DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Session: Jan 2021 – May 2021

COMPUTER NETWORK SECURITY
LAB – 3

NAME : AMRUTHA BS

Lab Setup

Task 1, 2

VM1 : 10.0.2.8

VM2 : 10.0.2.15

Task 3

VM1 : 10.0.2.15

VM2 : 10.0.2.5

VM3 : 10.0.2.4

Task 4

VM1 : 10.0.2.10

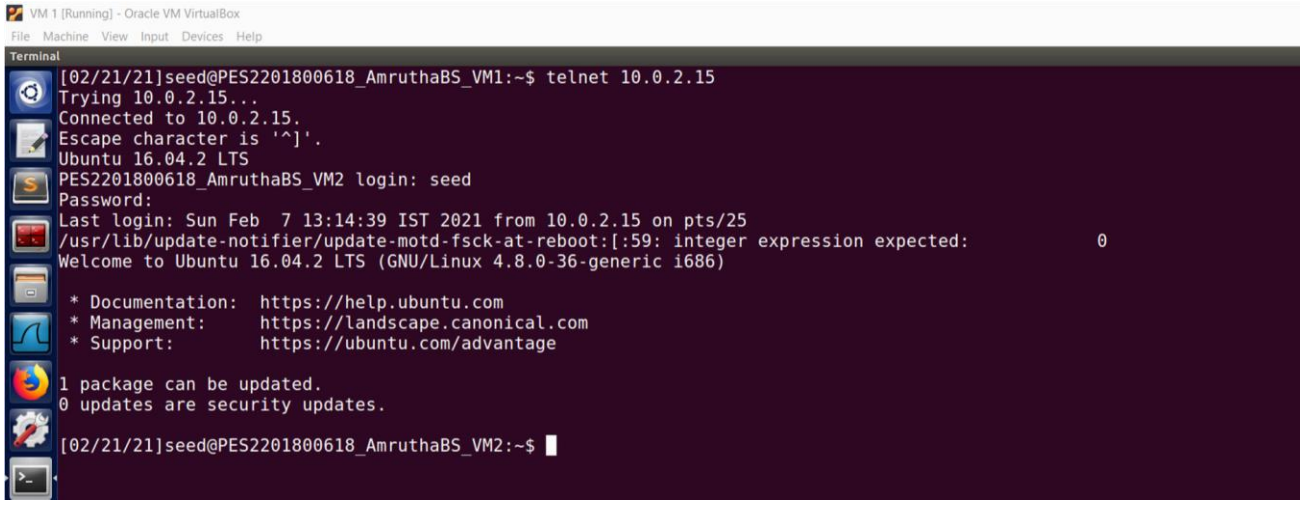
VM2 : 10.0.2.11

Task 1: Using Firewall

In this task, we will use two VMs (VM1 and VM2). We first show that VM1 can telnet to VM2.

Command:

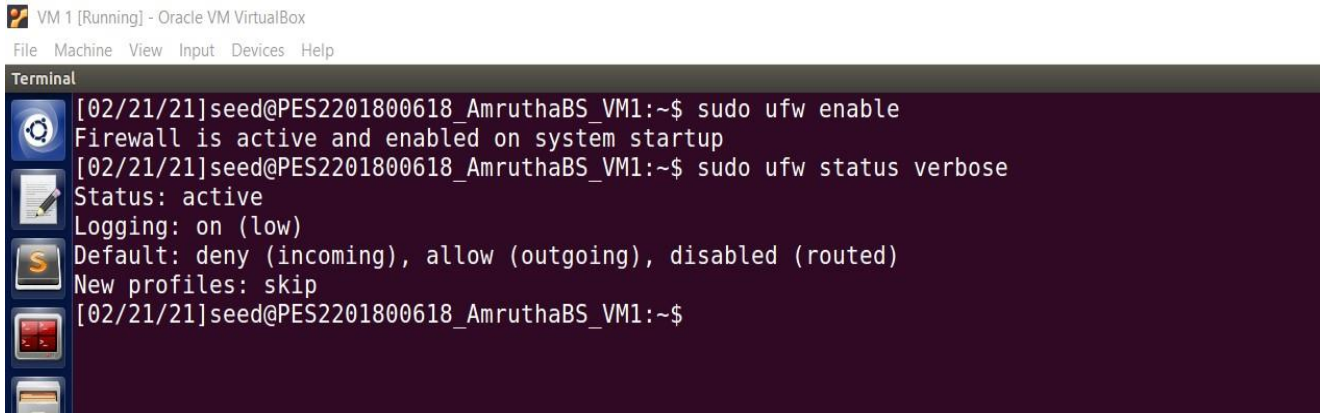
telnet 10.0.2.15



We need to first prevent VM1 from being able to telnet to VM2. For this we will configure the firewall using ufw. We will first enable the firewall on VM1.

Commands:

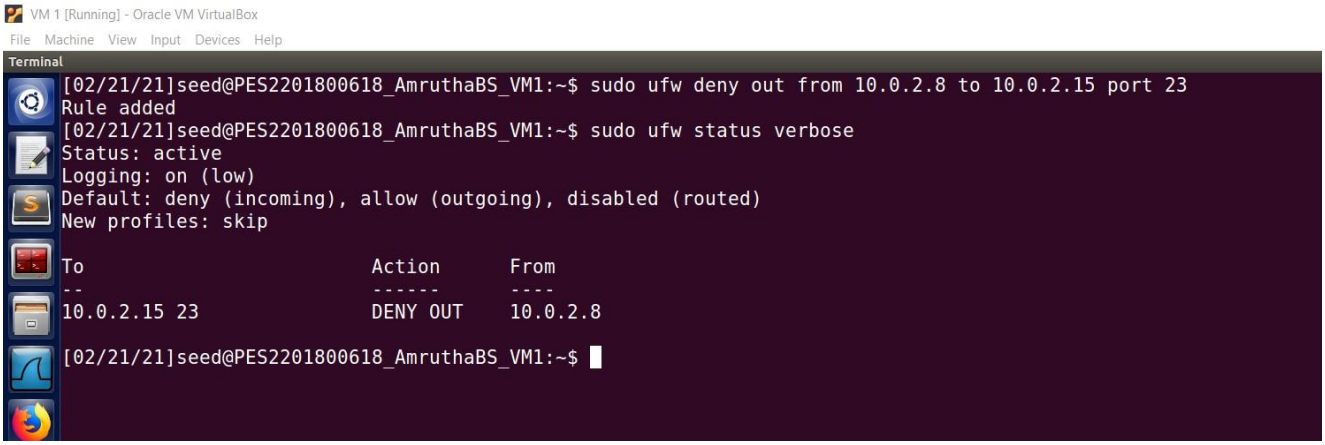
sudo ufw enable sudo ufw status verbose



We will next configure the firewall on VM1 to deny telnet (port 23) to VM2.

Command:

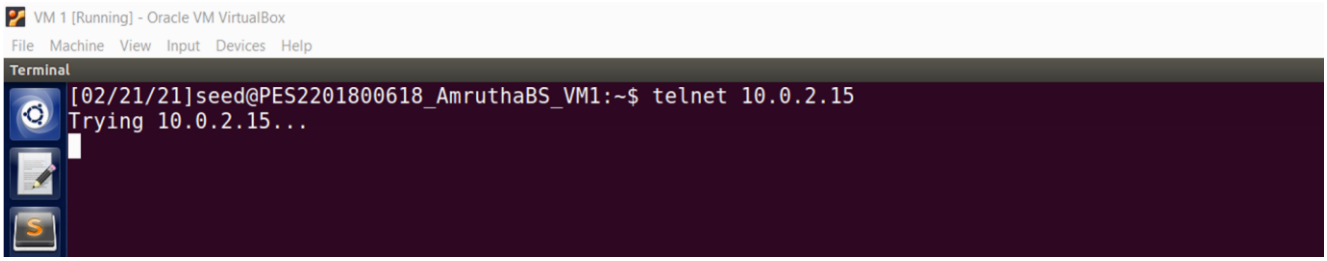
sudo ufw deny out from 10.0.2.8 to 10.0.2.15 port 23 sudo ufw status verbose



We now try to telnet from VM1 to VM2. Due to the firewall rule, the telnet is denied.

Command:

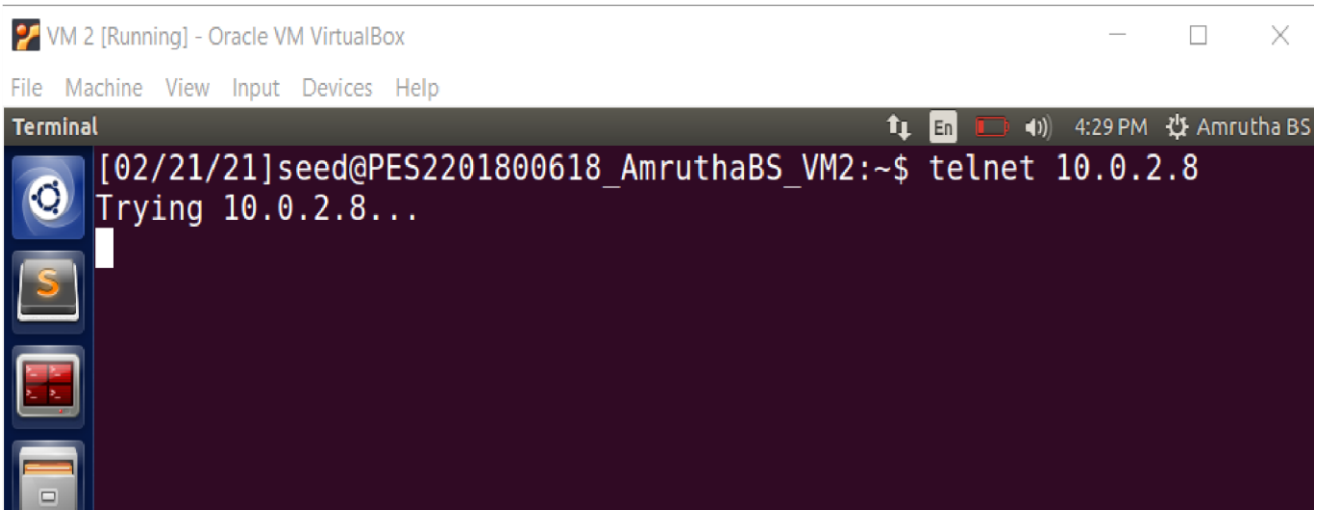
telnet 10.0.2.15



We next need to deny telnet from VM2 to VM1. We will first test whether telnet works from VM2 to VM1.

Command:

telnet 10.0.2.8

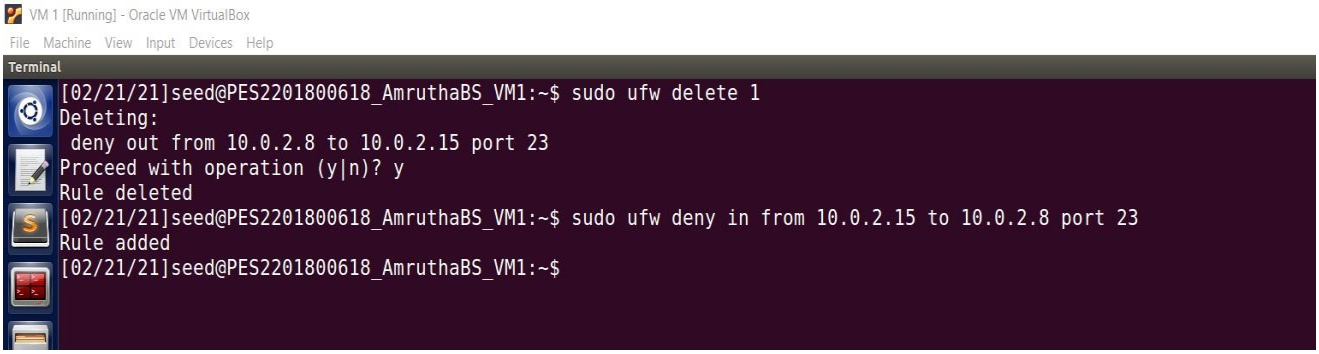


We next delete the firewall rule in VM1. We add a rule to prevent VM2 from being able to do telnet to VM1.

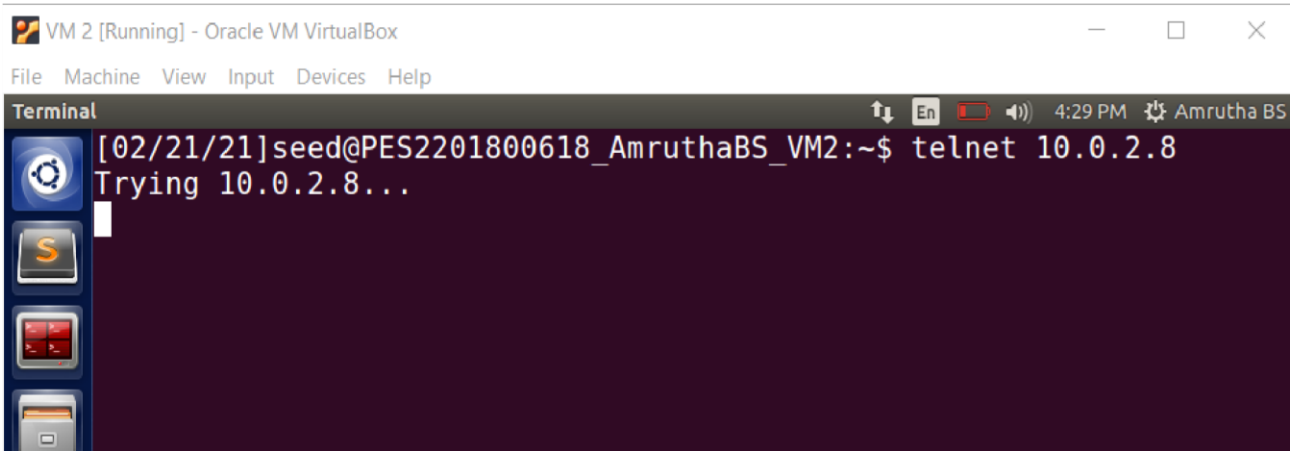
Commands:

sudo ufw delete 1

sudo ufw deny in from 10.0.2.10 to 10.0.2.9 port 23

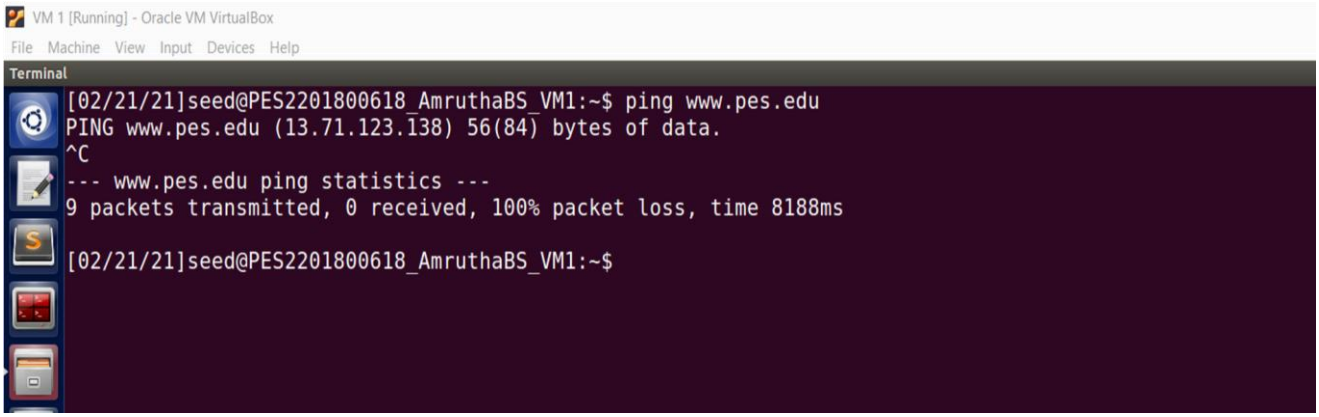


We can try to telnet from VM2 to VM1 again. The telnet is blocked and does not work.



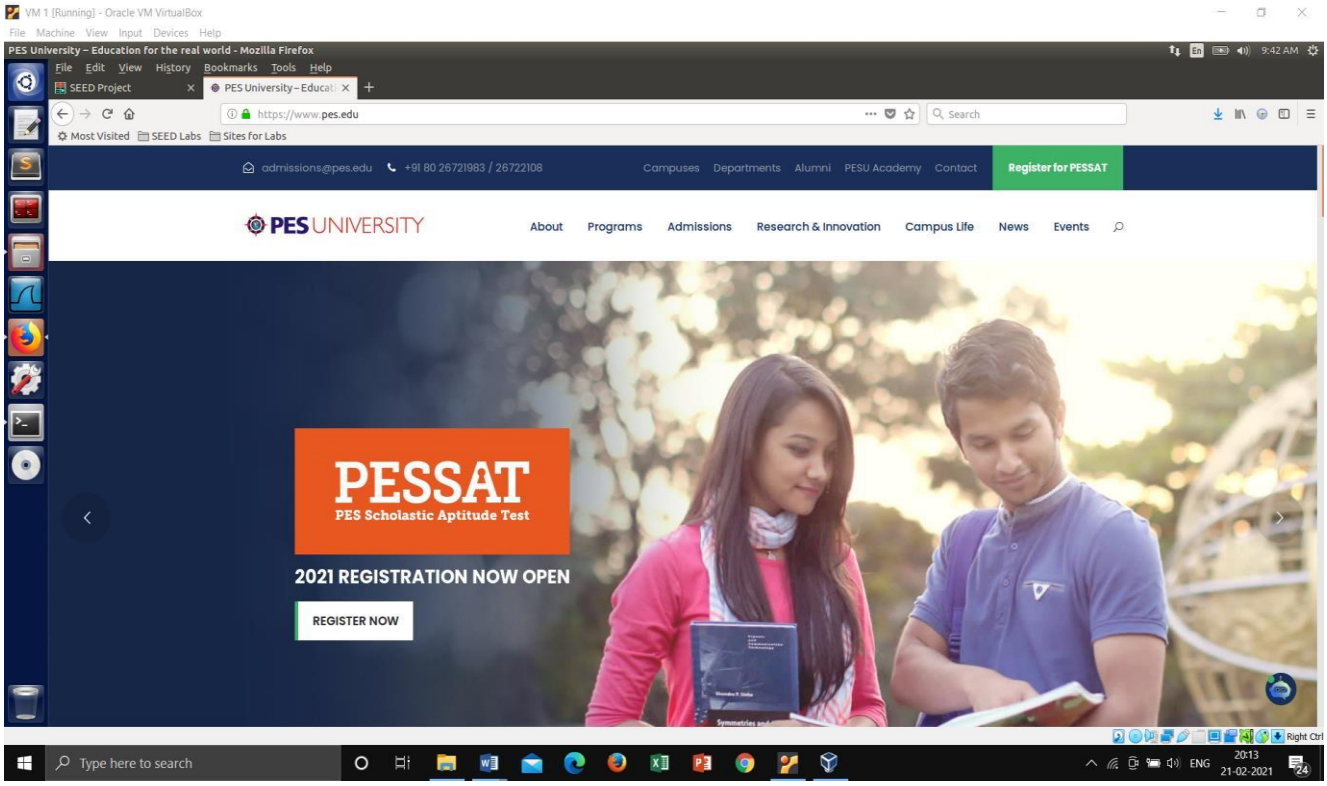
We next need to block VM1 from visiting a website. We use the www.pes.edu website. We first find its IP address.

Command: ping www.pes.edu



We also test whether the website is accessible to the browser.

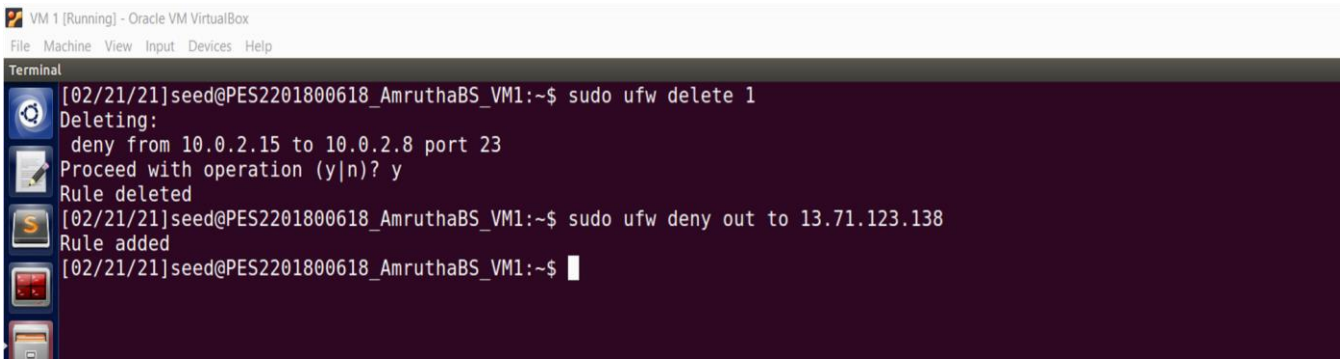
We are able to access the website.



We next add the firewall rule to prevent VM1 from accessing the IP address for www.pes.edu **Commands:**

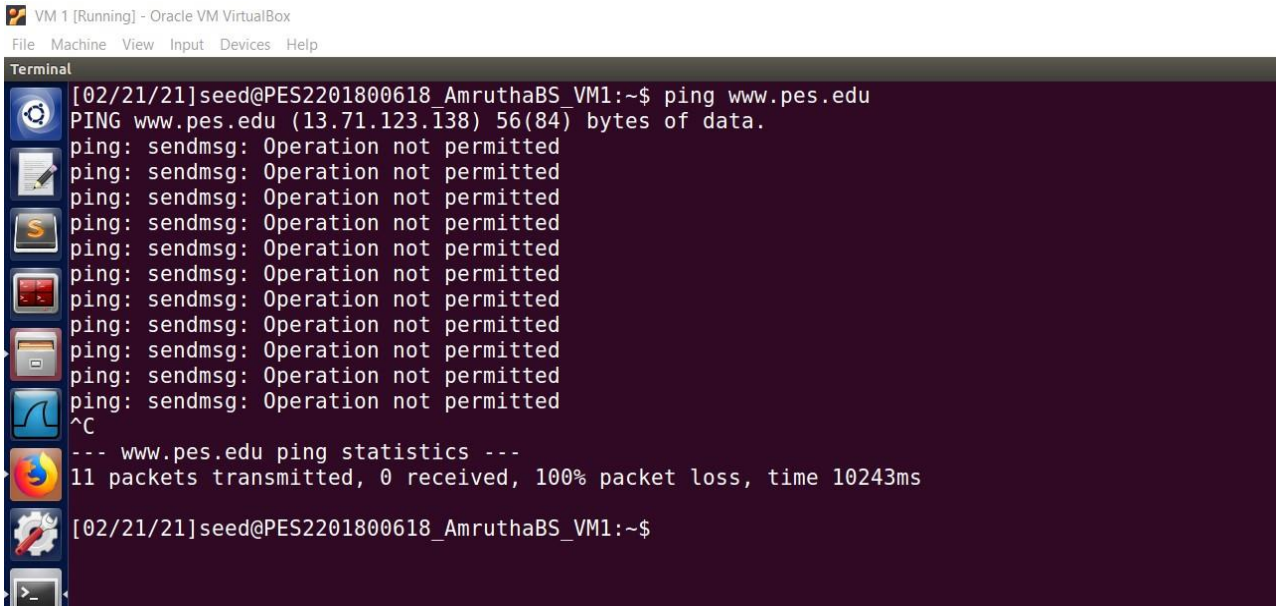
sudo ufw delete 1

sudo ufw deny out to 13.71.123.138

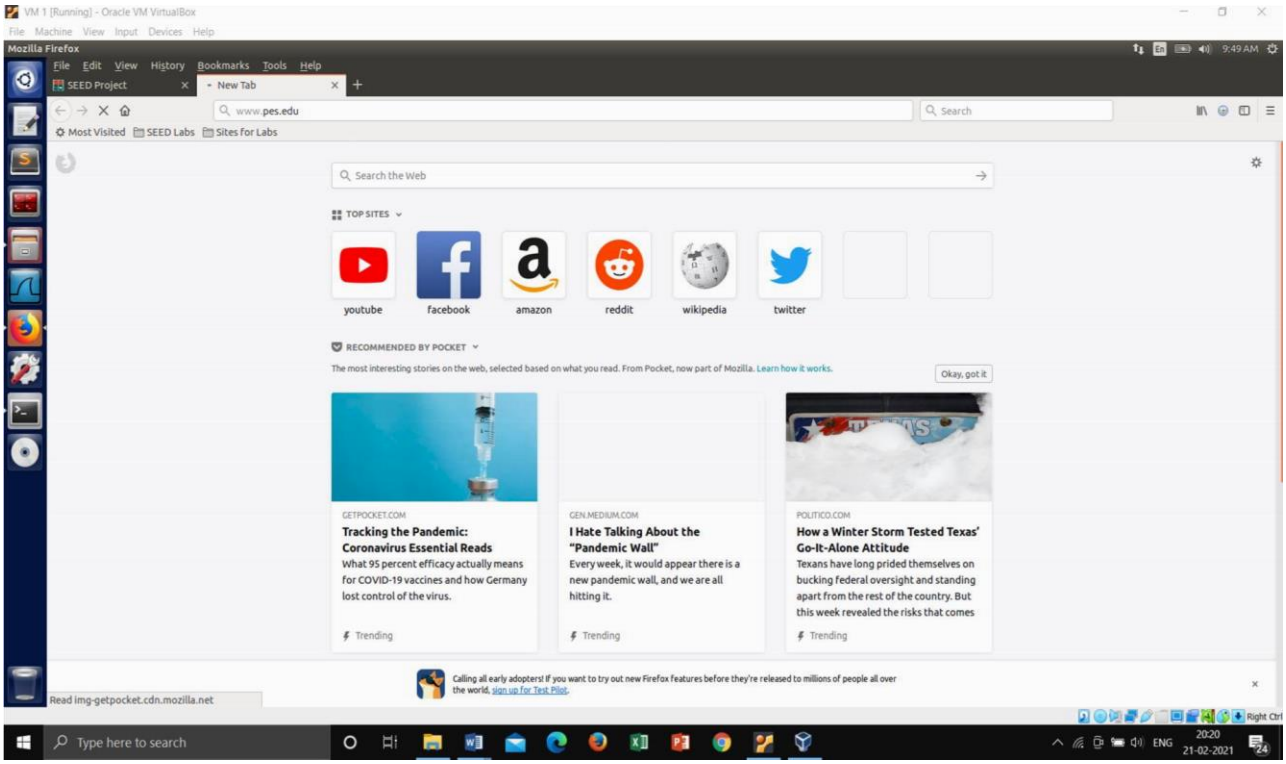


With the firewall rule in place, we next try to ping www.pes.edu. We see the message “Operation not permitted” because the firewall has blocked it.

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



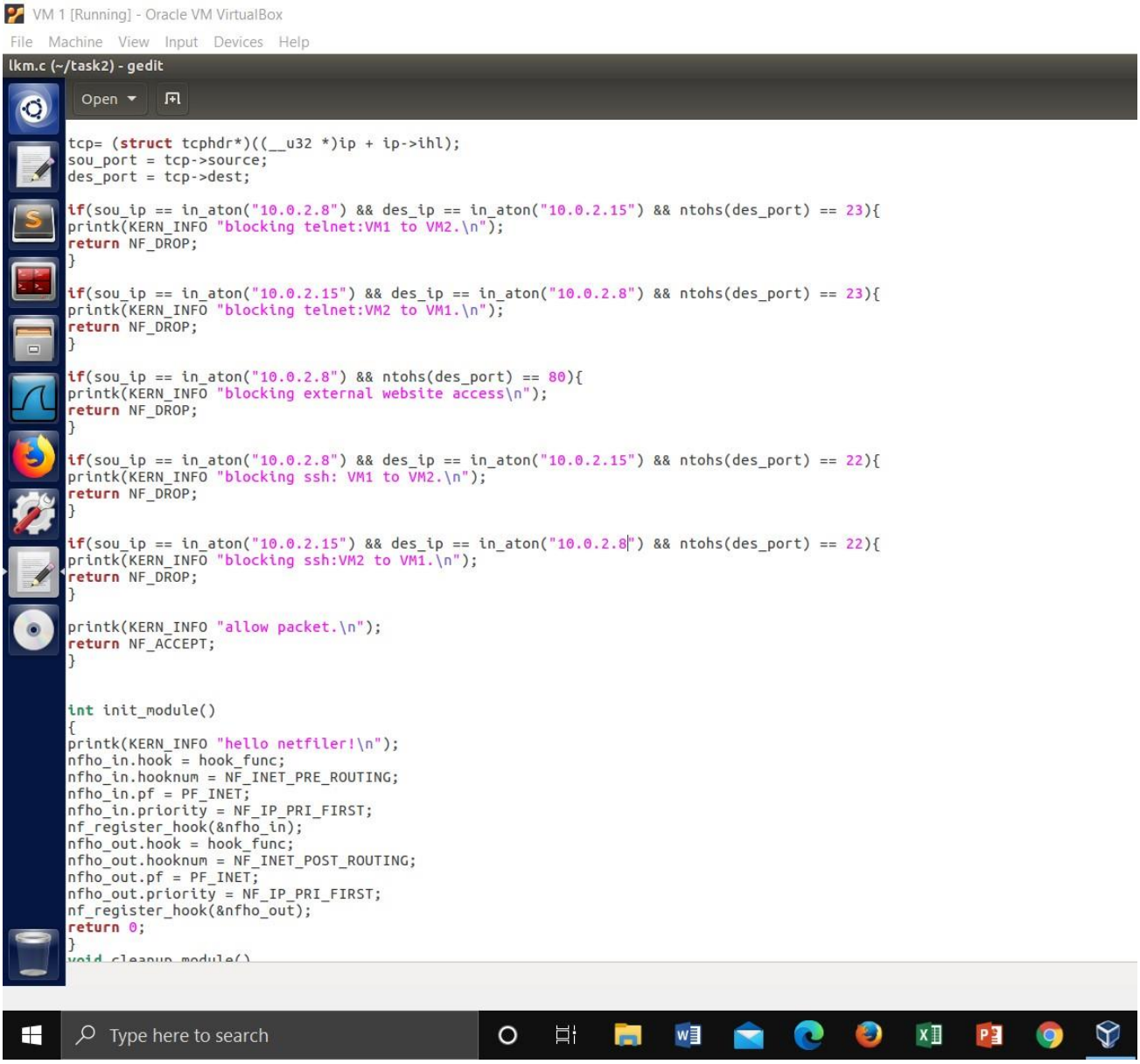
This can be verified by revisiting www.pes.edu in the Firefox browser. The page is not loaded.



Task 2: How Firewall Works

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

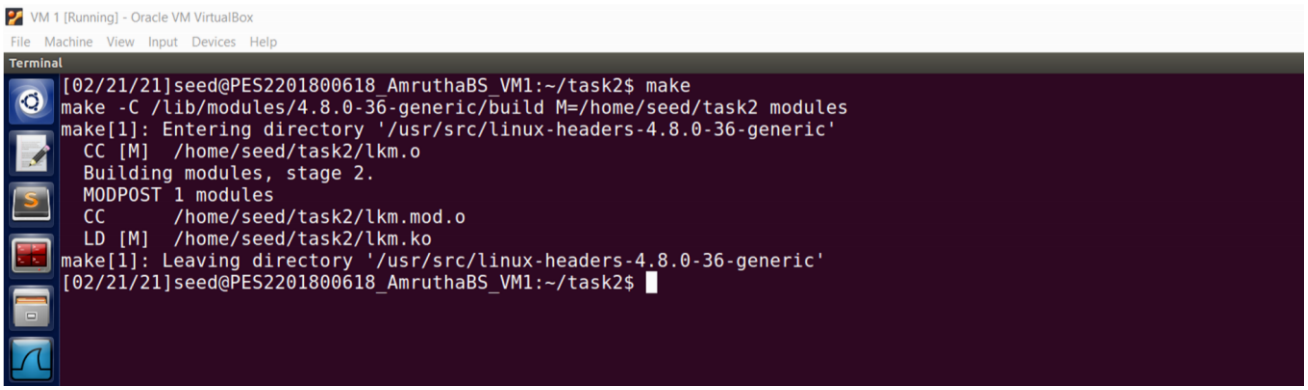
- Block telnet from VM1 to VM2
- Block telnet from VM2 to VM1
- Block external website access from VM1
- Block ssh from VM1 to VM2
- Block ssh from VM2 to VM1



Below is the code for the firewall: lkm.c

Makefile:

The program can be compiled using the make command.



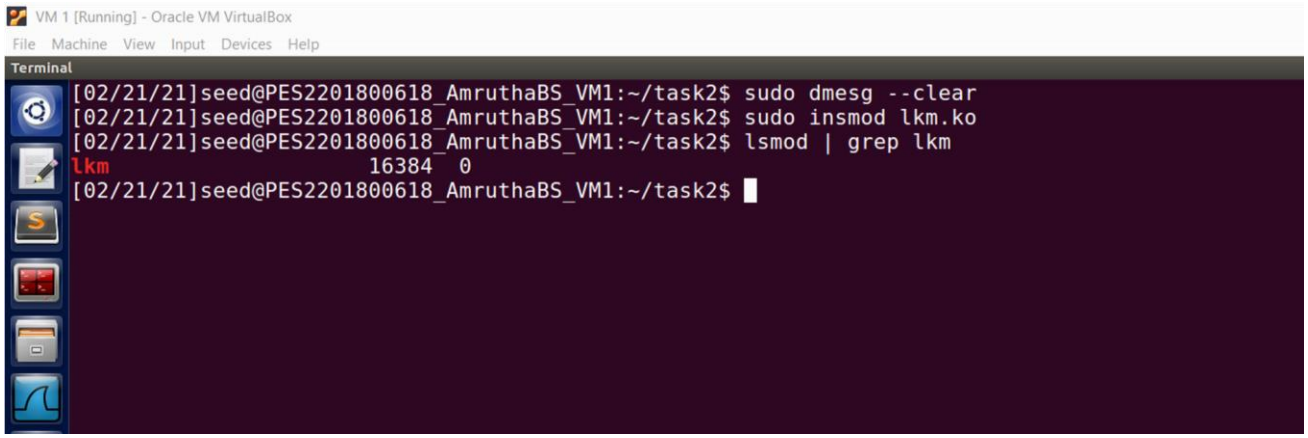
The compiled kernel module (lkm.ko) can be inserted using insmod:

Commands:

sudo dmesg --clear

sudo insmod lkm.ko

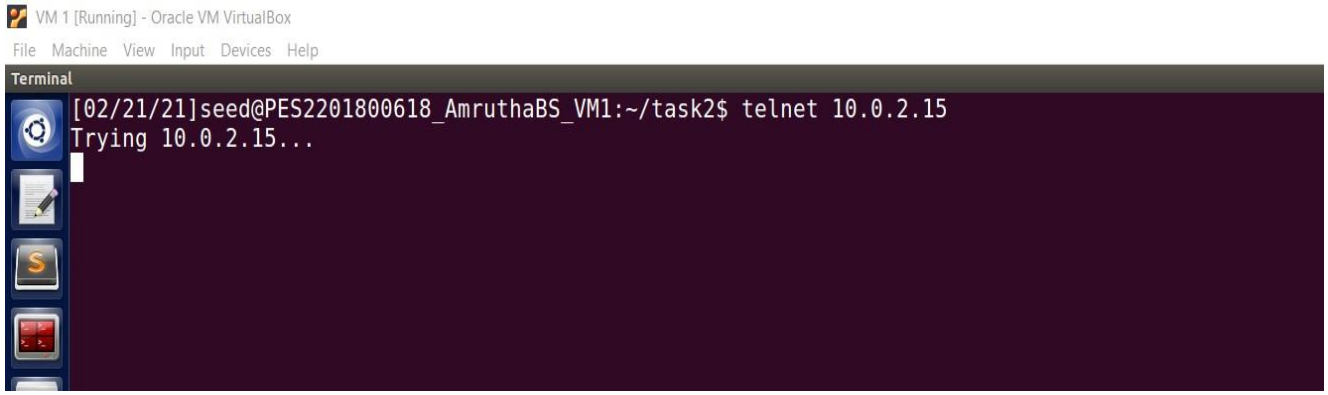
lsmod | grep lkm



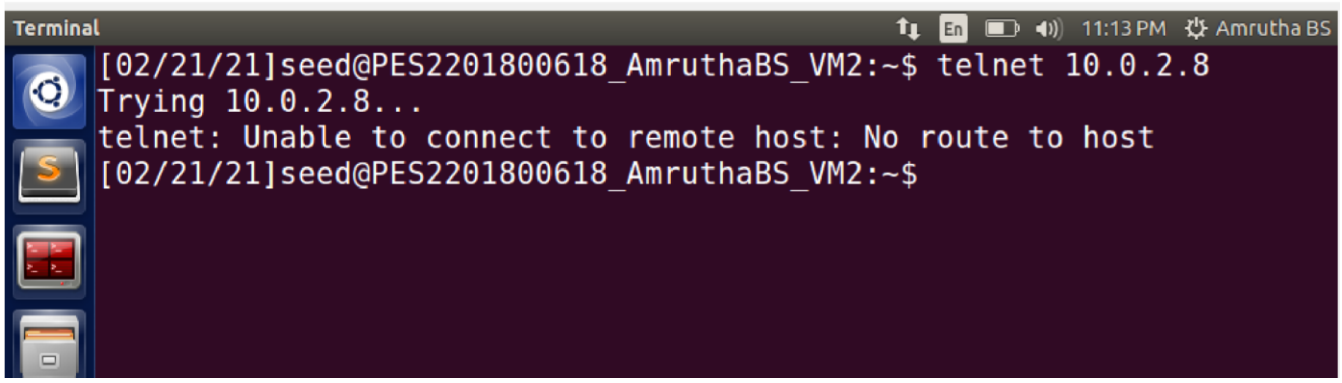
We first test whether telnet from VM1 to VM2 is blocked.

Command:

telnet 10.0.2.15



We next test whether telnet from VM2 to VM1 is blocked.

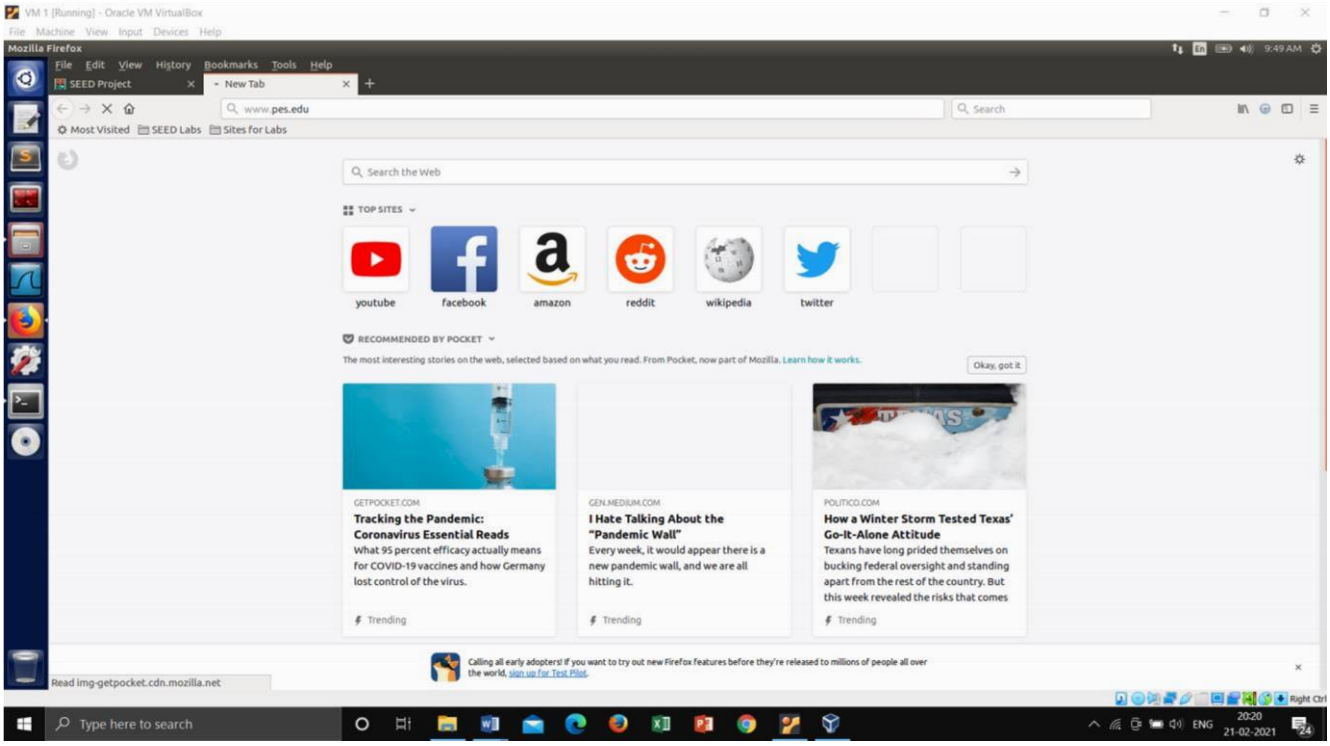


Packets being dropped can be seen here:

Commands:

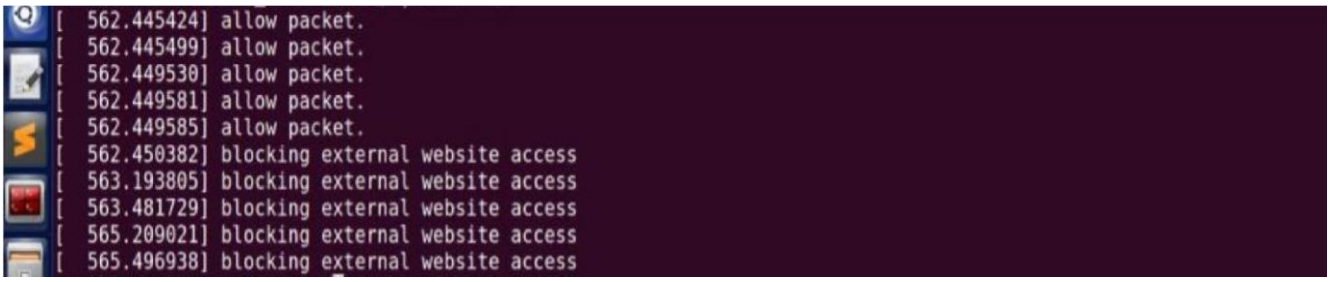
dmesg | tail -10 (This command will show u dropping packets)

We next test whether external website access (port 80) is allowed from VM1. We try to load the page www.pes.edu in the Firefox browser and the page does not load.

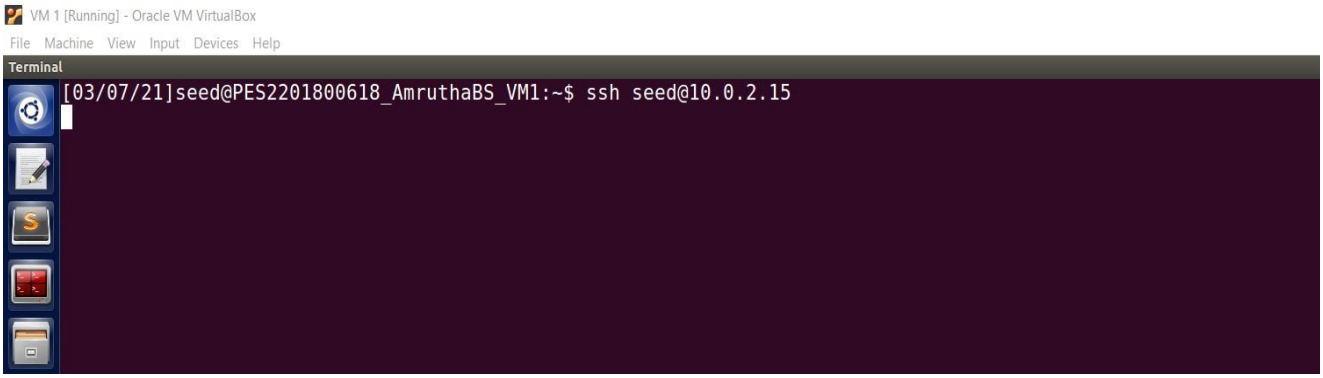


Command:

dmesg | tail -10

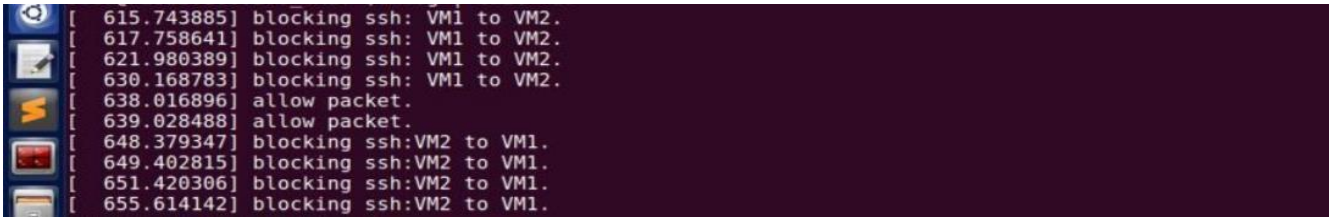


ssh is blocked from VM1 to VM2.



ssh is blocked from VM2 to VM1.

We can see the packets being dropped



Task 3: Evading Egress Filtering

In this task, we will be using a ssh tunnel to evade egress filtering. Please delete all the firewall rules from the previous task using the ufw delete <rule_number> command.

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

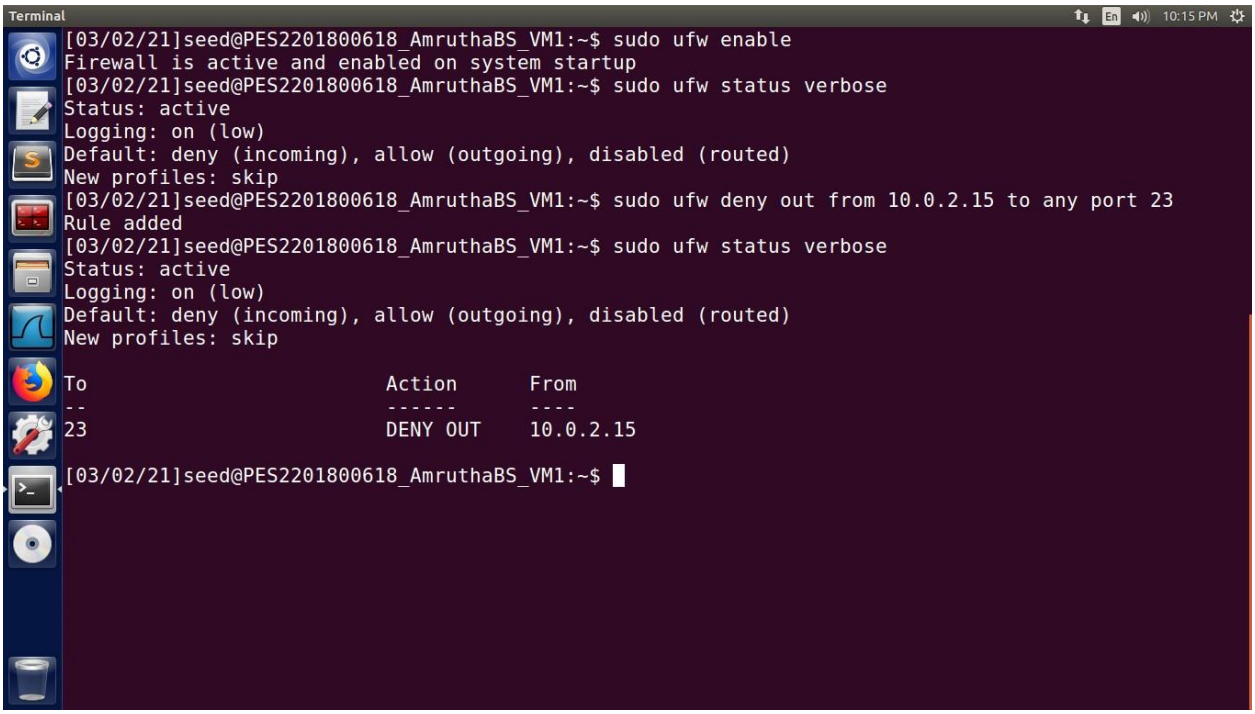
In this lab, we will use three VMs. VM1 will be blocked from being able to telnet to VM2. We will utilize a ssh tunnel to allow VM1 to telnet to VM3 via VM2. The diagram below depicts the tunnel (in the diagram, the home machine is VM1, the apollo machine is VM2 and the work machine is VM3).

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

Command:

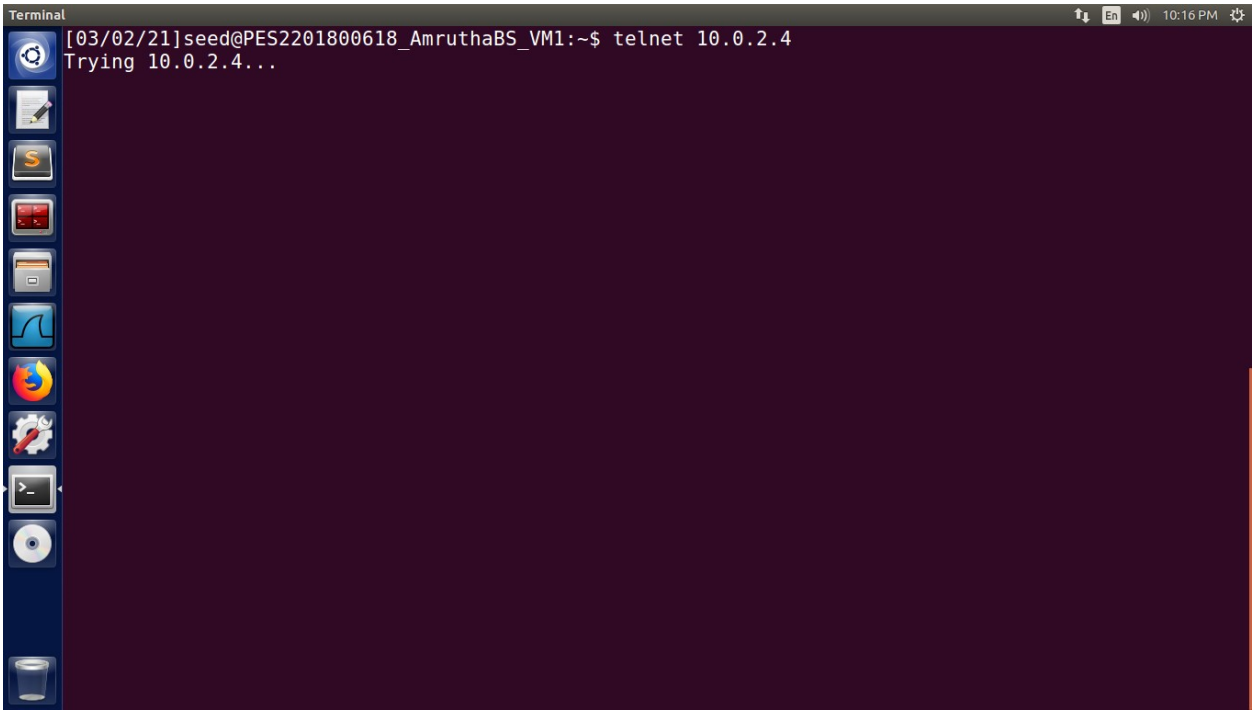
sudo ufw enable sudo
ufw status verbose

sudo ufw deny out from 10.0.2.15 to any port 23 sudo
ufw status verbose



The effect of the previous firewall rule can be observed below. The telnet is blocked. **Command:**

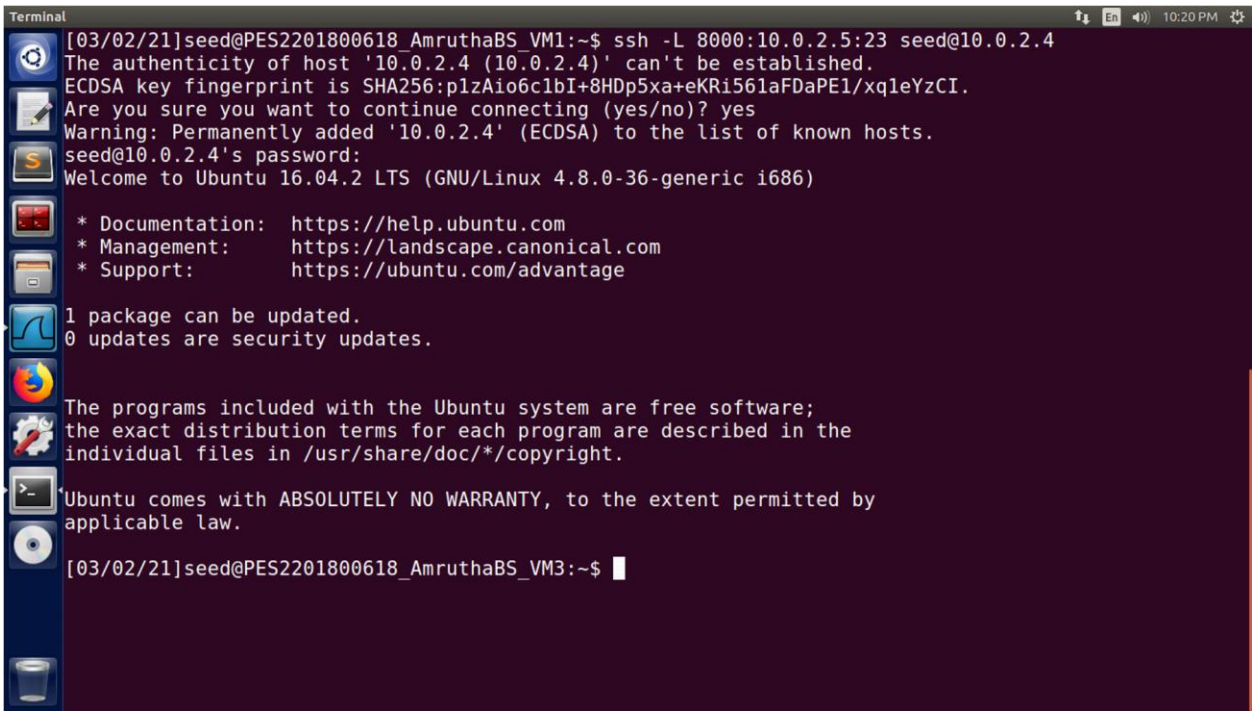
telnet 10.0.2.4



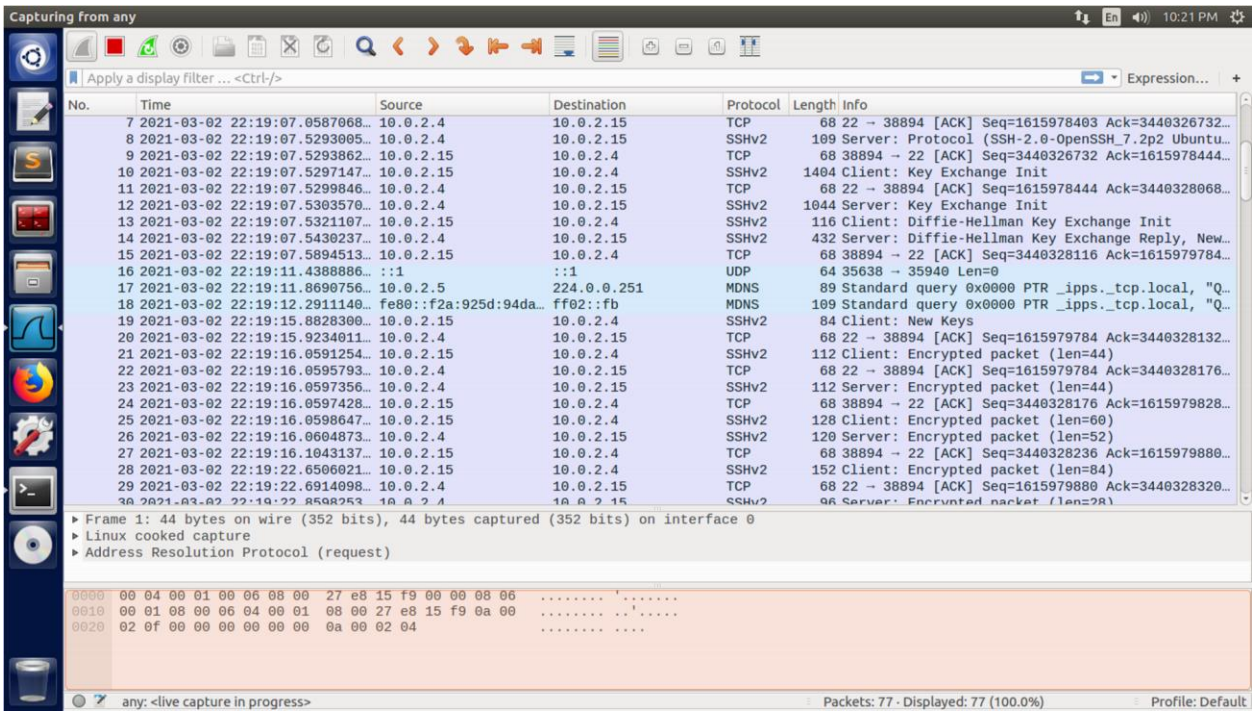
We next setup a ssh tunnel between VM1 and VM2 to allow VM1 to telnet via VM2, evading the firewall on VM1. The ssh command below allows VM1 to use its local port 8000 to telnet to VM3 via VM2.

Command:

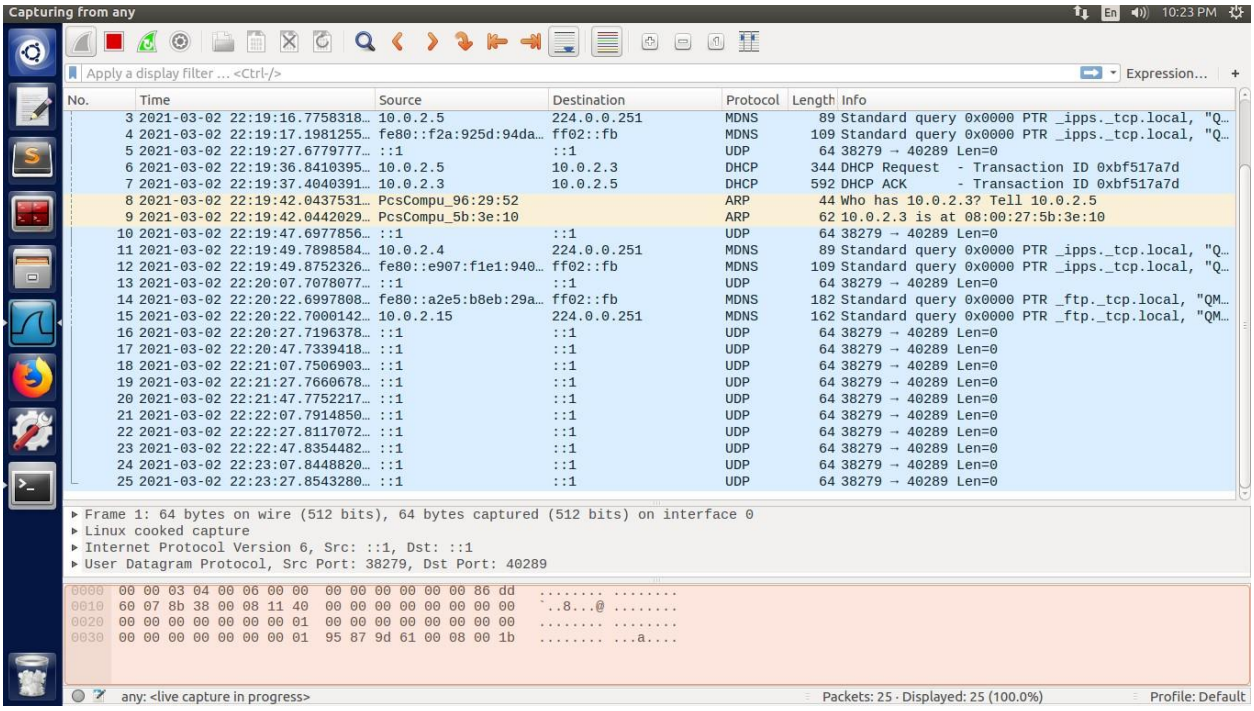
ssh -L 8000:10.0.2.5:23 seed@10.0.2.4



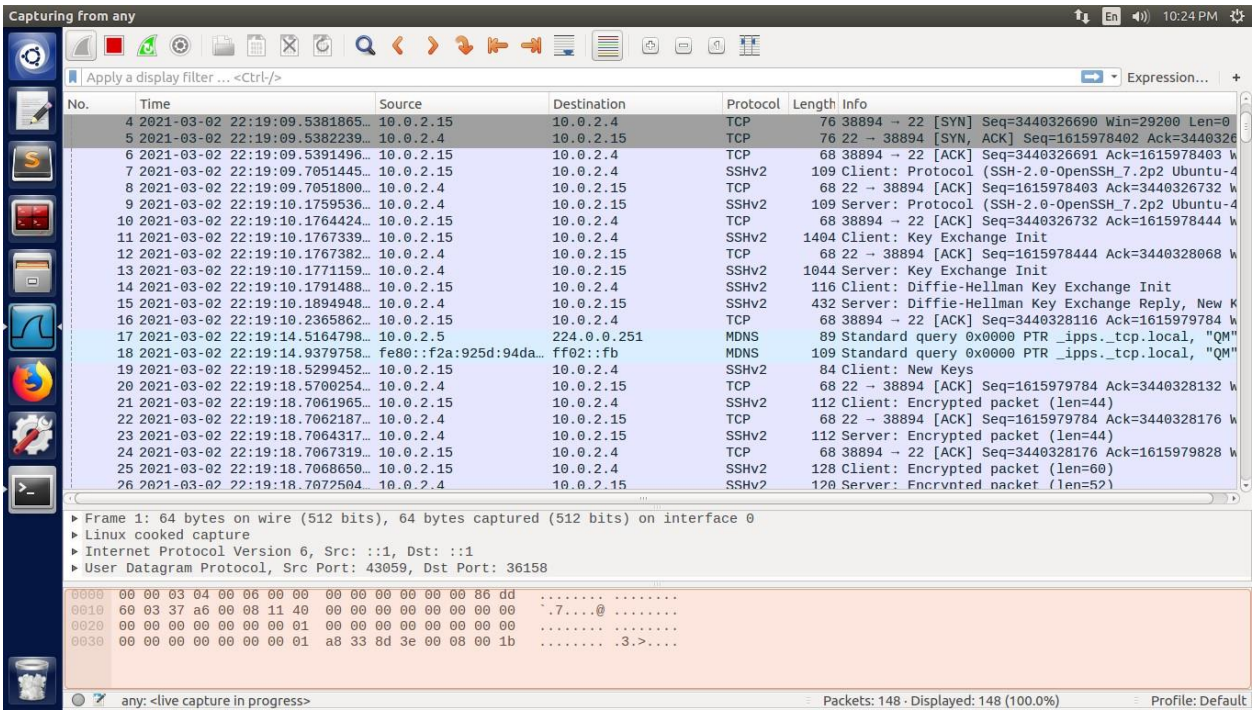
VM1 Wireshark capture.



VM2 Wireshark capture.



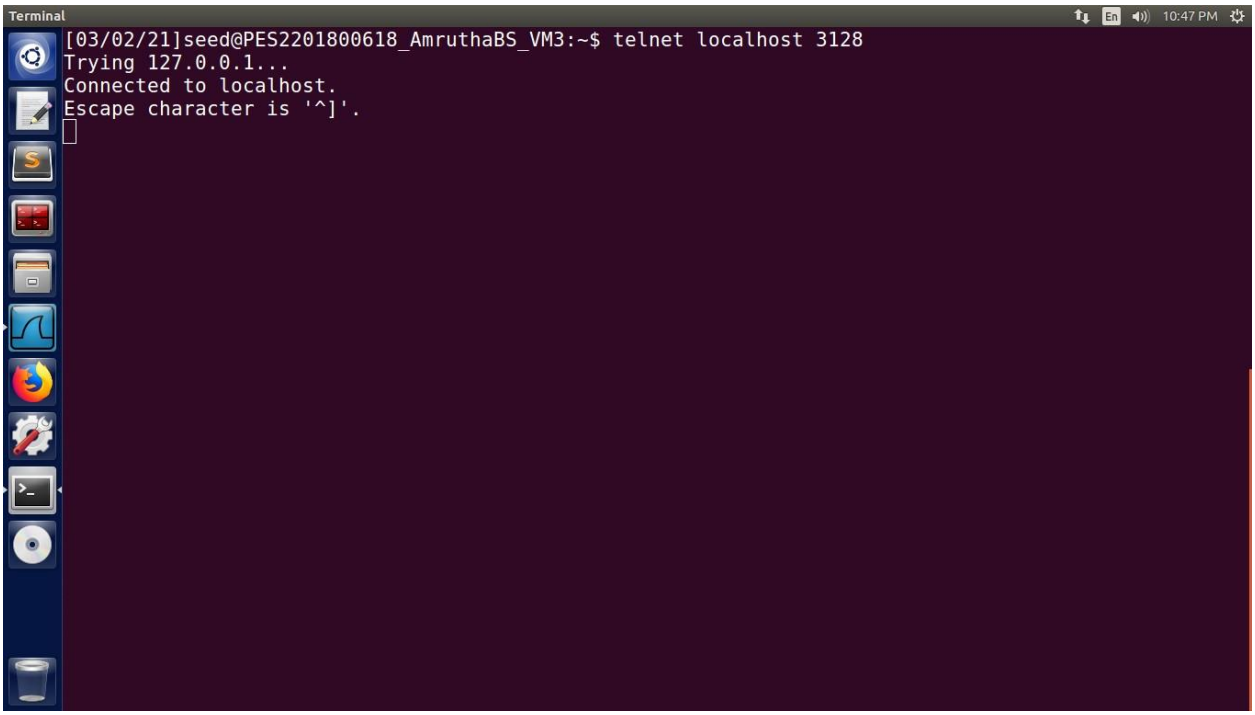
VM3 Wireshark capture.



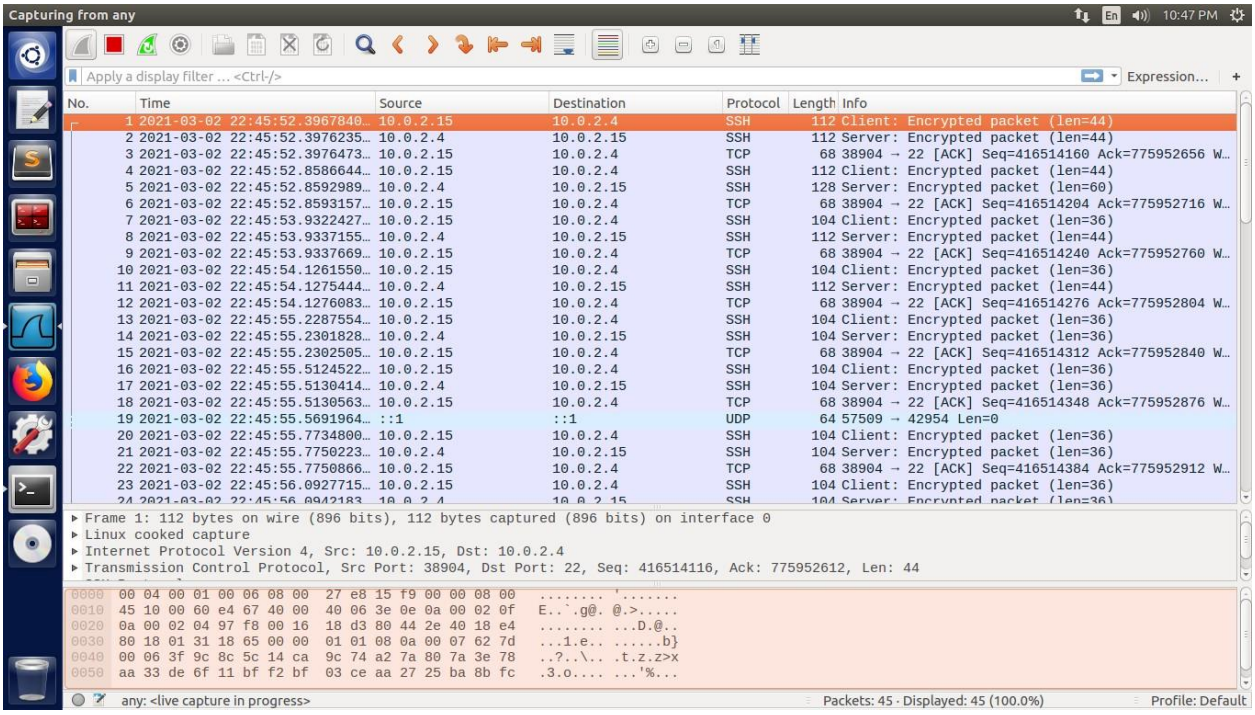
With the ssh tunnel setup, we can now telnet from VM1 to VM3 even though the firewall policy on VM1 denies outgoing telnet.

Command:

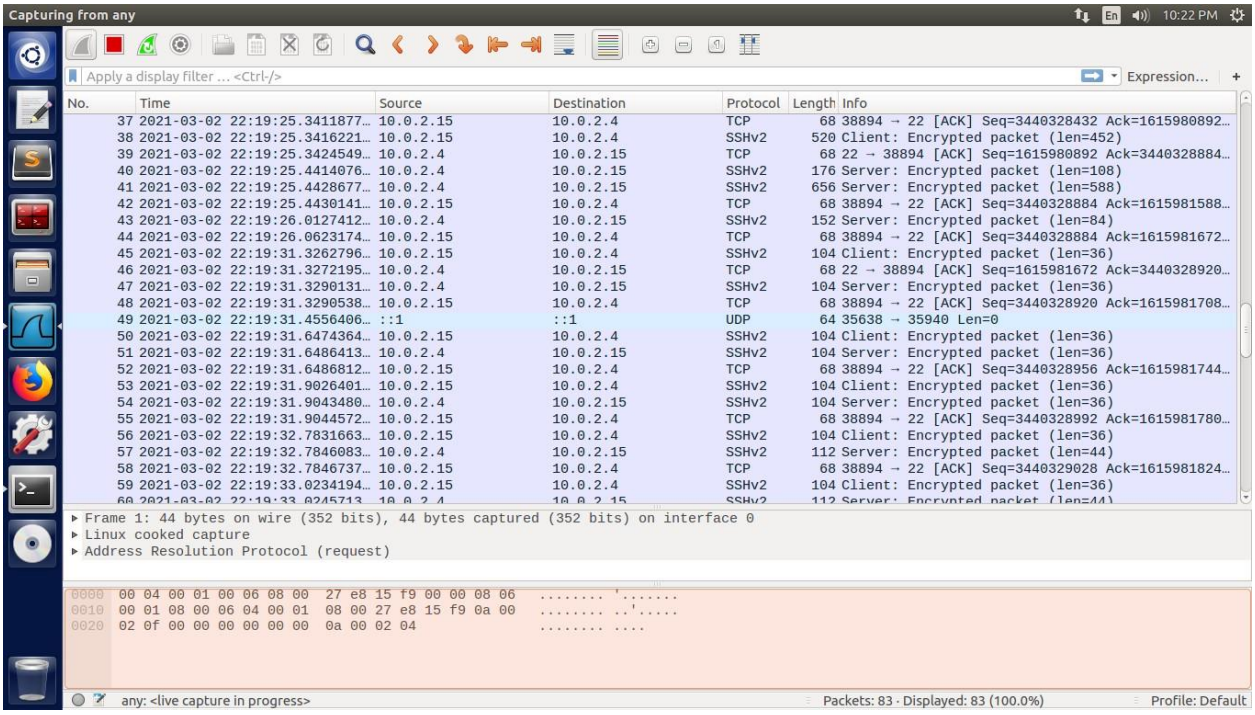
telnet localhost 3128



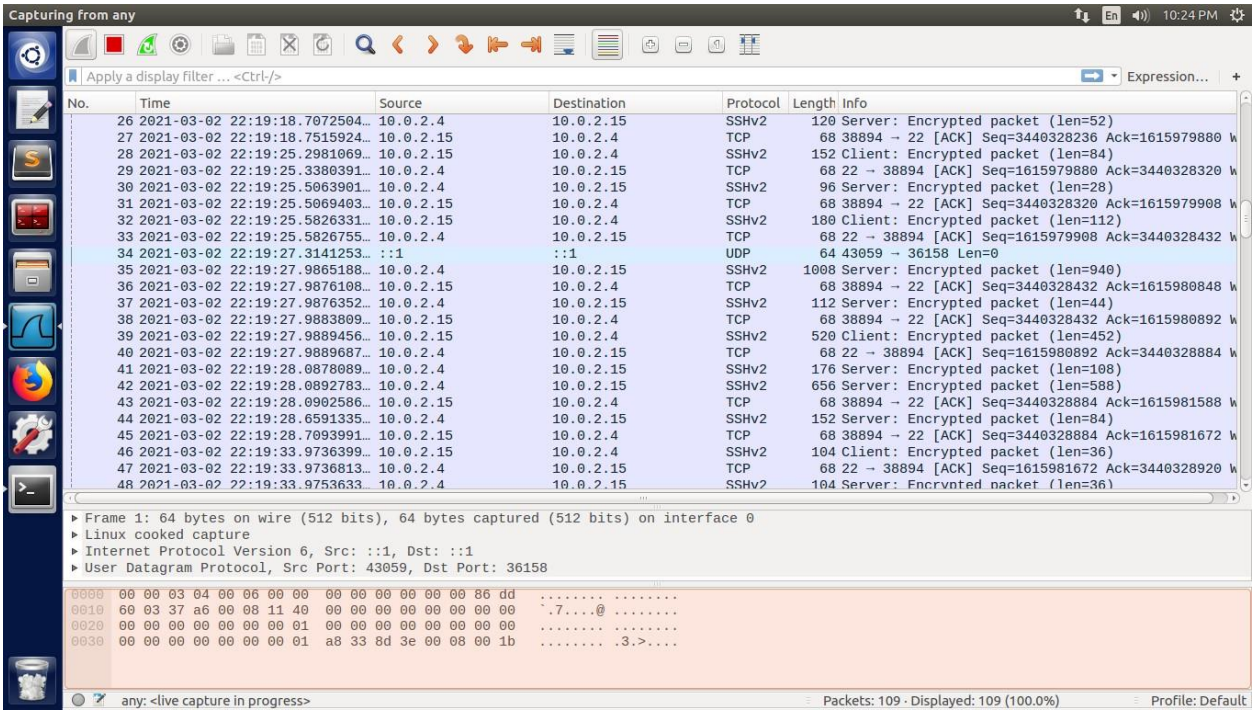
VM1 Wireshark capture.

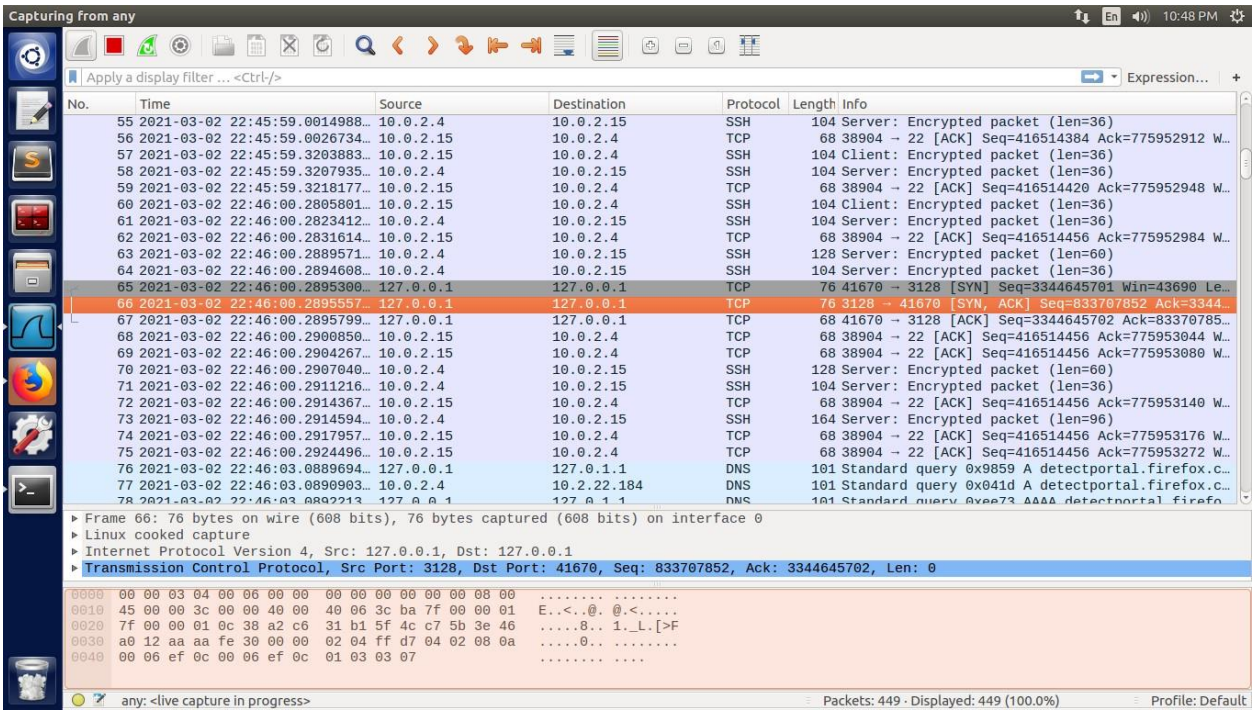


VM2 Wireshark capture.



VM3 Wireshark capture.


Capturing from any
10:24 PM
Apply a display filter … <Ctrl-/>
Expression…
No. Time Source Destination Protocol Length Info
26 2021-03-02 22:19:18.7072504… 10.0.2.4 10.0.2.15 SSHv2 120 Server: Encrypted packet (len=52)
27 2021-03-02 22:19:18.7515924… 10.0.2.15 10.0.2.4 TCP 68 38894 → 22 [ACK] Seq=3440328236 Ack=1615979880 W
28 2021-03-02 22:19:25.2981069… 10.0.2.15 10.0.2.4 SSHv2 152 Client: Encrypted packet (len=84)
29 2021-03-02 22:19:25.3380391… 10.0.2.4 10.0.2.15 TCP 68 22 → 38894 [ACK] Seq=1615979880 Ack=3440328320 W
30 2021-03-02 22:19:25.5063901… 10.0.2.4 10.0.2.15 SSHv2 96 Server: Encrypted packet (len=28)
31 2021-03-02 22:19:25.5069403… 10.0.2.15 10.0.2.4 TCP 68 38894 → 22 [ACK] Seq=3440328320 Ack=1615979908 W
32 2021-03-02 22:19:25.5826331… 10.0.2.15 10.0.2.4 SSHv2 180 Client: Encrypted packet (len=112)
33 2021-03-02 22:19:25.5826755… 10.0.2.4 10.0.2.15 TCP 68 22 → 38894 [ACK] Seq=1615979908 Ack=3440328432 W
34 2021-03-02 22:19:27.3141253… ::1 ::1 UDP 64 43059 → 36158 Len=0
35 2021-03-02 22:19:27.9865188… 10.0.2.4 10.0.2.15 SSHv2 1008 Server: Encrypted packet (len=940)
36 2021-03-02 22:19:27.9876108… 10.0.2.15 10.0.2.4 TCP 68 38894 → 22 [ACK] Seq=3440328432 Ack=1615980848 W
37 2021-03-02 22:19:27.9876352… 10.0.2.4 10.0.2.15 SSHv2 112 Server: Encrypted packet (len=44)
38 2021-03-02 22:19:27.9883809… 10.0.2.15 10.0.2.4 TCP 68 38894 → 22 [ACK] Seq=3440328432 Ack=1615980892 W
39 2021-03-02 22:19:27.9889456… 10.0.2.15 10.0.2.4 SSHv2 520 Client: Encrypted packet (len=452)
40 2021-03-02 22:19:27.9889687… 10.0.2.4 10.0.2.15 TCP 68 22 → 38894 [ACK] Seq=1615980892 Ack=3440328884 W
41 2021-03-02 22:19:28.0878089… 10.0.2.4 10.0.2.15 SSHv2 176 Server: Encrypted packet (len=108)
42 2021-03-02 22:19:28.0892783… 10.0.2.4 10.0.2.15 SSHv2 656 Server: Encrypted packet (len=588)
43 2021-03-02 22:19:28.0902586… 10.0.2.15 10.0.2.4 TCP 68 38894 → 22 [ACK] Seq=3440328884 Ack=1615981588 W
44 2021-03-02 22:19:28.6591335… 10.0.2.4 10.0.2.15 SSHv2 152 Server: Encrypted packet (len=84)
45 2021-03-02 22:19:28.7093991… 10.0.2.15 10.0.2.4 TCP 68 38894 → 22 [ACK] Seq=3440328884 Ack=1615981672 W
46 2021-03-02 22:19:33.9736399… 10.0.2.15 10.0.2.4 SSHv2 104 Client: Encrypted packet (len=36)
47 2021-03-02 22:19:33.9736813… 10.0.2.4 10.0.2.15 TCP 68 22 → 38894 [ACK] Seq=1615981672 Ack=3440328920 W
48 2021-03-02 22:19:33.9753633… 10.0.2.4 10.0.2.15 SSHv2 104 Server: Encrypted packet (len=36)
▸ 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: 43059, Dst Port: 36158
0000 00 00 03 04 00 06 00 00 00 00 00 00 00 00 86 dd ....
0010 60 03 37 a6 00 08 11 40 00 00 00 00 00 00 00 00 `.7....@ ....
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 a8 33 8d 3e 00 08 00 1b .... .3.>....
any: <live capture in progress>
Packets: 109 · Displayed: 109 (100.0%)
Profile: Default


Capturing from any
10:48 PM
Apply a display filter … <Ctrl-/>
Expression…
No. Time Source Destination Protocol Length Info
55 2021-03-02 22:45:59.0014988… 10.0.2.4 10.0.2.15 SSH 104 Server: Encrypted packet (len=36)
56 2021-03-02 22:45:59.0026734… 10.0.2.15 10.0.2.4 TCP 68 38904 → 22 [ACK] Seq=416514384 Ack=775952912 W…
57 2021-03-02 22:45:59.3203883… 10.0.2.15 10.0.2.4 SSH 104 Client: Encrypted packet (len=36)
58 2021-03-02 22:45:59.3207935… 10.0.2.4 10.0.2.15 SSH 104 Server: Encrypted packet (len=36)
59 2021-03-02 22:45:59.3218177… 10.0.2.15 10.0.2.4 TCP 68 38904 → 22 [ACK] Seq=416514420 Ack=775952948 W…
60 2021-03-02 22:46:00.2805801… 10.0.2.15 10.0.2.4 SSH 104 Client: Encrypted packet (len=36)
61 2021-03-02 22:46:00.2823412… 10.0.2.4 10.0.2.15 SSH 104 Server: Encrypted packet (len=36)
62 2021-03-02 22:46:00.2831614… 10.0.2.15 10.0.2.4 TCP 68 38904 → 22 [ACK] Seq=416514456 Ack=775952984 W…
63 2021-03-02 22:46:00.2889571… 10.0.2.4 10.0.2.15 SSH 128 Server: Encrypted packet (len=60)
64 2021-03-02 22:46:00.2894608… 10.0.2.4 10.0.2.15 SSH 104 Server: Encrypted packet (len=36)
65 2021-03-02 22:46:00.2895300… 127.0.0.1 127.0.0.1 TCP 76 41670 → 3128 [SYN] Seq=3344645701 Win=43690 Le…
66 2021-03-02 22:46:00.2895557… 127.0.0.1 127.0.0.1 TCP 76 3128 → 41670 [SYN, ACK] Seq=833707852 Ack=3344…
67 2021-03-02 22:46:00.2895799… 127.0.0.1 127.0.0.1 TCP 68 41670 → 3128 [ACK] Seq=3344645702 Ack=83370785…
68 2021-03-02 22:46:00.2900850… 10.0.2.15 10.0.2.4 TCP 68 38904 → 22 [ACK] Seq=416514456 Ack=775953044 W…
69 2021-03-02 22:46:00.2904267… 10.0.2.15 10.0.2.4 TCP 68 38904 → 22 [ACK] Seq=416514456 Ack=775953080 W…
70 2021-03-02 22:46:00.2907040… 10.0.2.4 10.0.2.15 SSH 128 Server: Encrypted packet (len=60)
71 2021-03-02 22:46:00.2911216… 10.0.2.4 10.0.2.15 SSH 104 Server: Encrypted packet (len=36)
72 2021-03-02 22:46:00.2914367… 10.0.2.15 10.0.2.4 TCP 68 38904 → 22 [ACK] Seq=416514456 Ack=775953140 W…
73 2021-03-02 22:46:00.2914594… 10.0.2.4 10.0.2.15 SSH 164 Server: Encrypted packet (len=96)
74 2021-03-02 22:46:00.2917957… 10.0.2.15 10.0.2.4 TCP 68 38904 → 22 [ACK] Seq=416514456 Ack=775953176 W…
75 2021-03-02 22:46:00.2924496… 10.0.2.15 10.0.2.4 TCP 68 38904 → 22 [ACK] Seq=416514456 Ack=775953272 W…
76 2021-03-02 22:46:03.0889694… 127.0.0.1 127.0.1.1 DNS 101 Standard query 0x9859 A detectportal.firefox.c…
77 2021-03-02 22:46:03.0890903… 10.0.2.4 10.2.22.184 DNS 101 Standard query 0x041d A detectportal.firefox.c…
▸ Frame 66: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 3128, Dst Port: 41670, Seq: 833707852, Ack: 3344645702, Len: 0
0000 00 00 03 04 00 06 00 00 00 00 00 00 08 00 ....
0010 45 00 00 3c 00 00 40 00 40 06 3c ba 7f 00 00 01 E..<..@. @.<.....
0020 7f 00 00 01 0c 38 a2 c6 31 b1 5f 4c c7 5b 3e 46 .....8.. 1._L.[>F
0030 a0 12 aa aa fe 30 00 00 02 04 ff d7 04 02 08 0a .....0.. ....
0040 00 06 ef 0c 00 06 ef 0c 01 03 03 07 ....
any: <live capture in progress>
Packets: 449 · Displayed: 449 (100.0%)
Profile: Default

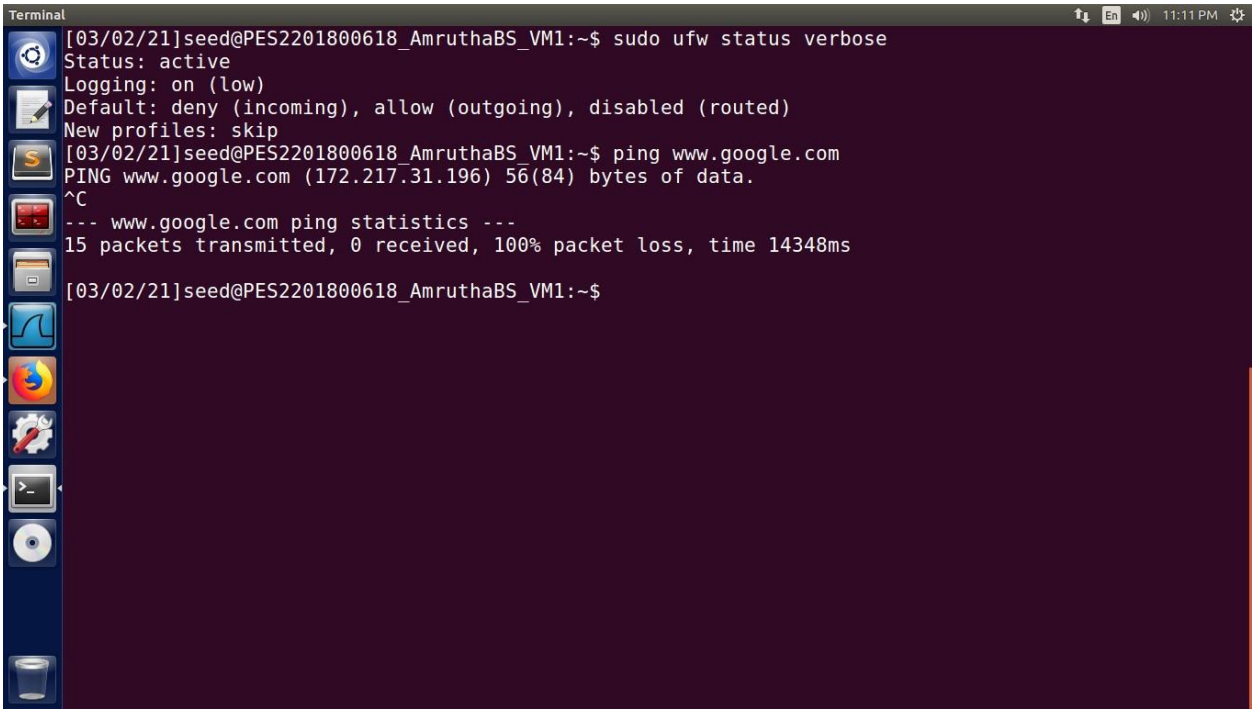
Task 3.b: Connecting to Google using SSH tunnel

In this task, we will setup a firewall rule to block VM1 from visiting www.google.com but then leverage dynamic forwarding via ssh tunnel to visit www.google.com from VM1 via VM2.

We first find the IP address of www.google.com a shown below:

Commands:

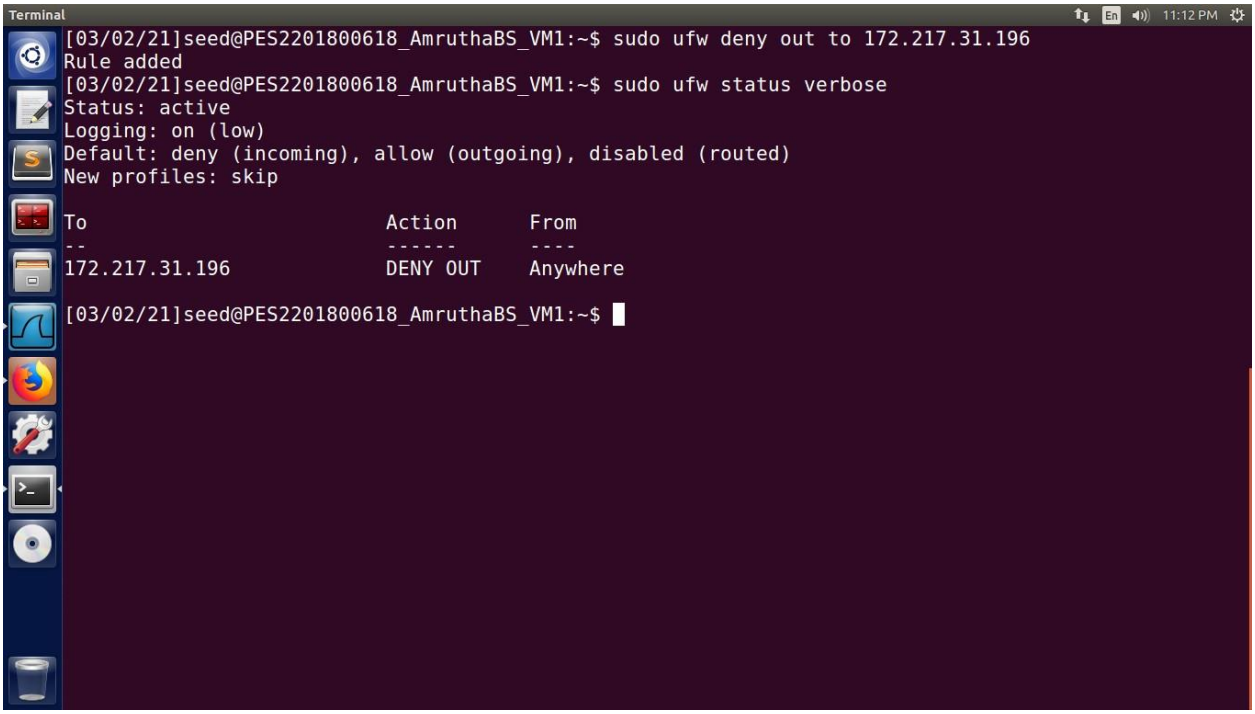
sudo ufw status verbose ping www.google.com



Given the IP address, we can setup a firewall rule to block traffic to that IP.

Command:

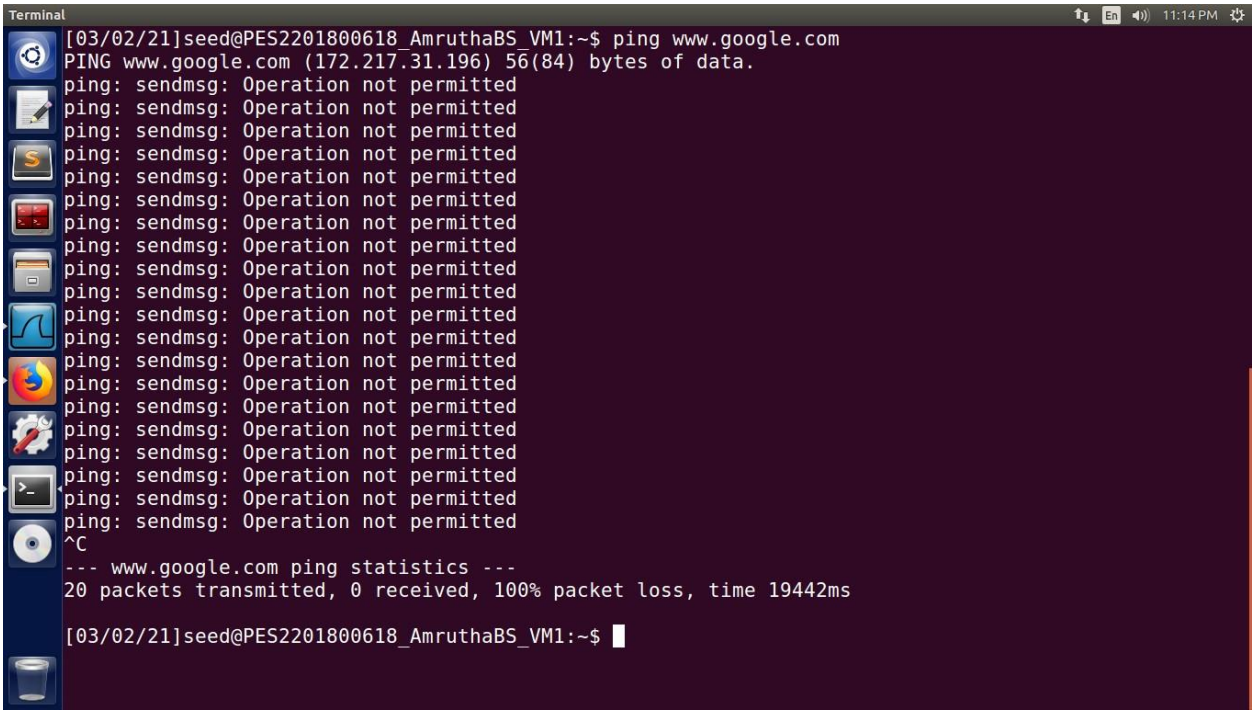
sudo ufw deny out to 172.217.31.196 sudo ufw status verbose



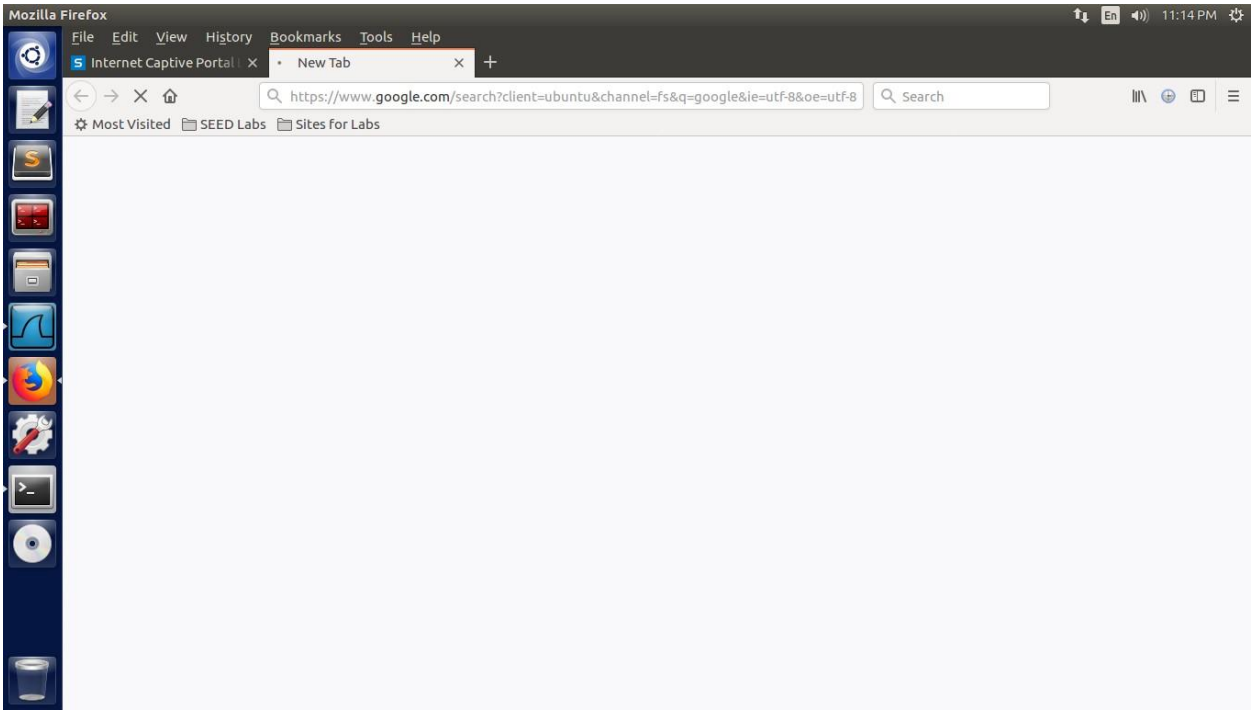
With the firewall rule in place, we can try to ping www.google.com. The operation is not permitted because it is being blocked by the firewall.

Command:

ping www.google.com



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

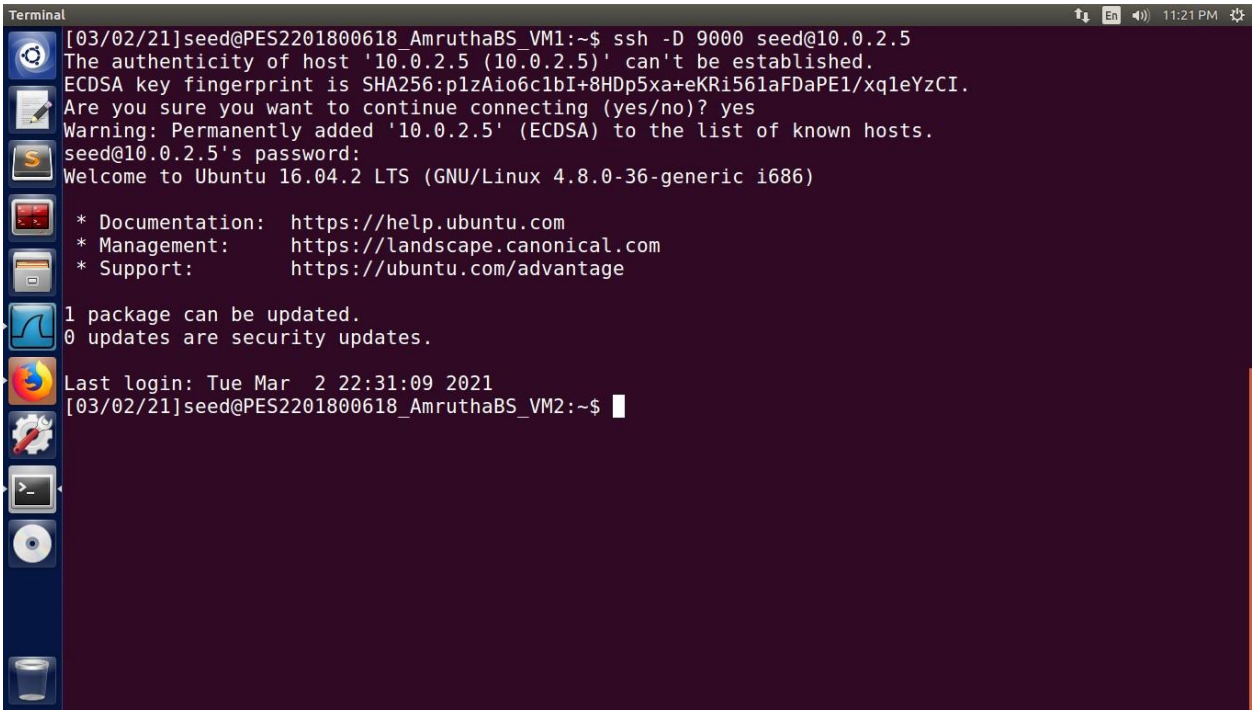


.

We now setup a ssh tunnel with dynamic port forwarding between VM1 and VM2. With this tunnel setup, VM1 will be able to use its local port 9000 to send a request to www.google.com via VM2.

Command:

ssh -D 9000 seed@10.0.2.11

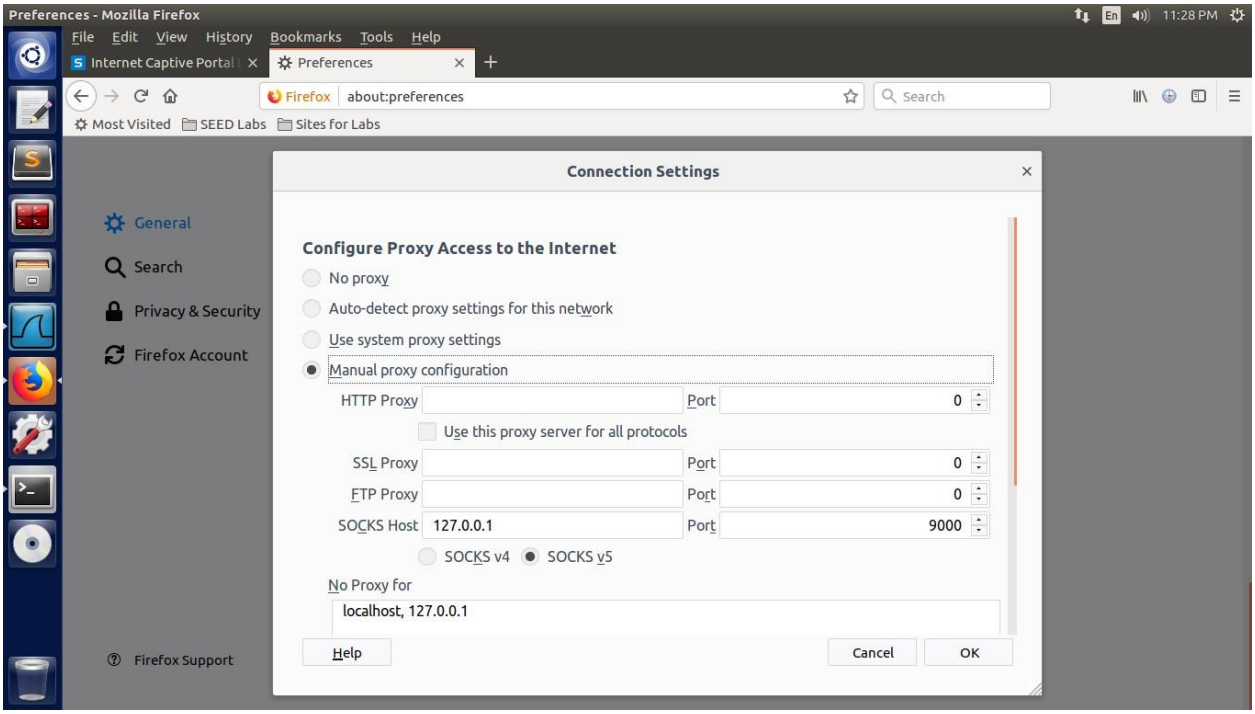


To use the established tunnel, we need to set the proxy settings in the firefox browser as shown below. With the proxy settings in place, we are able to visit www.google.com from VM1 even though the firewall has a policy to block it.

We next disable the tunnel to see its effect.

Command:

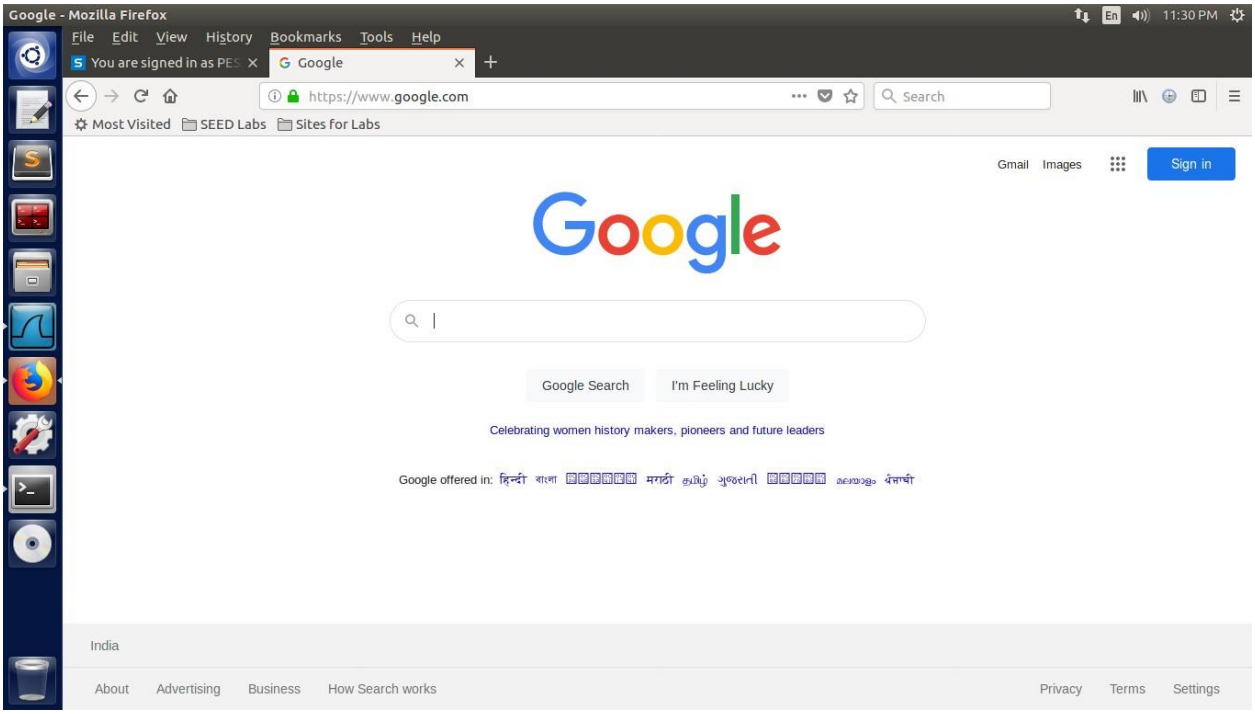
ssh -D 9000 seed@10.0.2.11



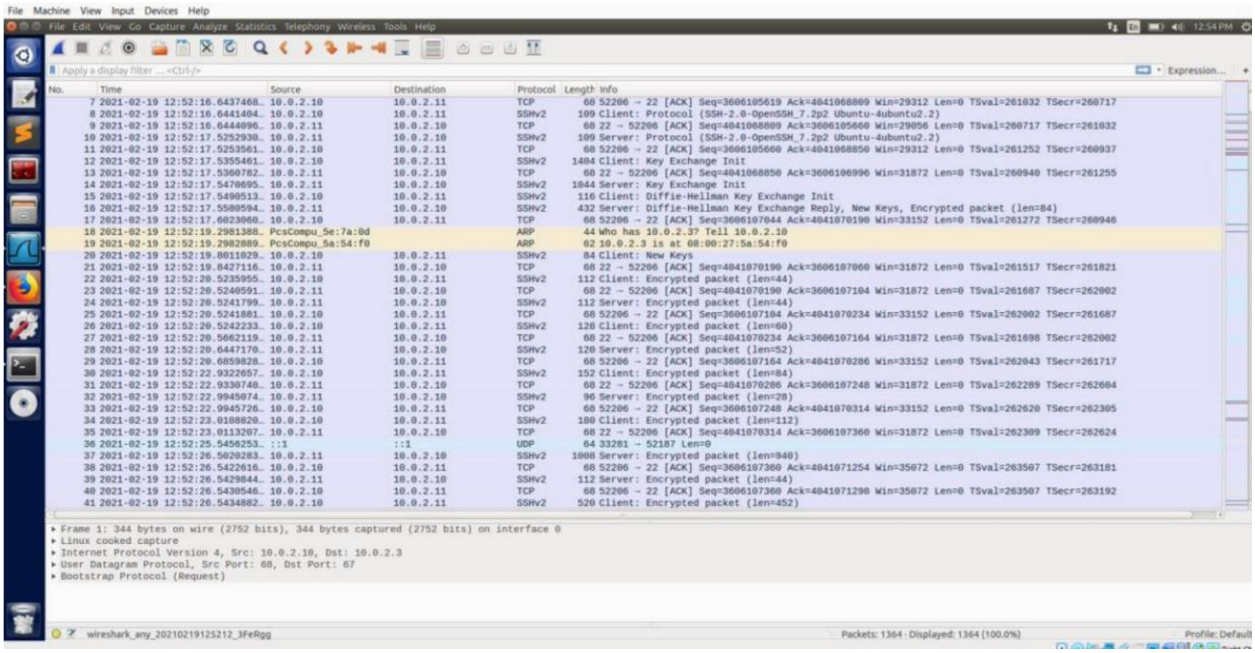
We also clear the browser cache to make sure firefox does not show the webpage it just loaded.

Command:

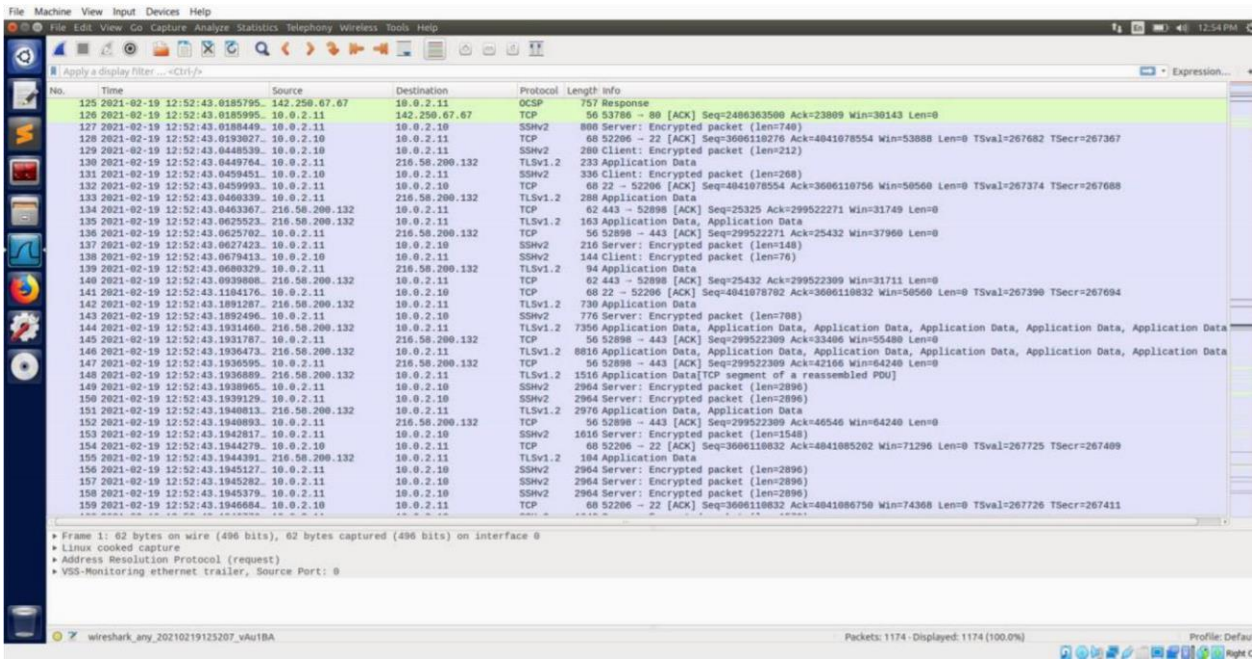
ssh -D 9000 seed@10.0.2.11



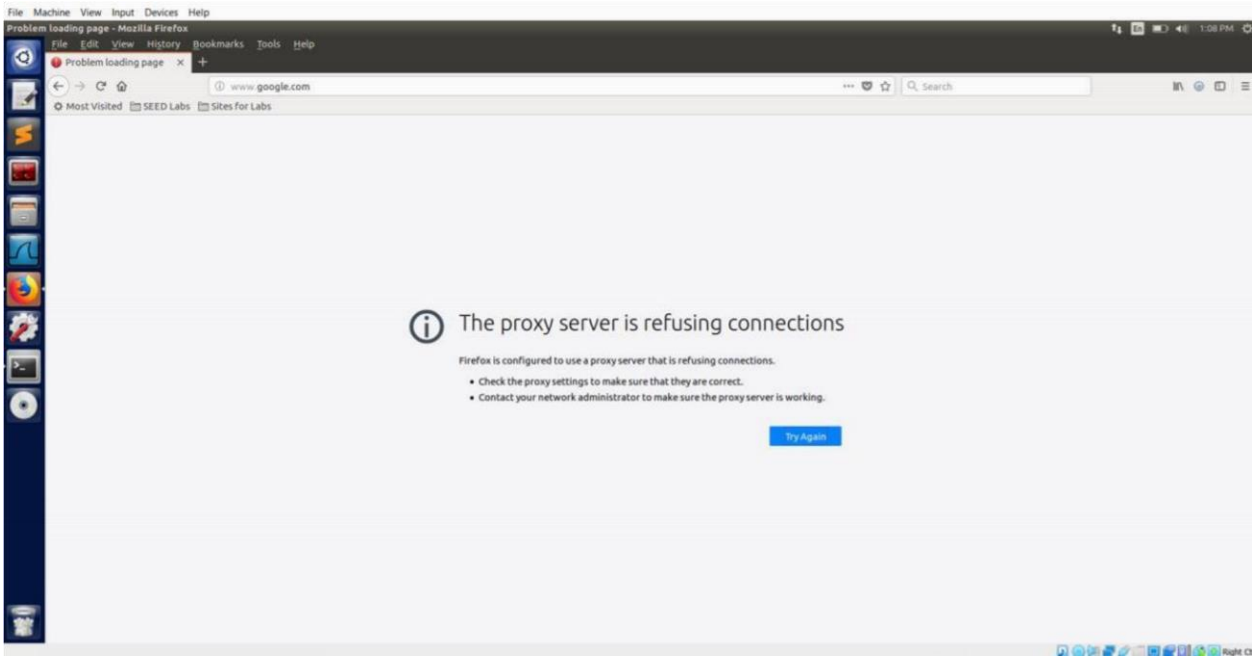
VM1 Wireshark capture.



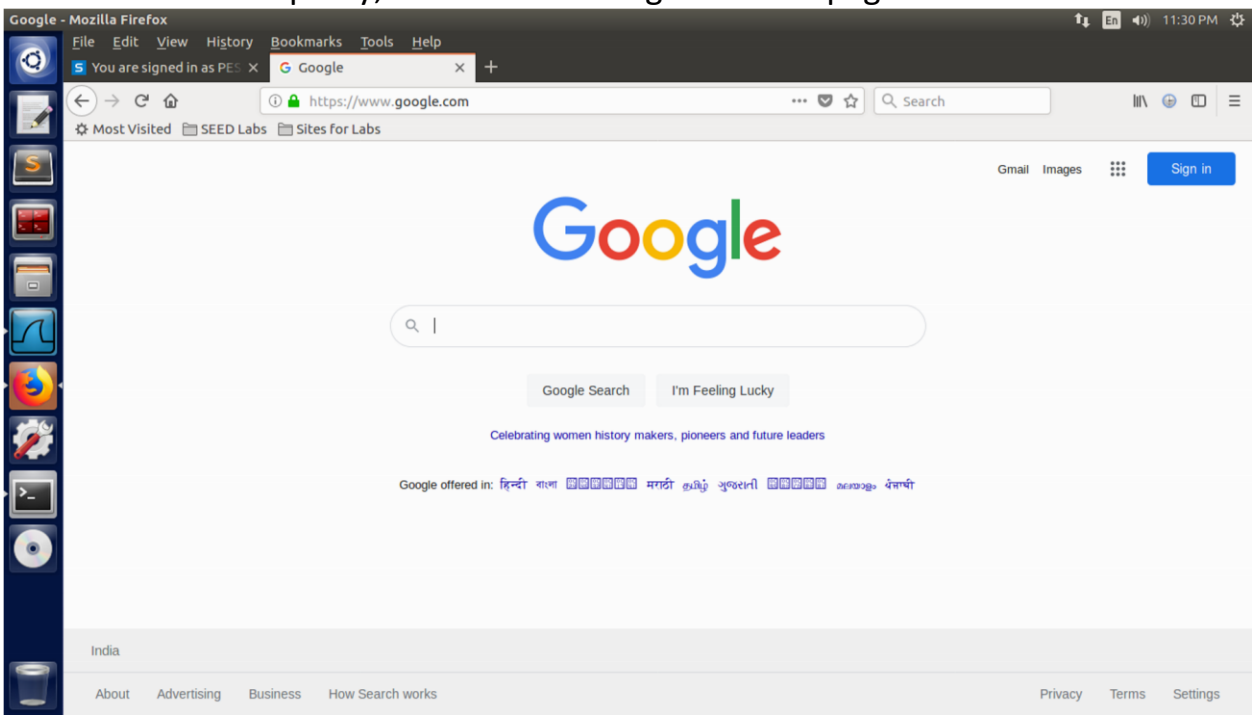
VM2 Wireshark capture.



If we try to visit www.google.com again, we observe that the browser informs the proxy is refusing connections. The browser is still configured to use proxy, but with the tunnel not running any more, the browser cannot use local port 9000 to get the result.



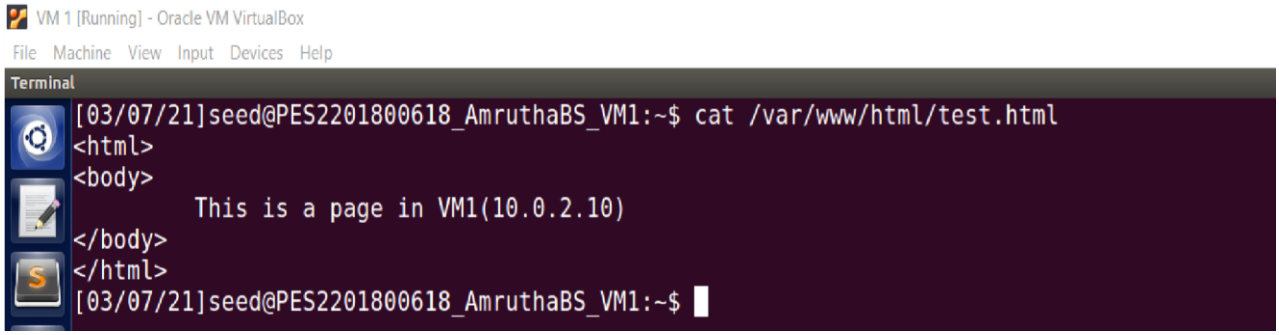
If we re-enable the proxy, the browser will get the webpage.



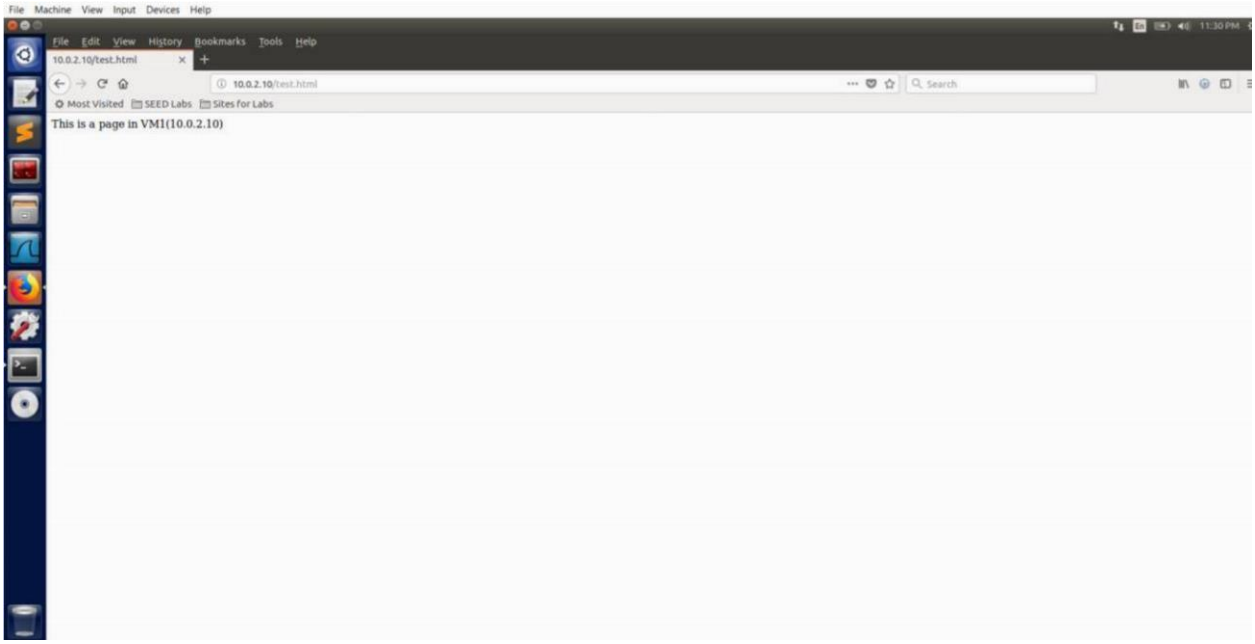
Task 4: Evade Ingress Filtering

In this task, we will block incoming port 80 and port 22 on VM1, but still access a web page on the web server in VM1 from VM2 by using a reverse ssh tunnel.

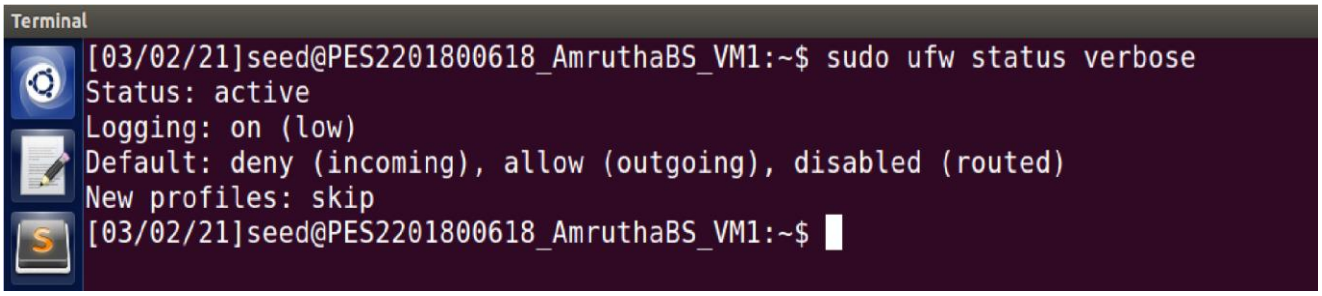
Our goal is to access a secret page on VM1 (test.html) from VM2. The content of the page is shown below:



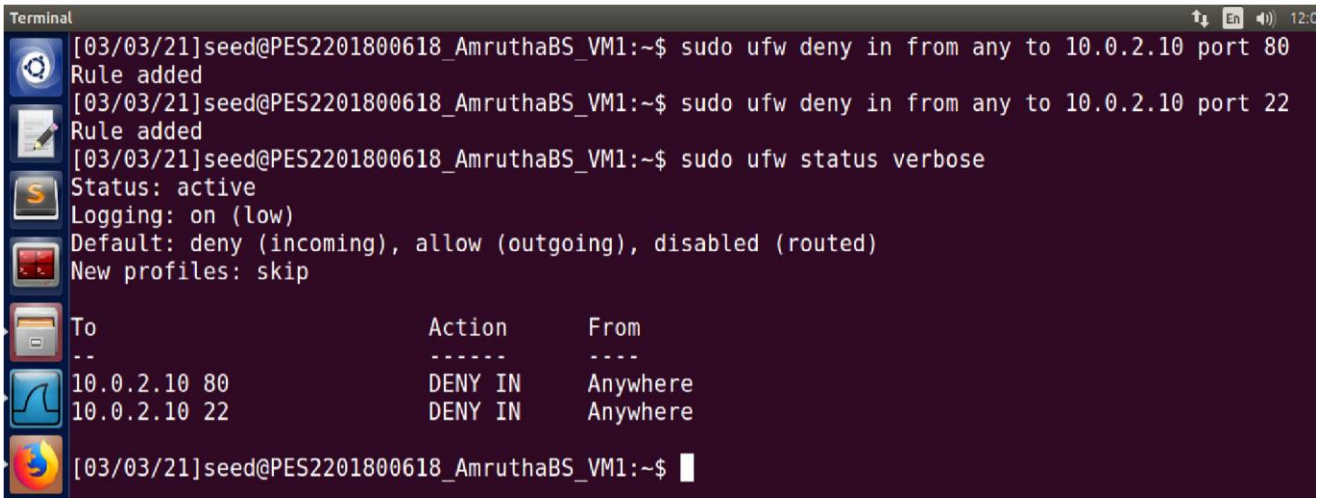
With no firewall rules setup on VM1, we try to access the page from VM2.



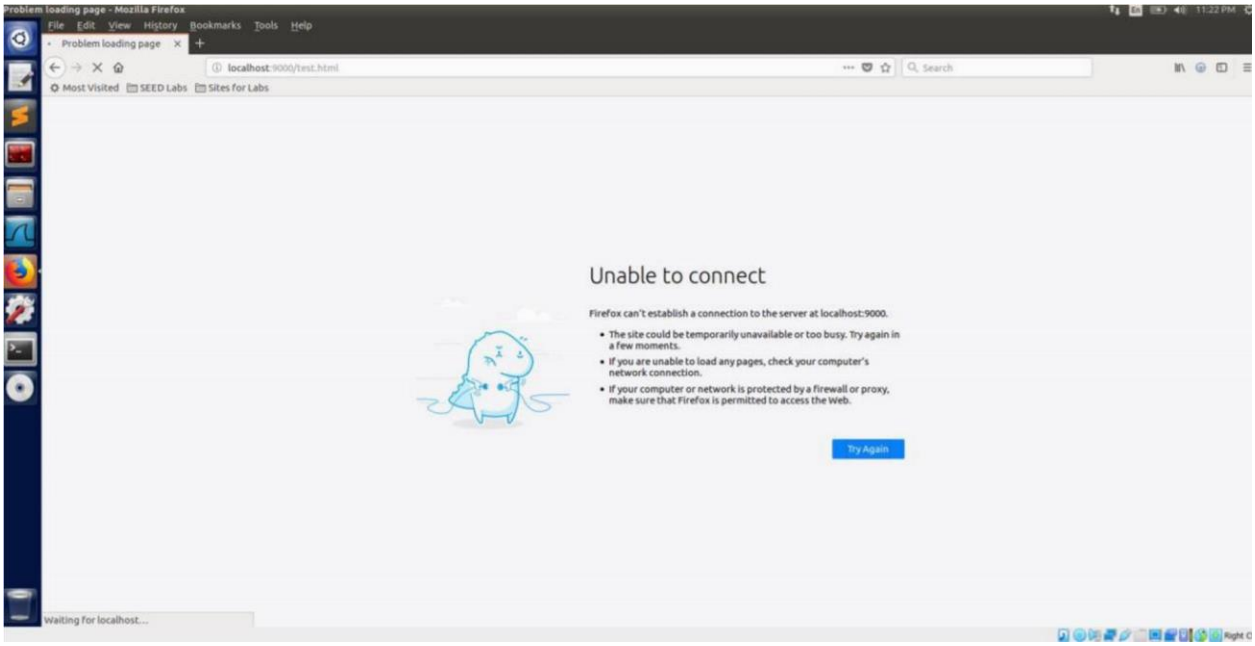
sudo ufw status verbose



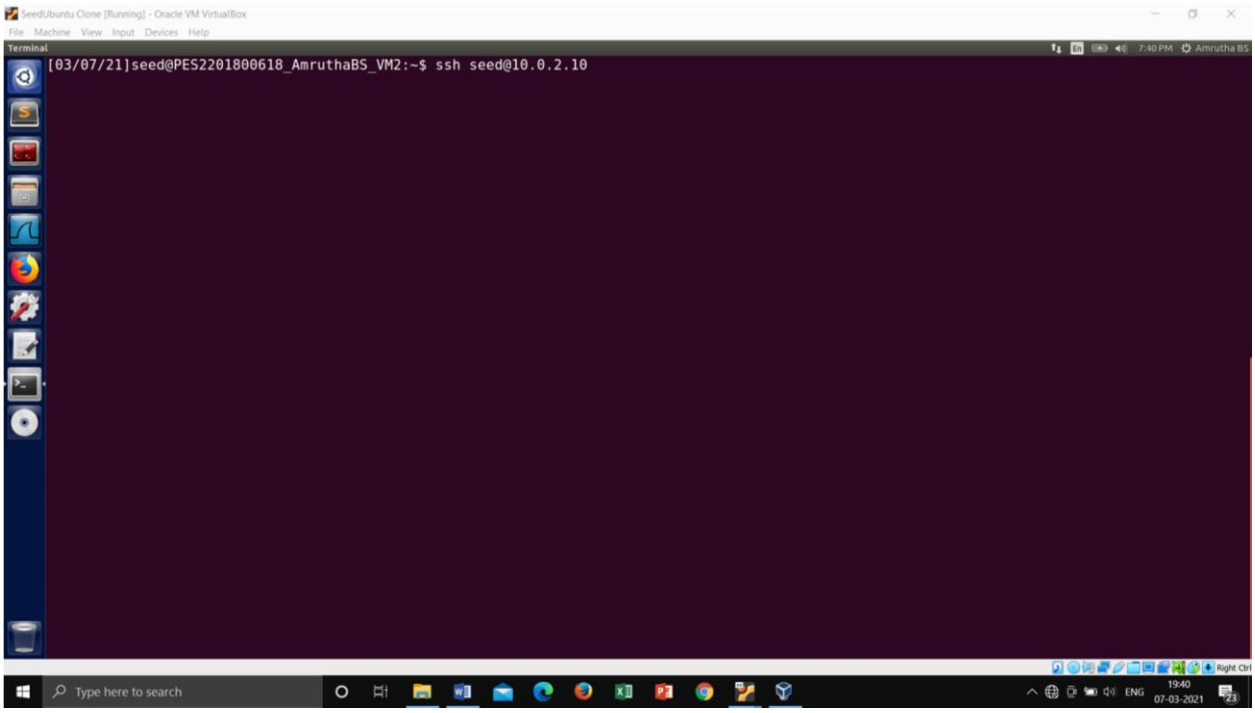
We block incoming requests on port 80 and port 22 on VM1



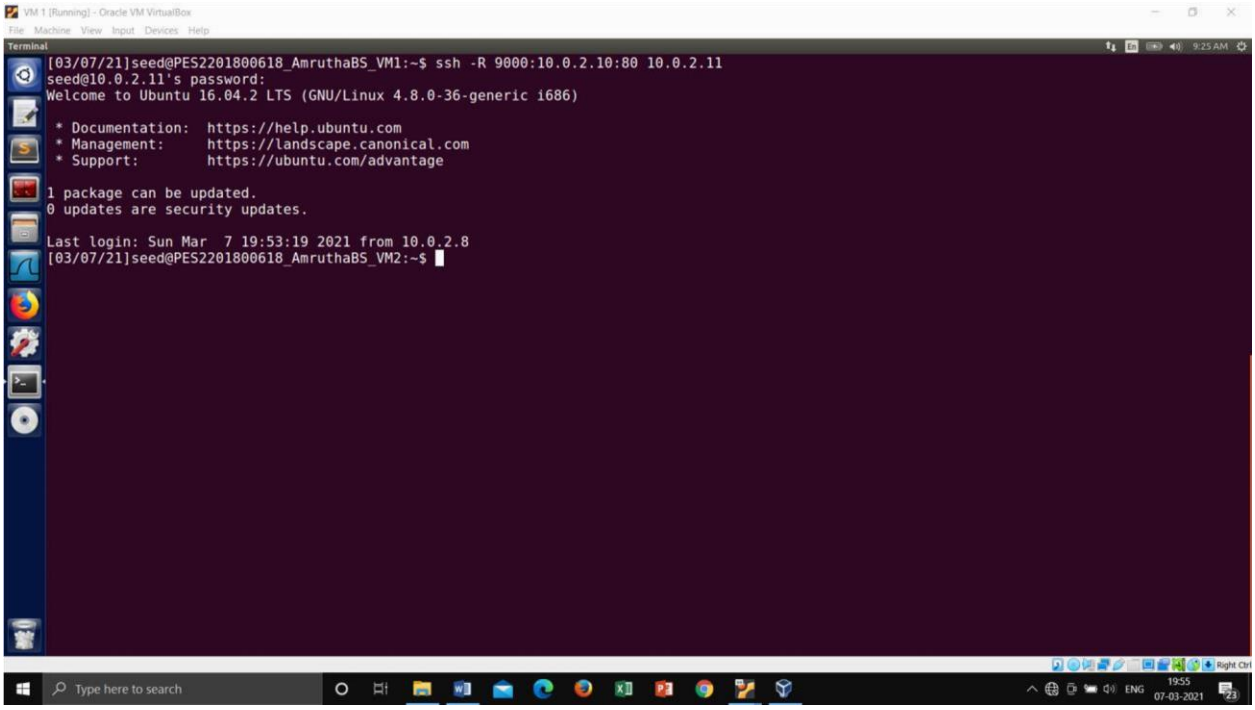
If we try to access the page again from VM2, the page is no longer accessible



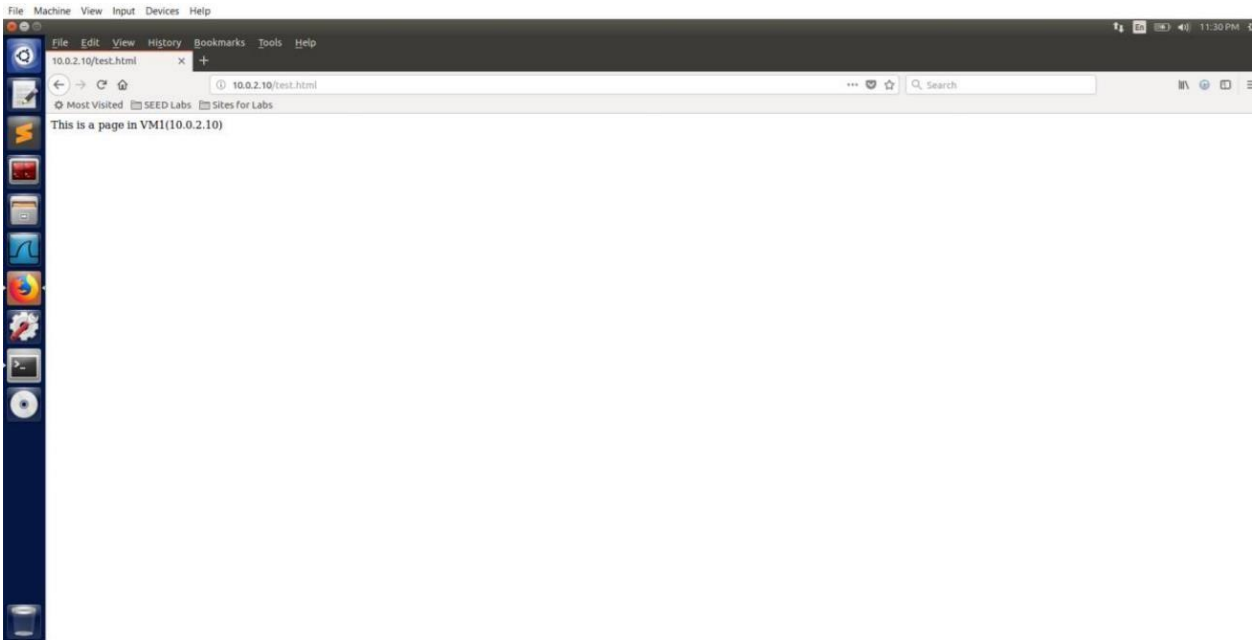
VM2 cannot ssh to VM1 because port 22 is blocked.



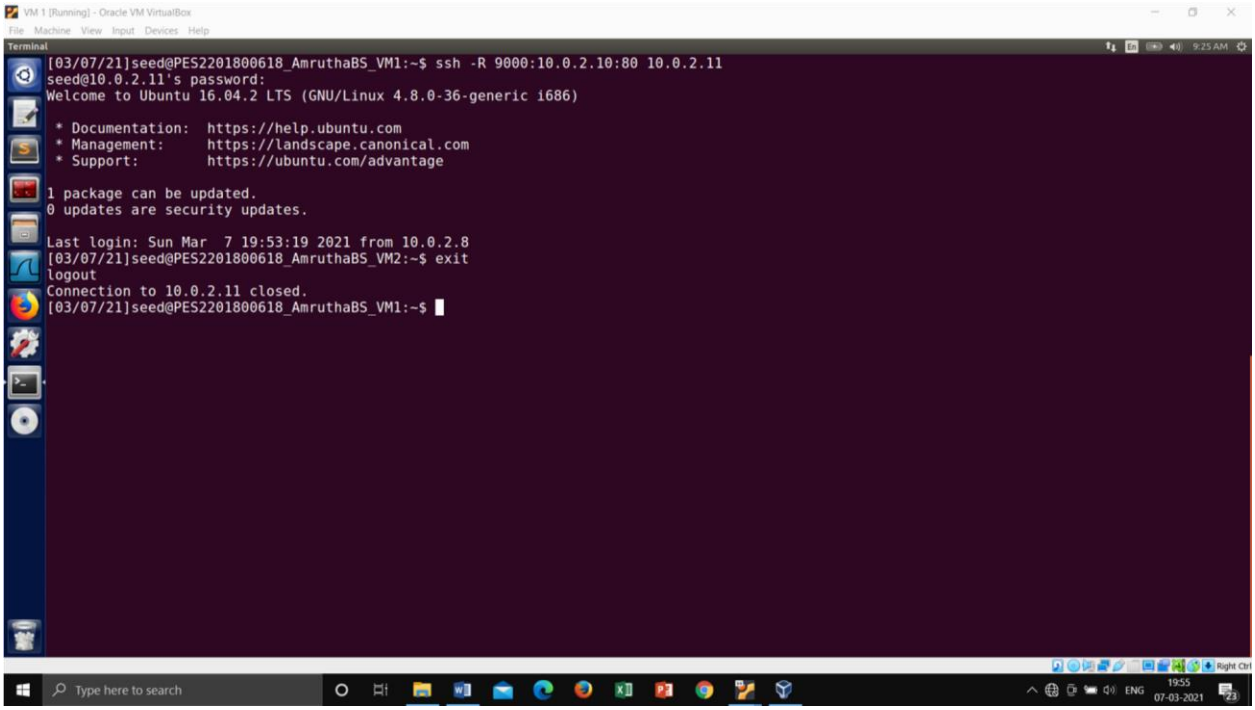
We set up a reverse tunnel. Using this VM2 can use its local port 9000 to access port 80 on VM1.



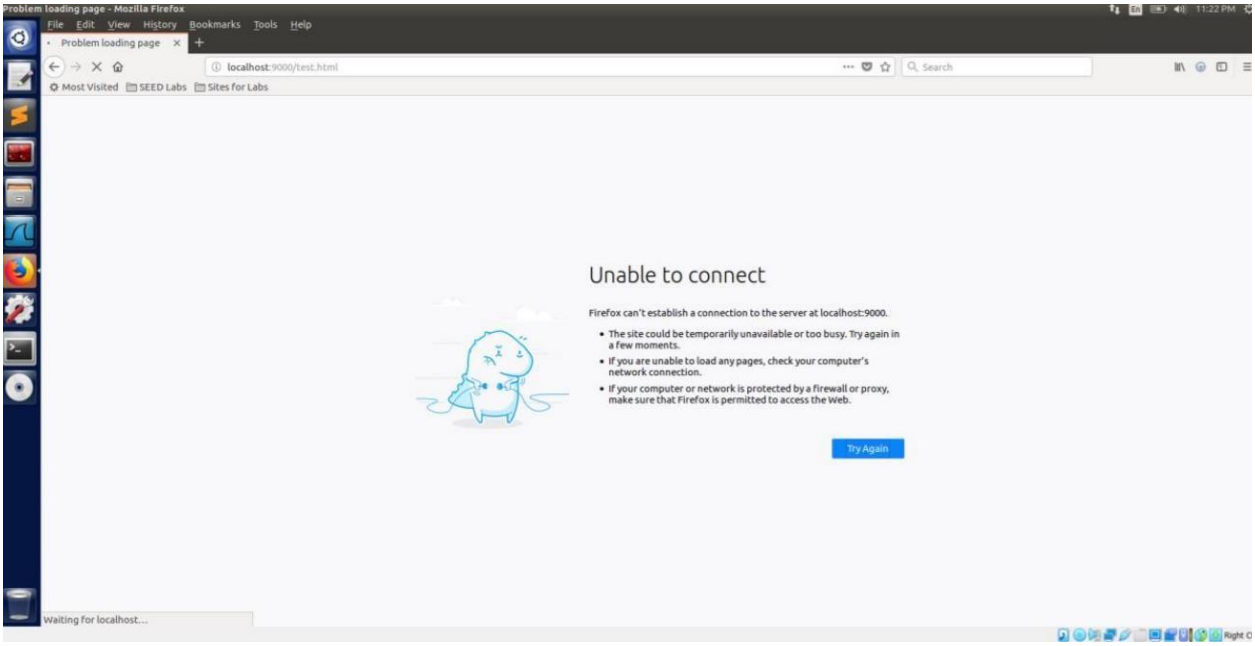
With the tunnel setup, we can access the webpage using port 9000 on VM2



Breaking the tunnel by using exit command.



With the tunnel broken, we can no longer access the page on VM1 from VM2


Problem loading page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Problem loading page
localhost:9000/test.html
Search
Most Visited
SEED Labs
Sites for Labs
Unable to connect
Firefox can't establish a connection to the server at localhost:9000.
The site could be temporarily unavailable or too busy. Try again in a few moments.
If you are unable to load any pages, check your computer's network connection.
If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.
Try Again
Waiting for localhost...