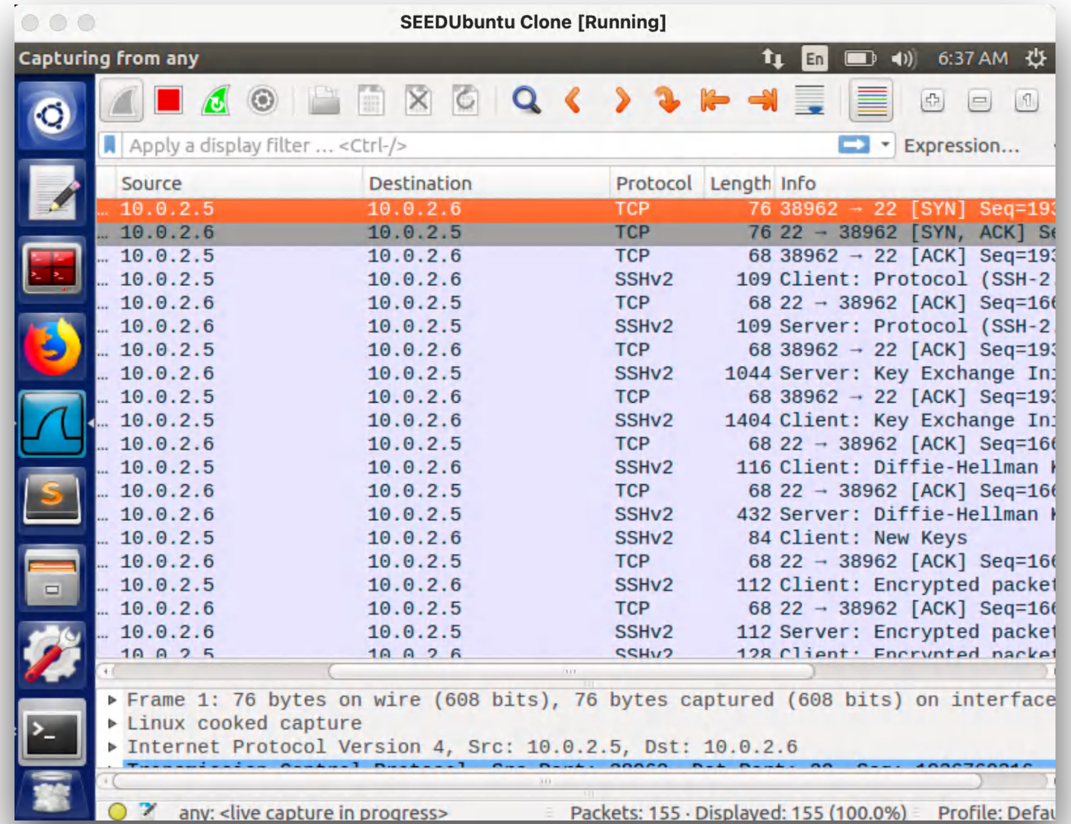
With the ssh tunnel setup, we can now telnet from VM1 to VM3 even though the firewall policy on VM1 denies outgoing telnet.



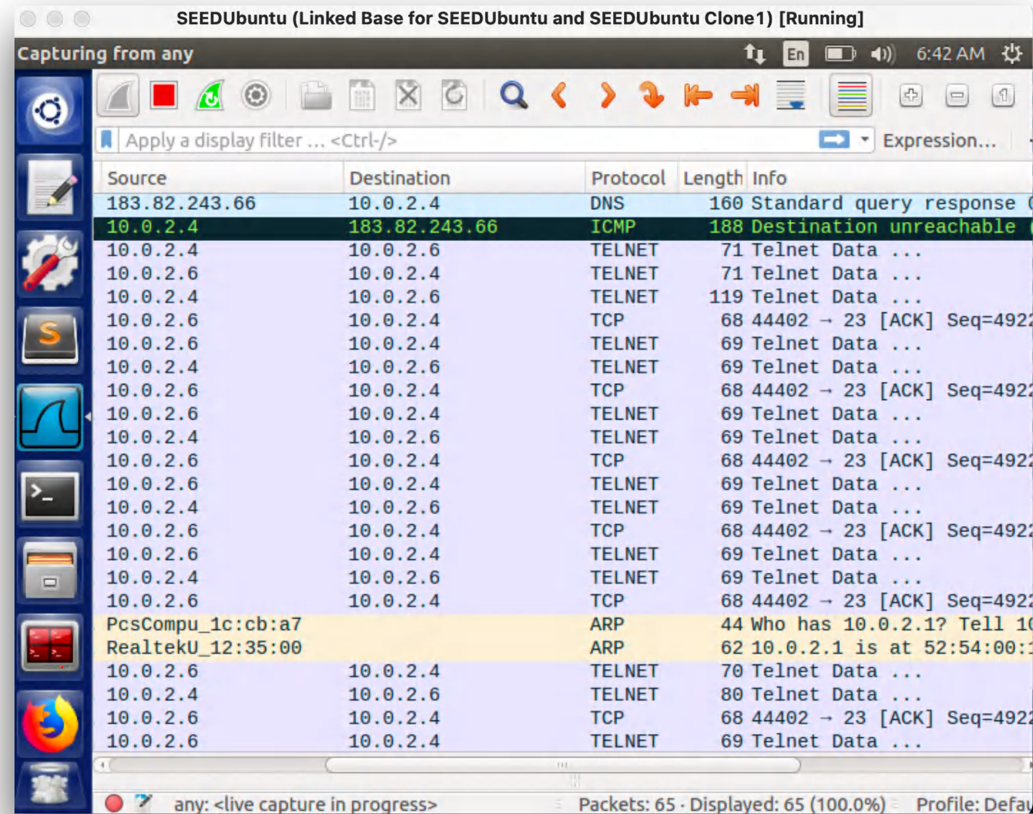
Wireshark Screenshots:



VM1's Wireshark shows that packets were sent and received from VM3



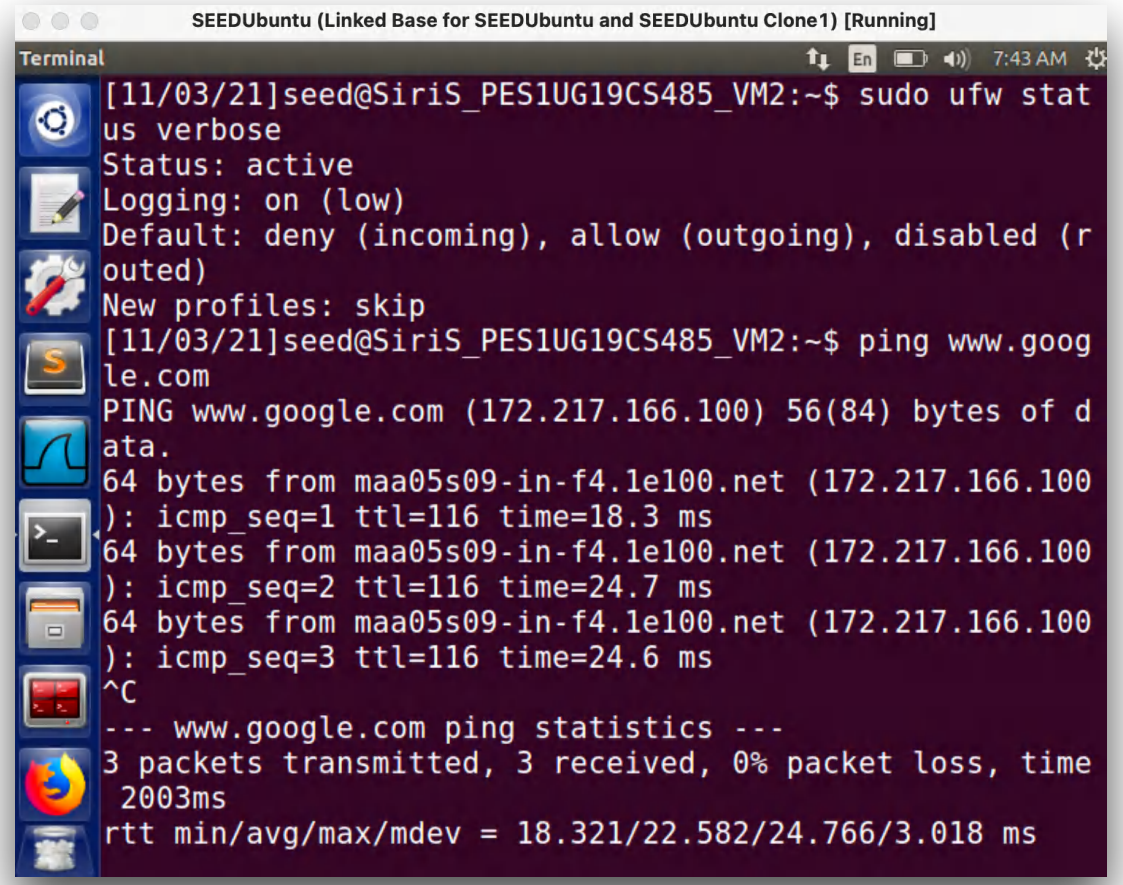
VM2's Wireshark shows that packets were sent and received from VM3.

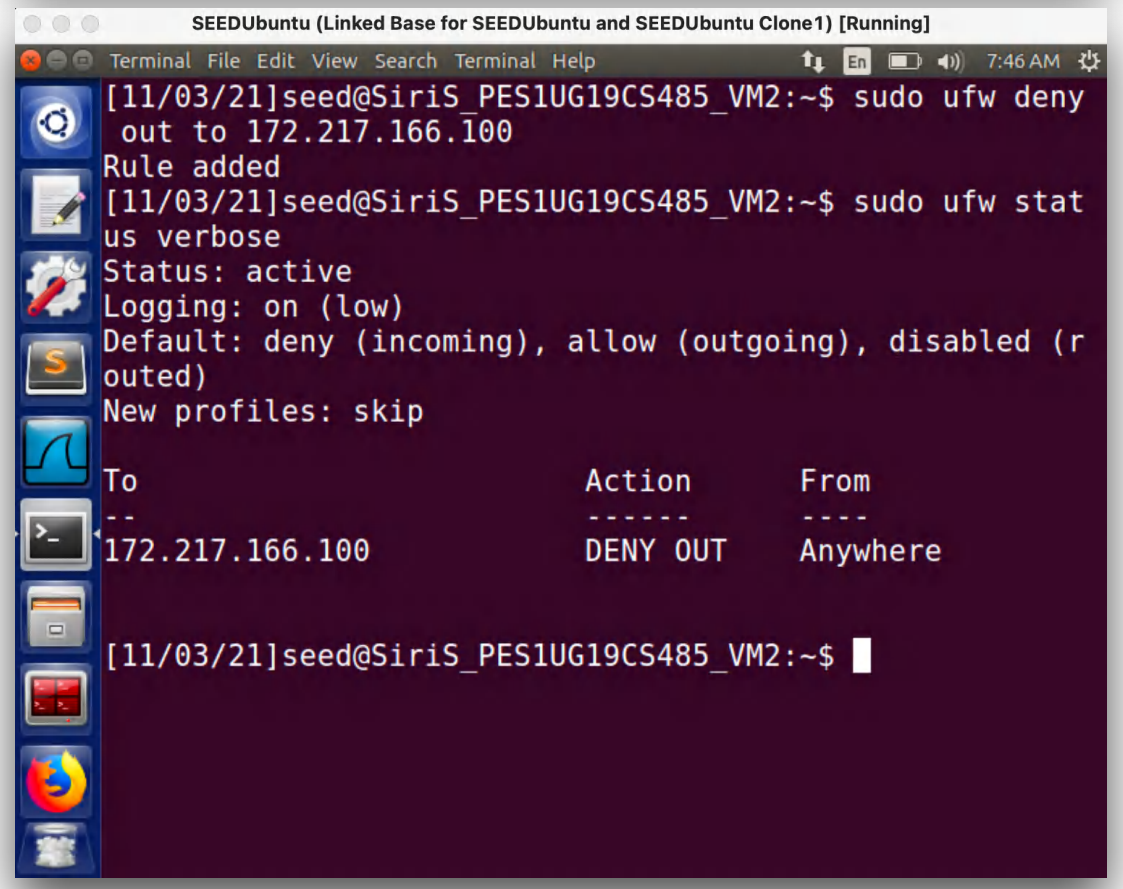


VM3's Wireshark that the packets were sent back to VM1. Hence all packets obtained by VM2 are forwarded to VM3 through port 23. VM3 is able to respond back to the telnet request from VM1, evading the firewall.

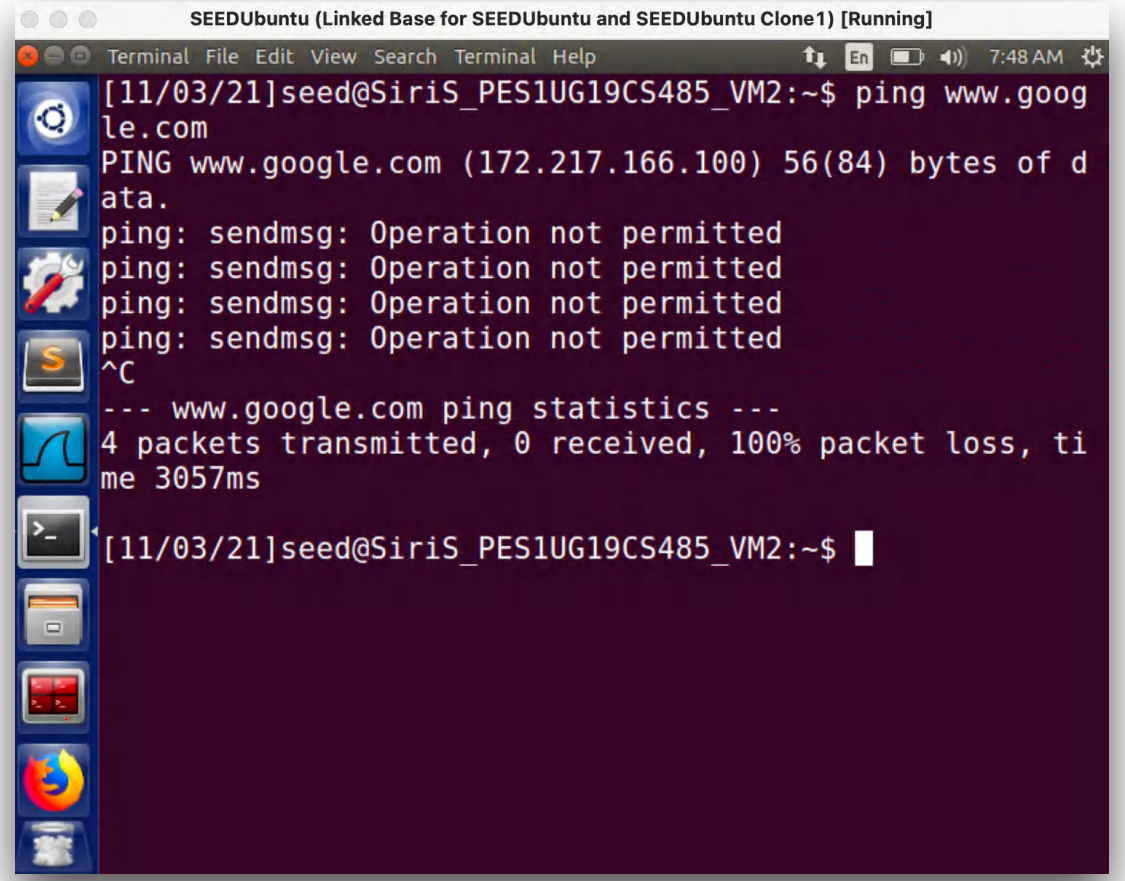
Task 3. b: Connecting to Google using SSH tunnel

First we delete other firewall rules, then we check the status of ***ufw*** and setup a firewall rule:

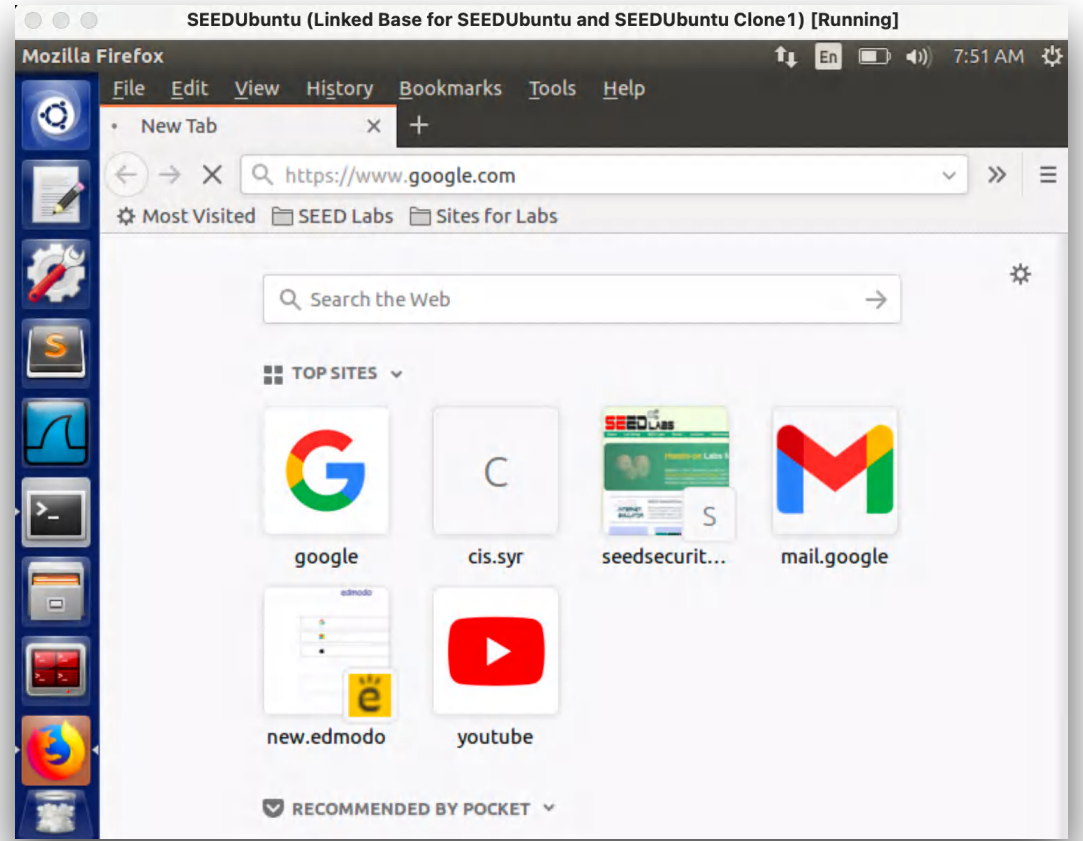




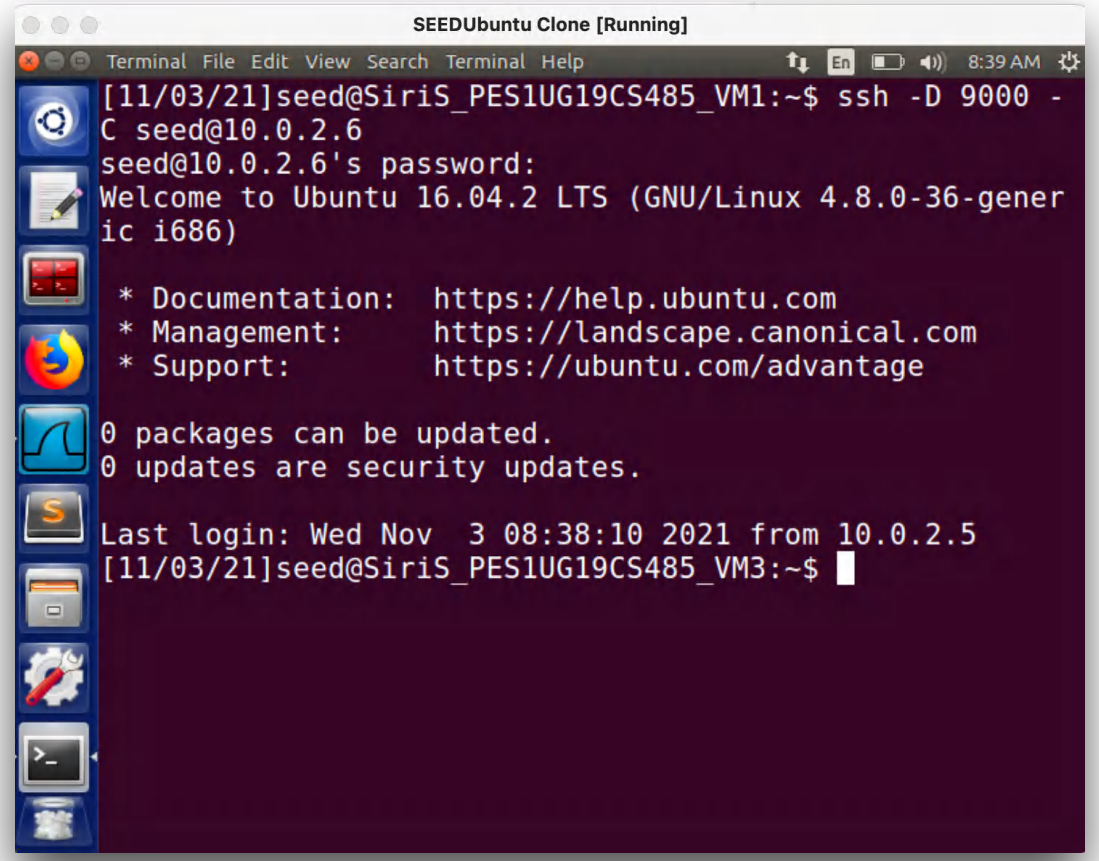
With the firewall rule in place, we can try to ping www.google.com. The operation is not permitted because it is being blocked by the firewall.



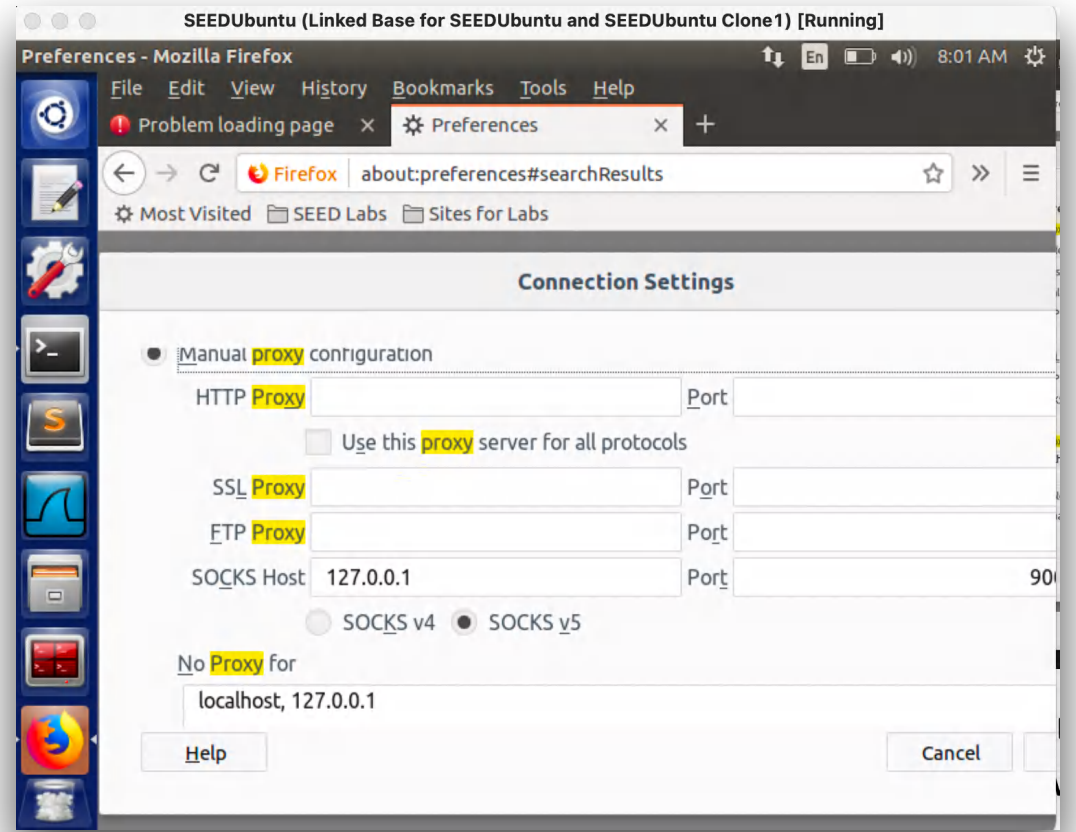
We can also try to visit www.google.com through the browser, where we are unable to load the page.



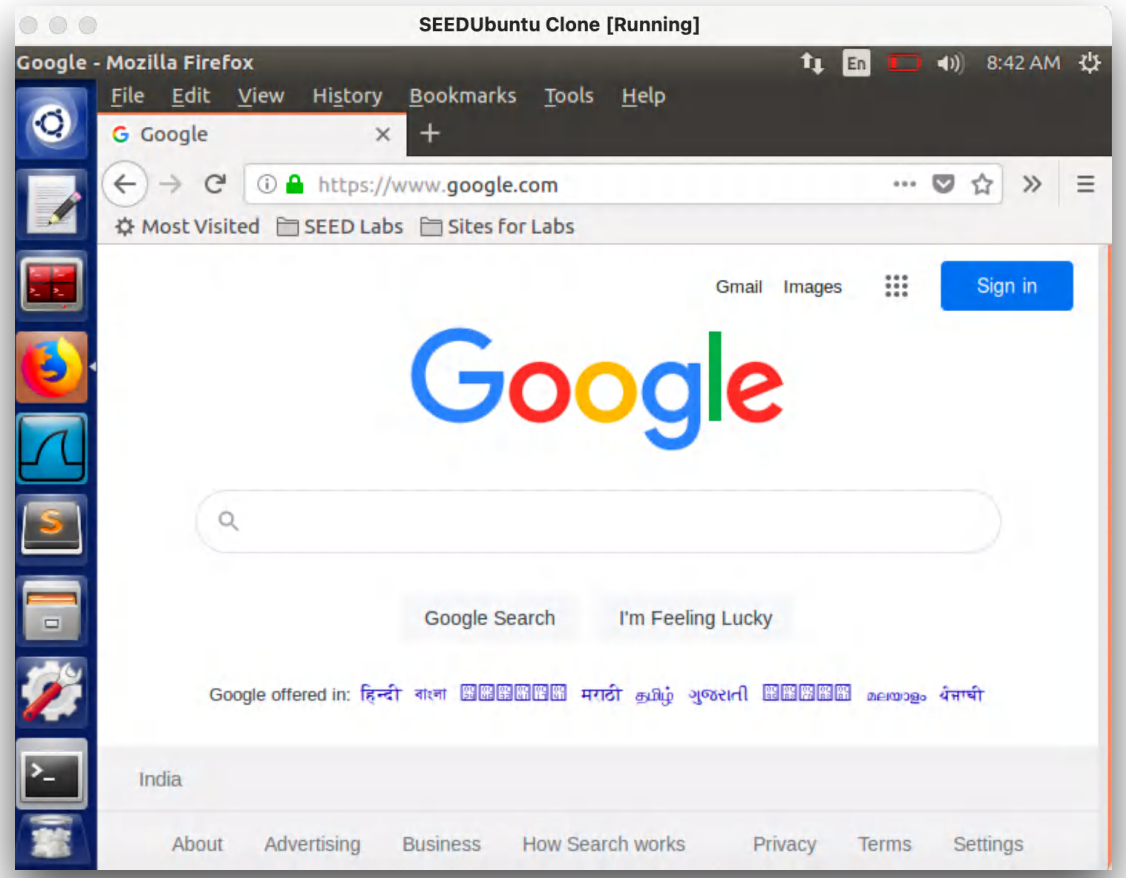
We now setup a ssh tunnel with dynamic port forwarding between VM1 and VM3. With this tunnel setup, VM1 will be able to use its local port 9000 to send a request to www.google.com via VM3.



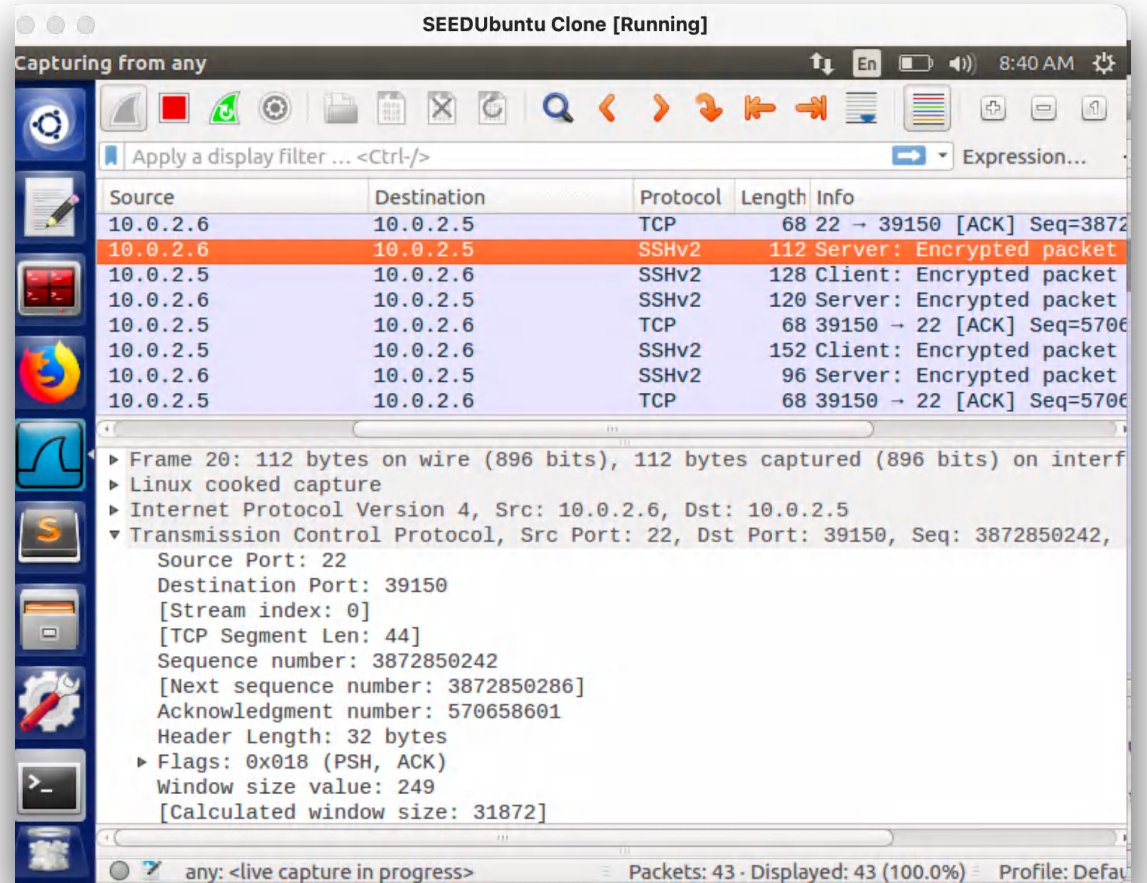
To use the established tunnel, we need to set the proxy settings in the Firefox browser as shown below.



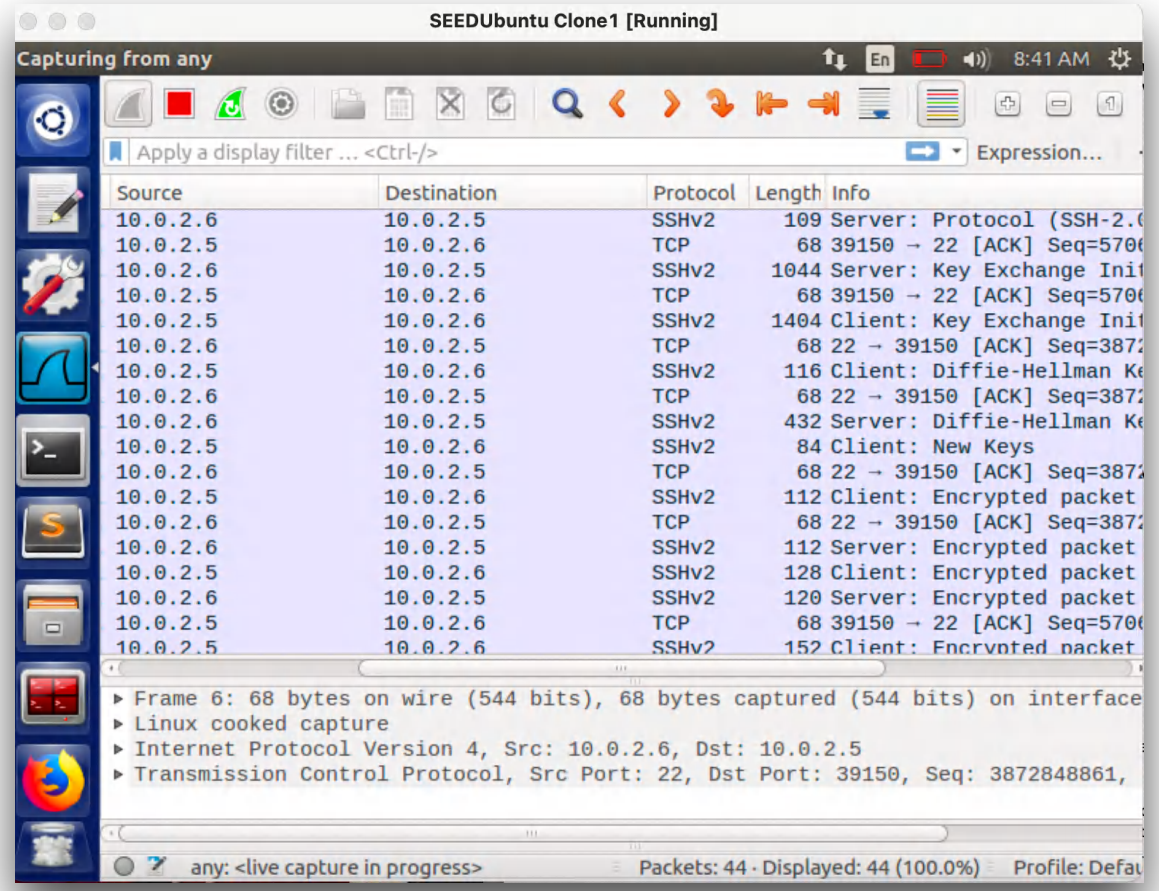
With the proxy settings in place, we are able to visit www.google.com from VM1 even though the firewall has a policy to block it. We can see that page has loaded:



Wireshark Capture from VM 1:

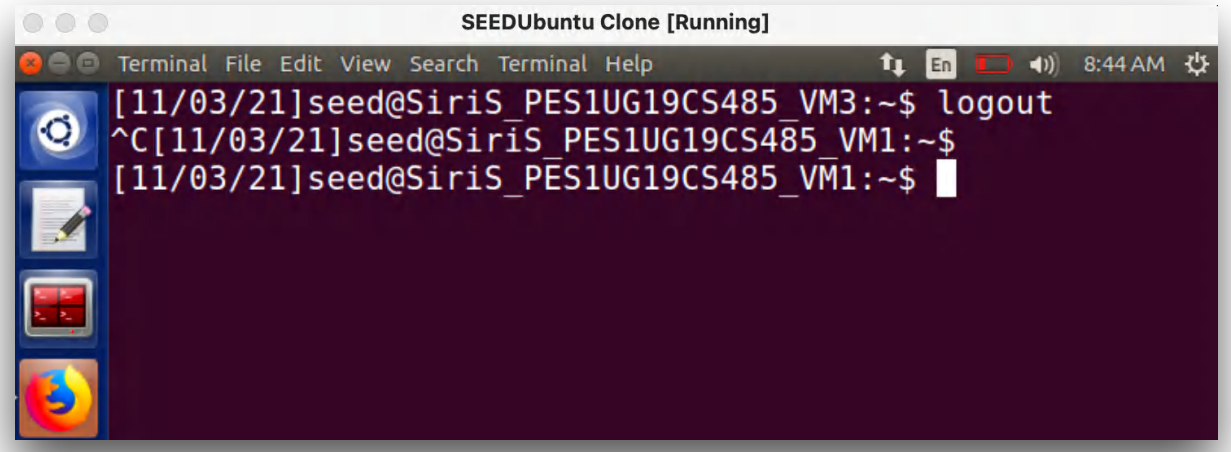


Wireshark Capture from VM 3:

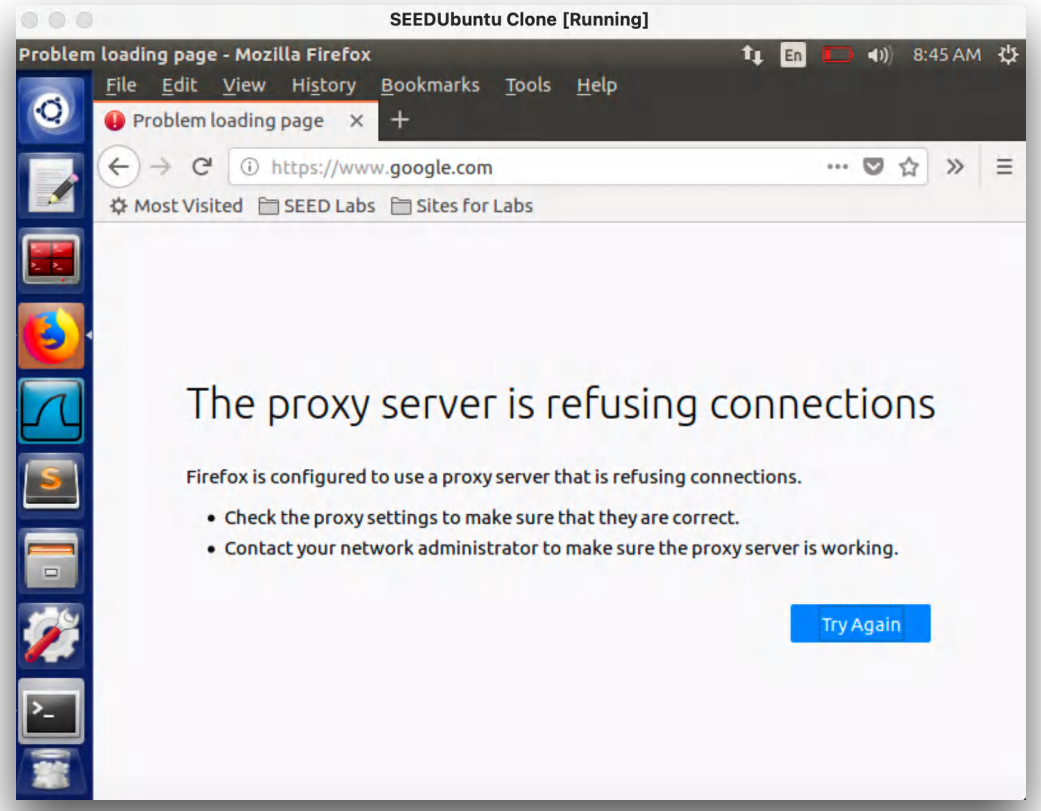


Wireshark capture on VM1 shows that it sent packets to VM3 instead of Google's IP and from VM3's wireshark, we see that it has contacted google on behalf of VM1.It sends back the replies to VM1 thereby bypassing the firewall.

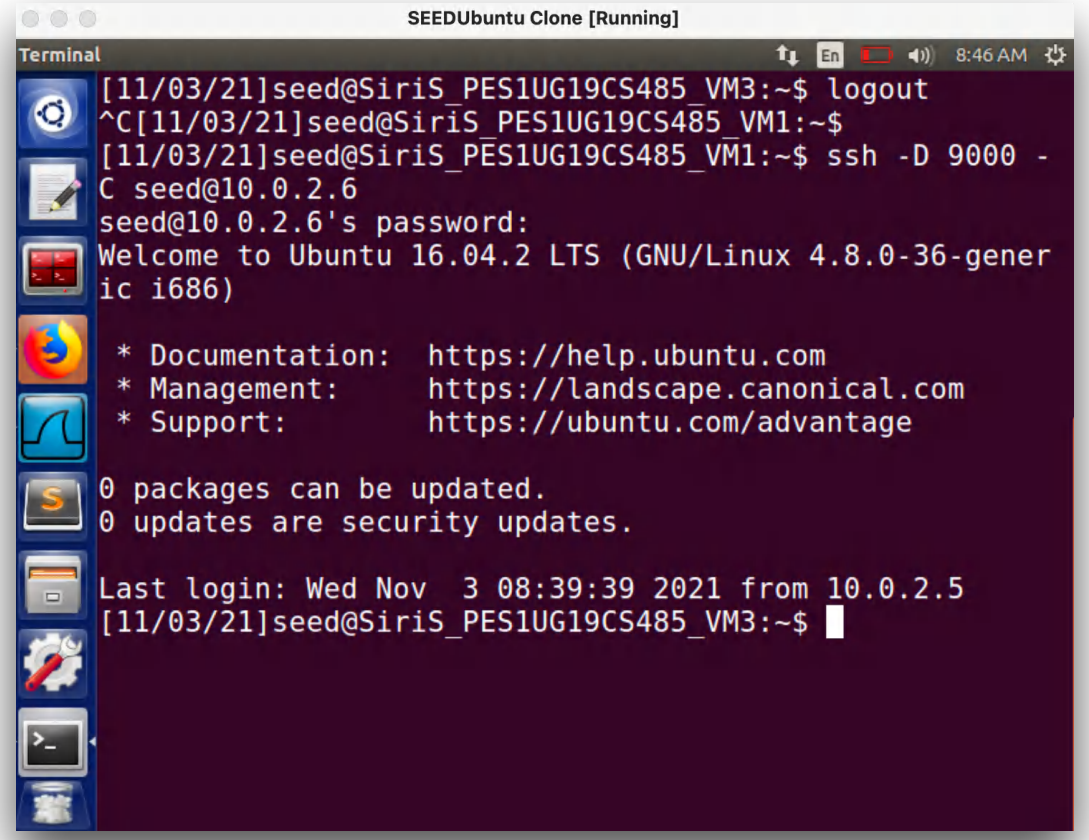
Once we logout of the *ssh* and break the tunnel, www.google.com is no longer accessible:

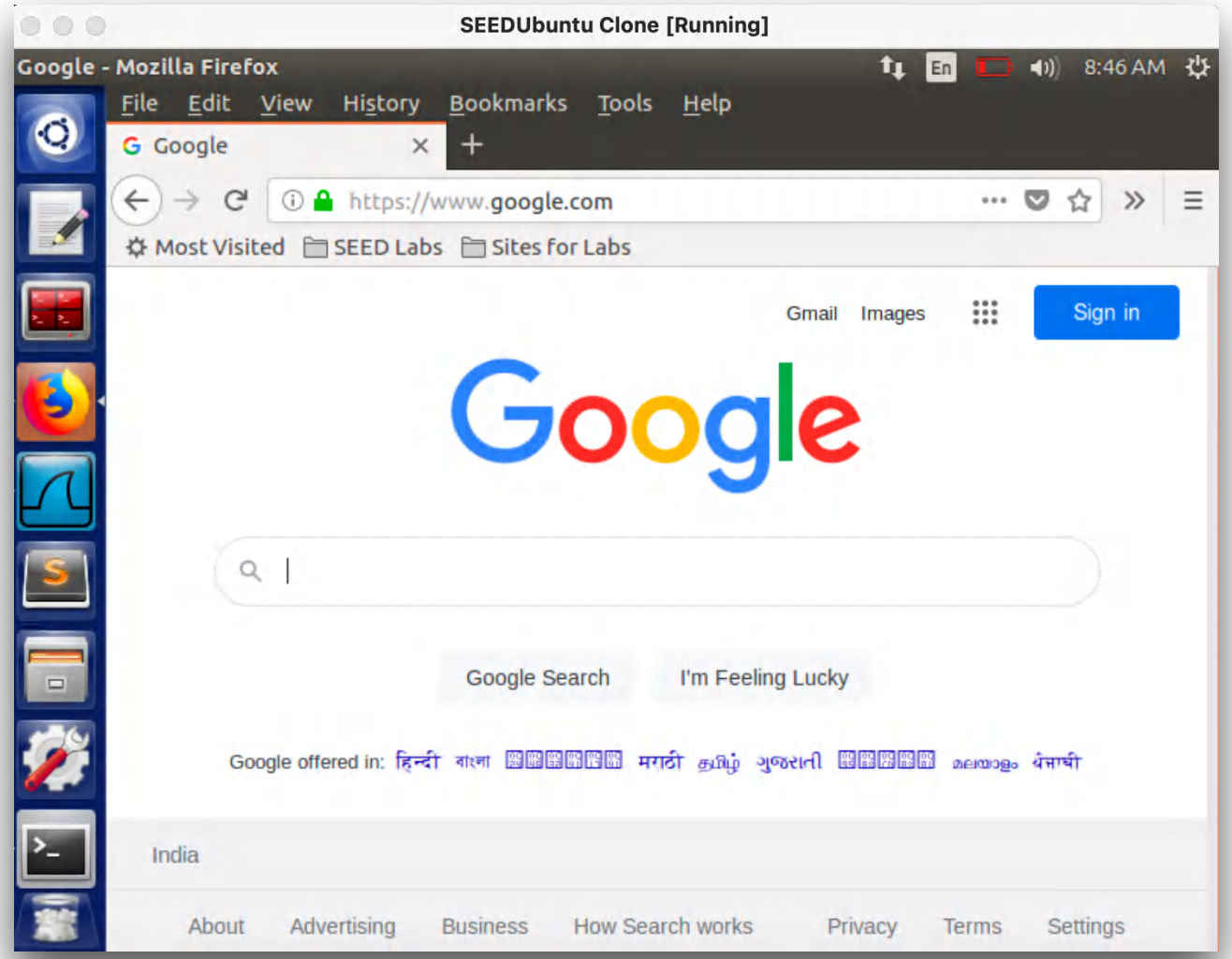


We can see that we are unable to access www.google.com instead it says that proxy is refusing connections:



However, if we activate *ssh* and recode the page we are able to access the website:

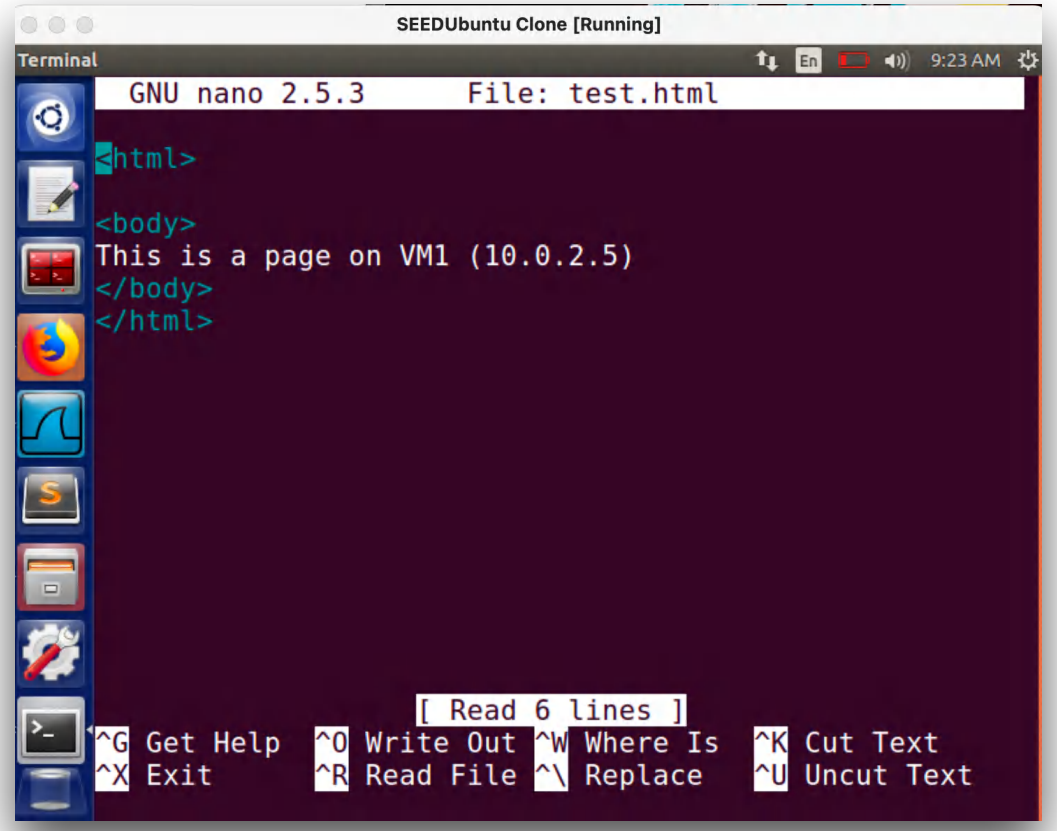




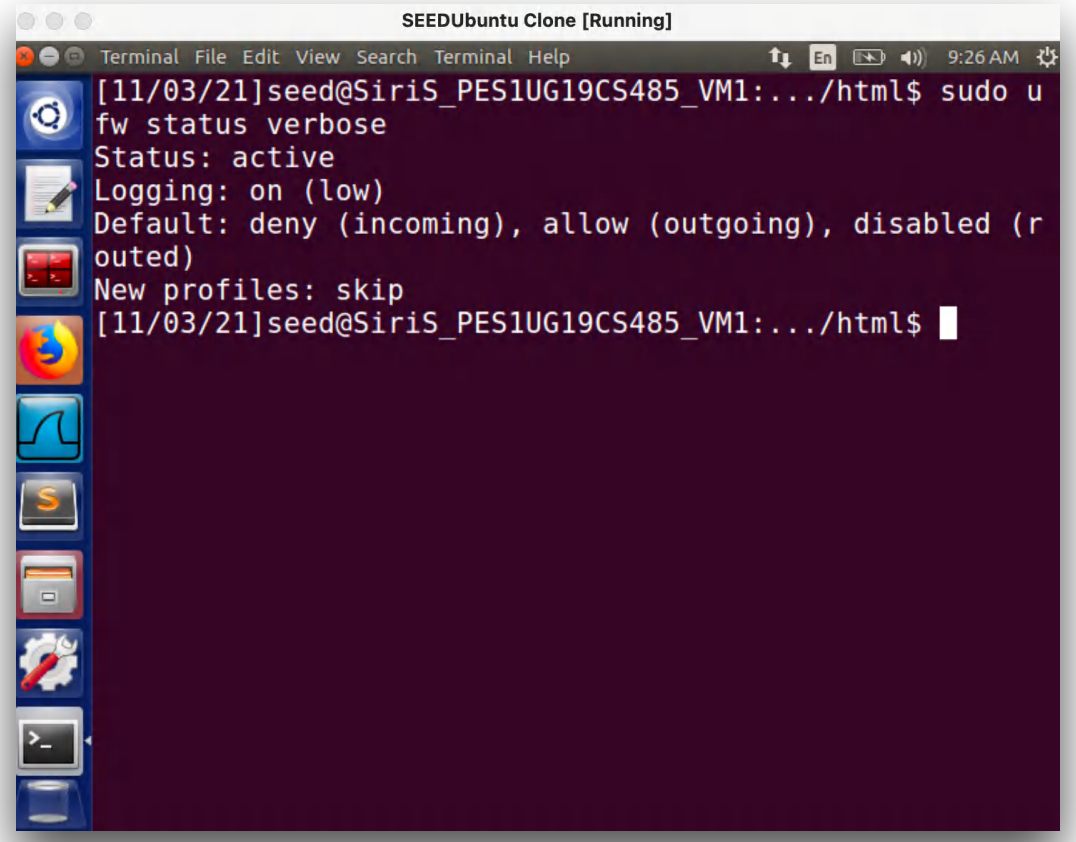
We can see that the website is able to load as soon as we re-enable the ssh tunnel.

Task 4: Evade Ingress Filtering

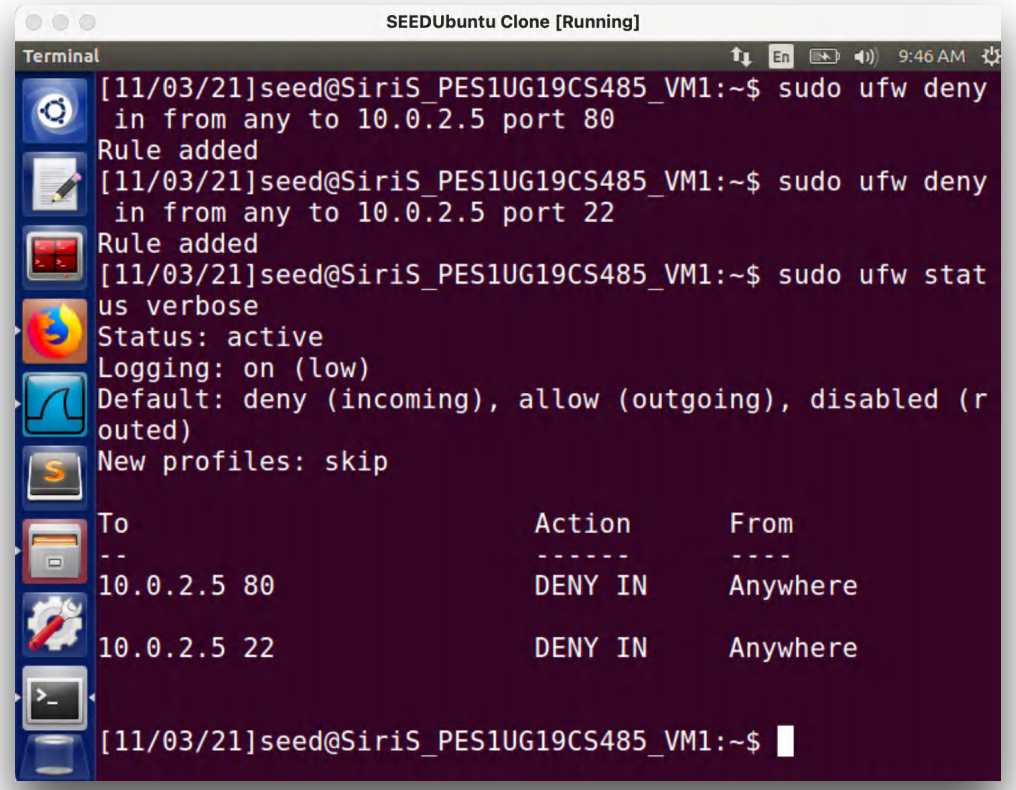
First remove all firewall rules that were added in the previous tasks. The content of the page in VM1 is:



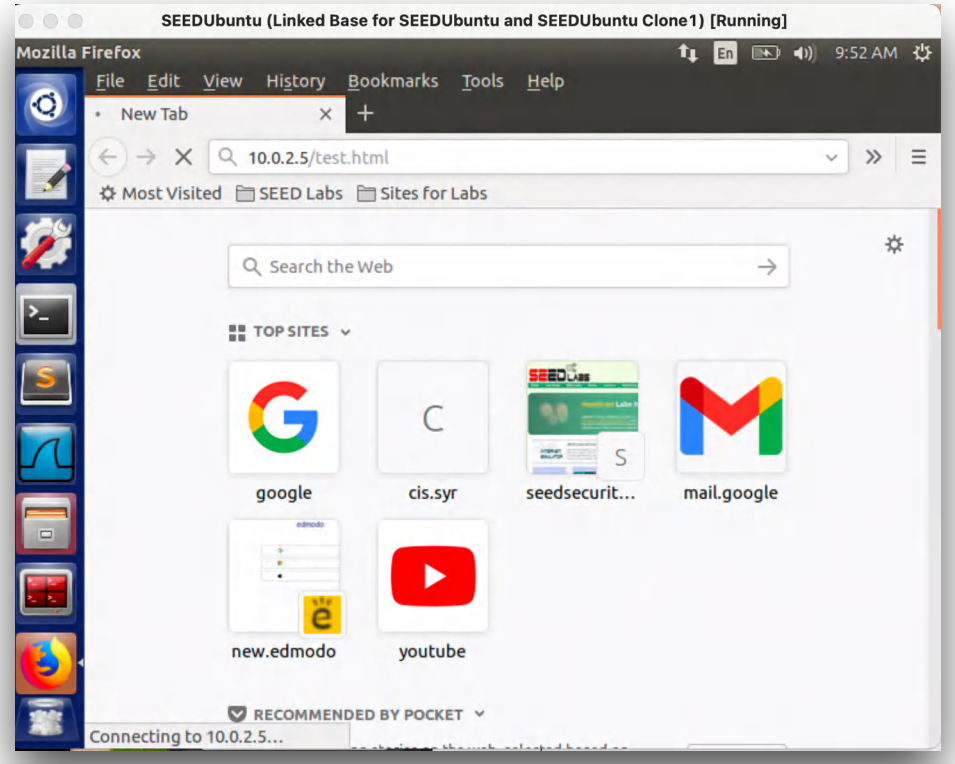
No firewall rules setup on VM1:



We next block incoming requests on port 80 and port 22 on VM1.

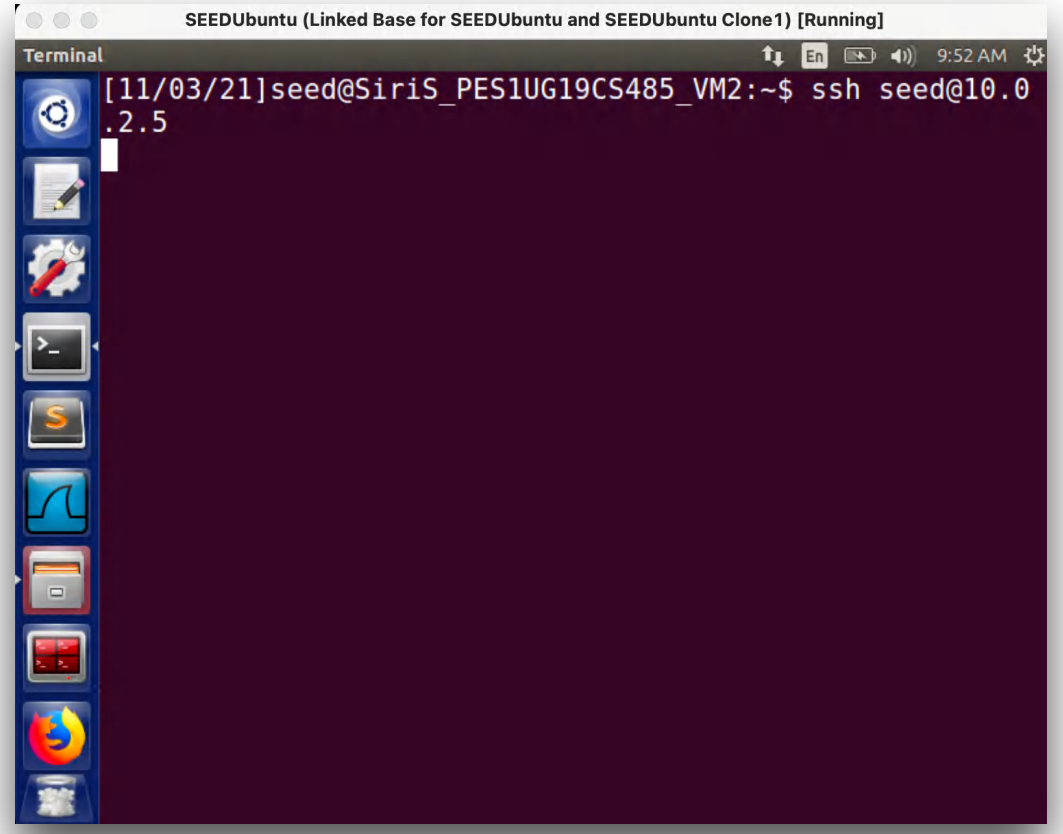


Upon clearing cache and trying to access the page from VM2:

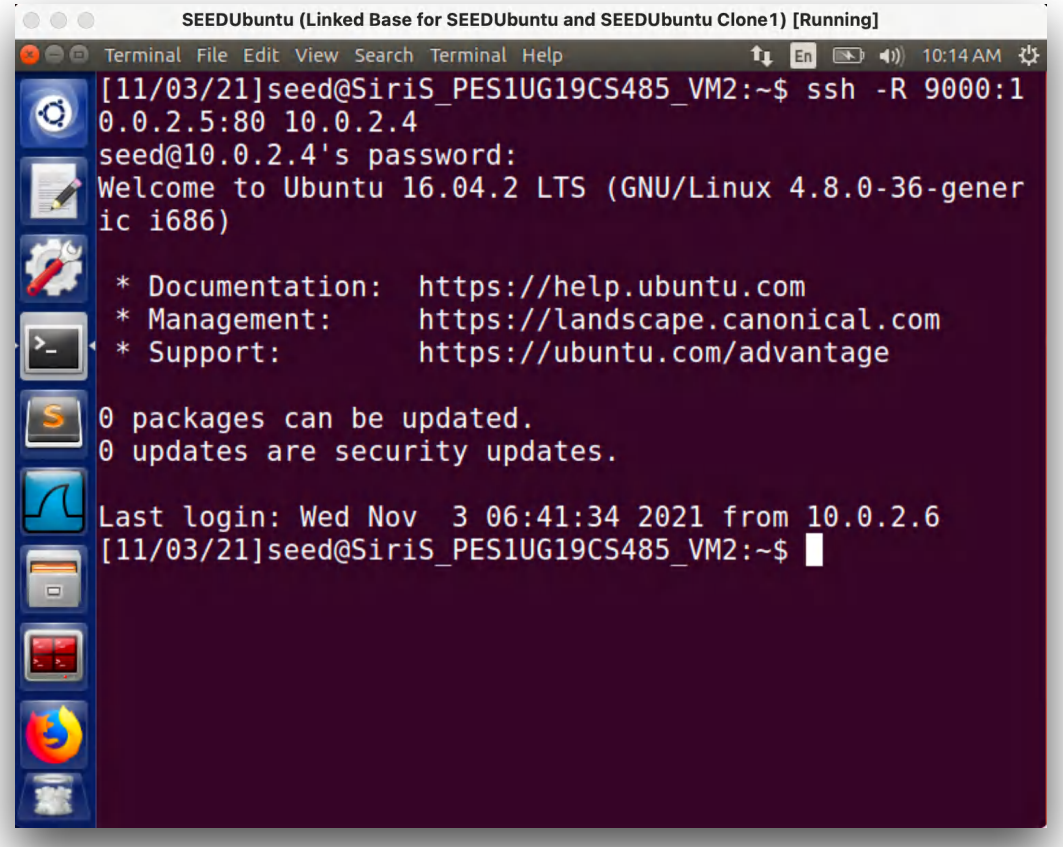


It takes too long to load because of the firewalls

Also, because port 22 is blocked, VM2 cannot ssh to connect to VM1.



Now lets set up a reverse ssh tunnel as shown below



Thus the page is now accessible on VM2 even though the firewall rules were existing.

However when we logout of the ssh connection, the page again doesn't load and displays a ‘Unable to connect’ message: