COMPUTER NETWORK SECURITY LAB - UE18CS335

LAB 1: Linux Firewall Exploration Lab

Name: Ankitha P
Srn : PES1201801491 (43)
Class: 6 'D'
Date : 22/02/2021

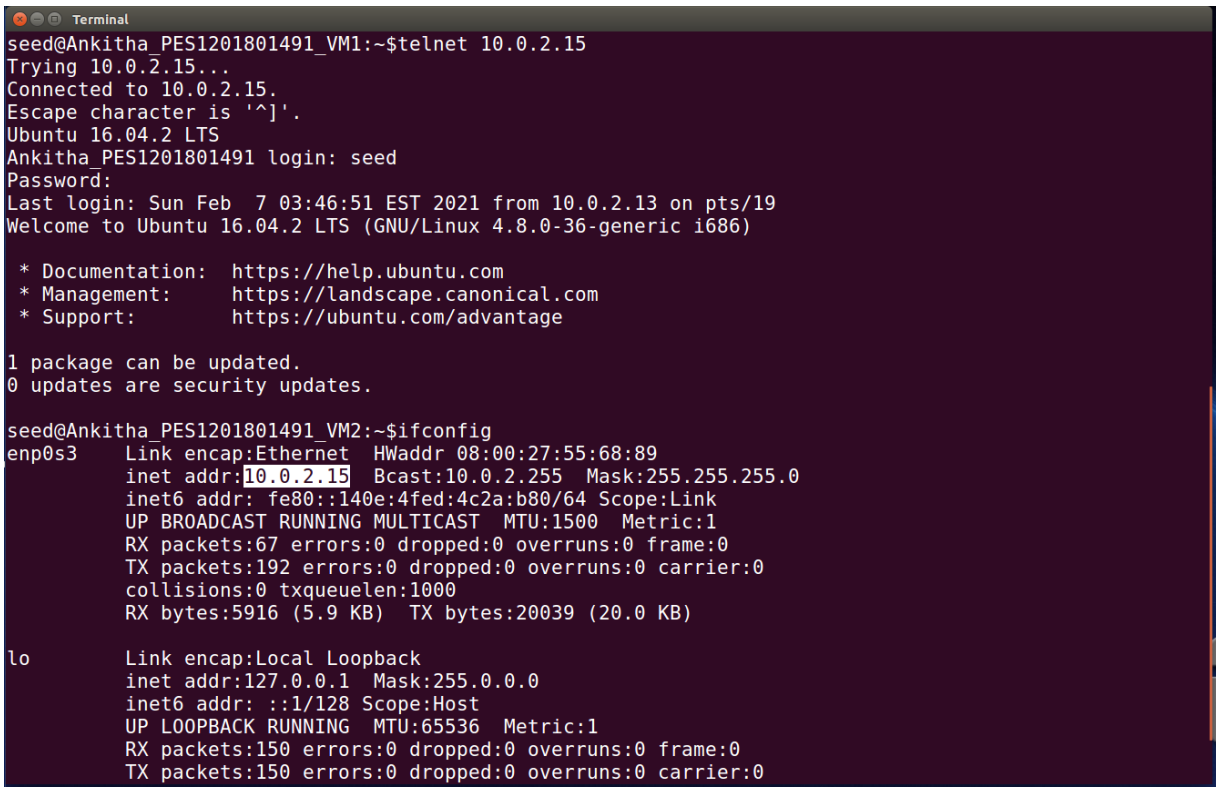
LAB SETUP

VM1: 10.0.2.21

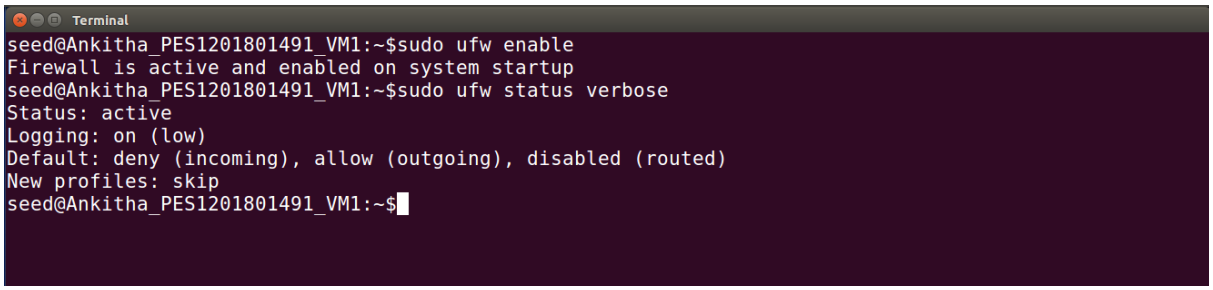
VM2: 10.0.2.15

Task 1 : Using Firewall

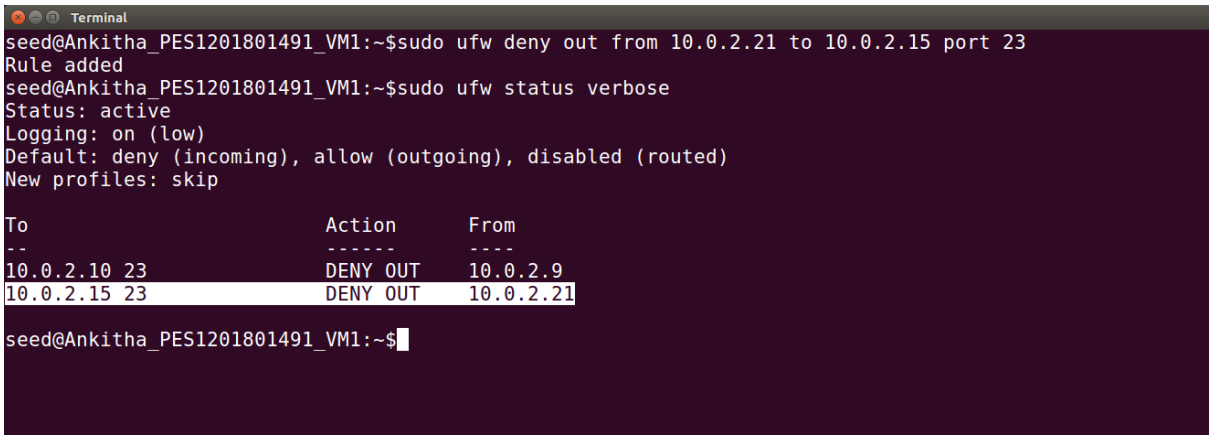
Initially, we check the telnet connectivity from VM1 to VM2. VM1 gets successfully connected to VM2 as shown in the screenshot below.



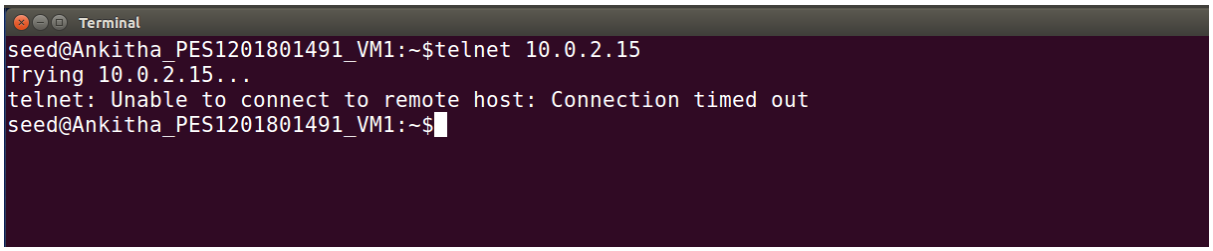
We prevent VM1 from being able to telnet to VM2. For this we configure the firewall using ufw. We will first enable the firewall on VM1 and check the status. We can see the firewall is made active but no rules have been set yet.



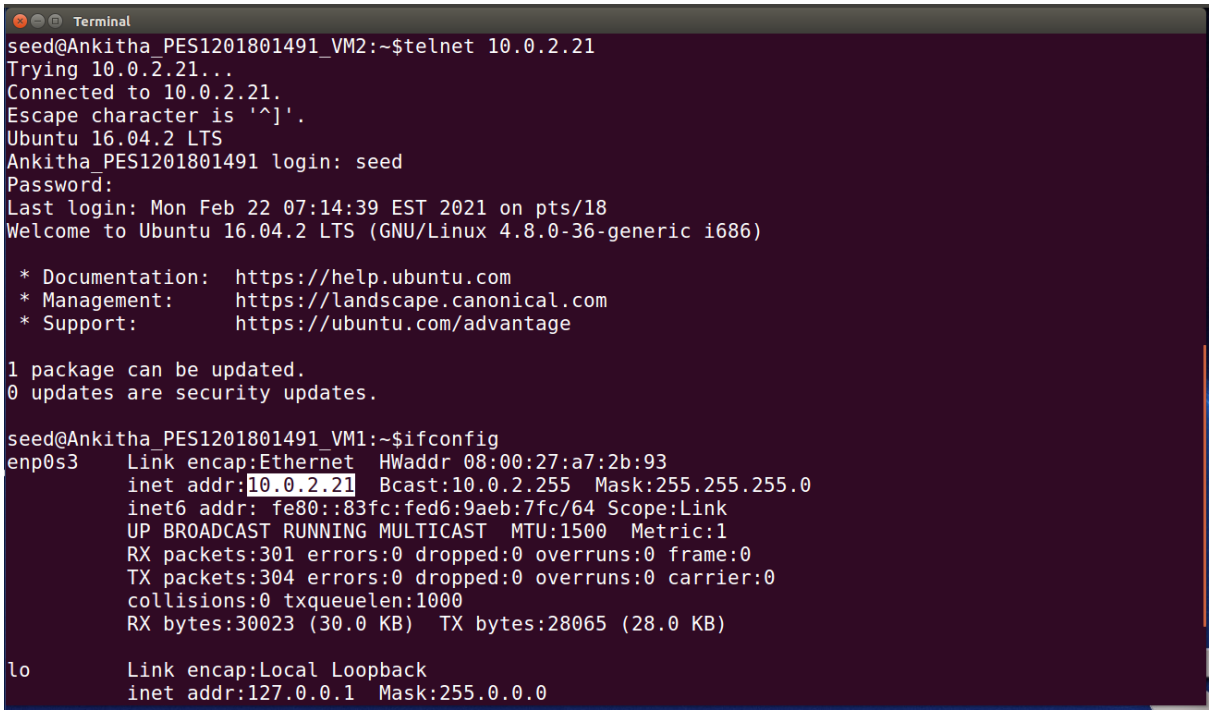
We will next configure the firewall on VM1 to deny telnet (port 23) from VM1(10.0.2.21) to VM2(10.0.2.15) and check the status again. We can see that a firewall rule has been added.



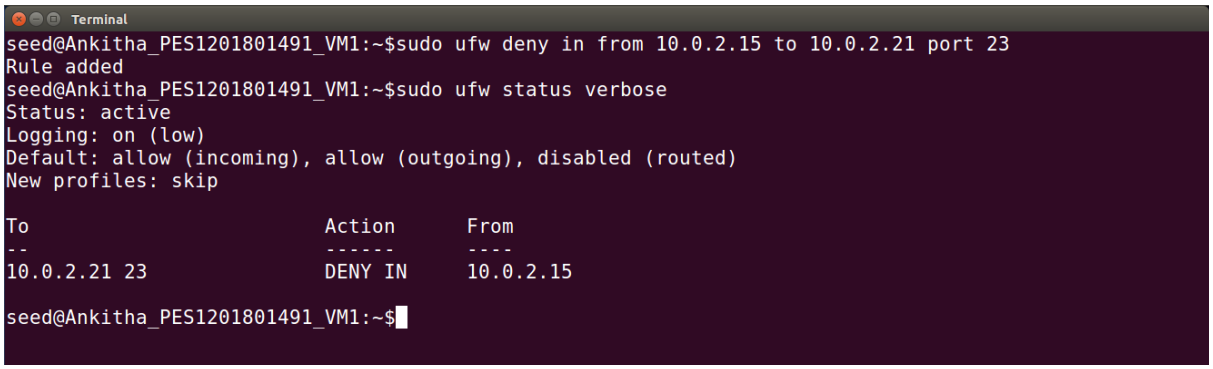
We now try to telnet from VM1 to VM2. Due to the firewall rule, the telnet is denied. It gives a Connection timeout error.



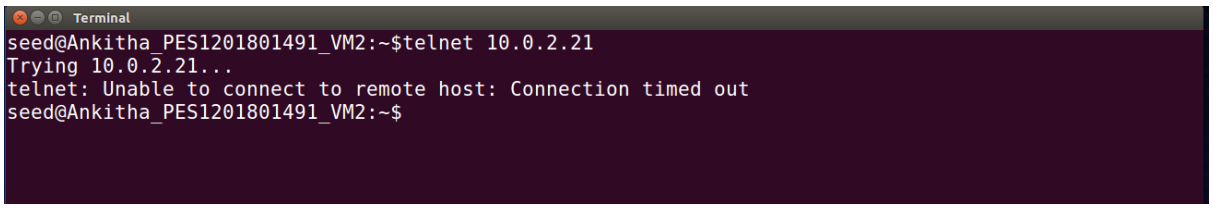
We next need to deny telnet from VM2 to VM1. We will first test whether telnet works from VM2 to VM1. We can see that we still can access VM1 from VM2 as the firewall rule only blocks requests at port 23 from VM1 to VM2.



We delete the previous firewall rule and add a new rule to deny all traffic from VM2 (10.0.2.15) to VM1 (10.0.2.21) on port 23.

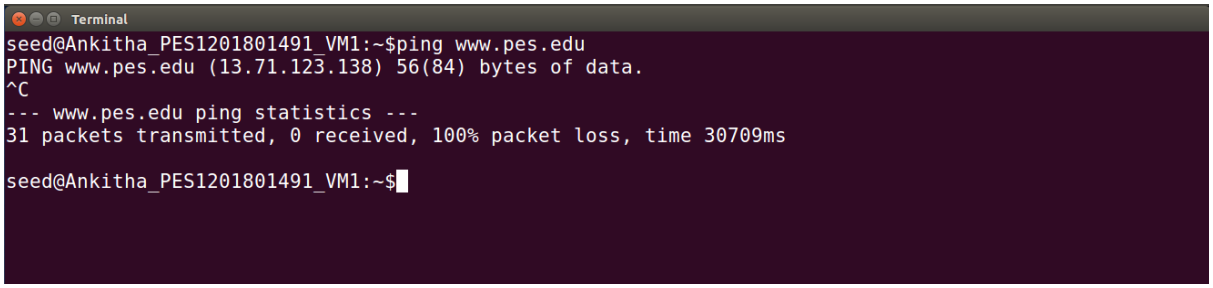


When VM2 tries to telnet to VM1, it doesn't receive any response from VM1 due to the firewall blocking telnet packets from VM2.

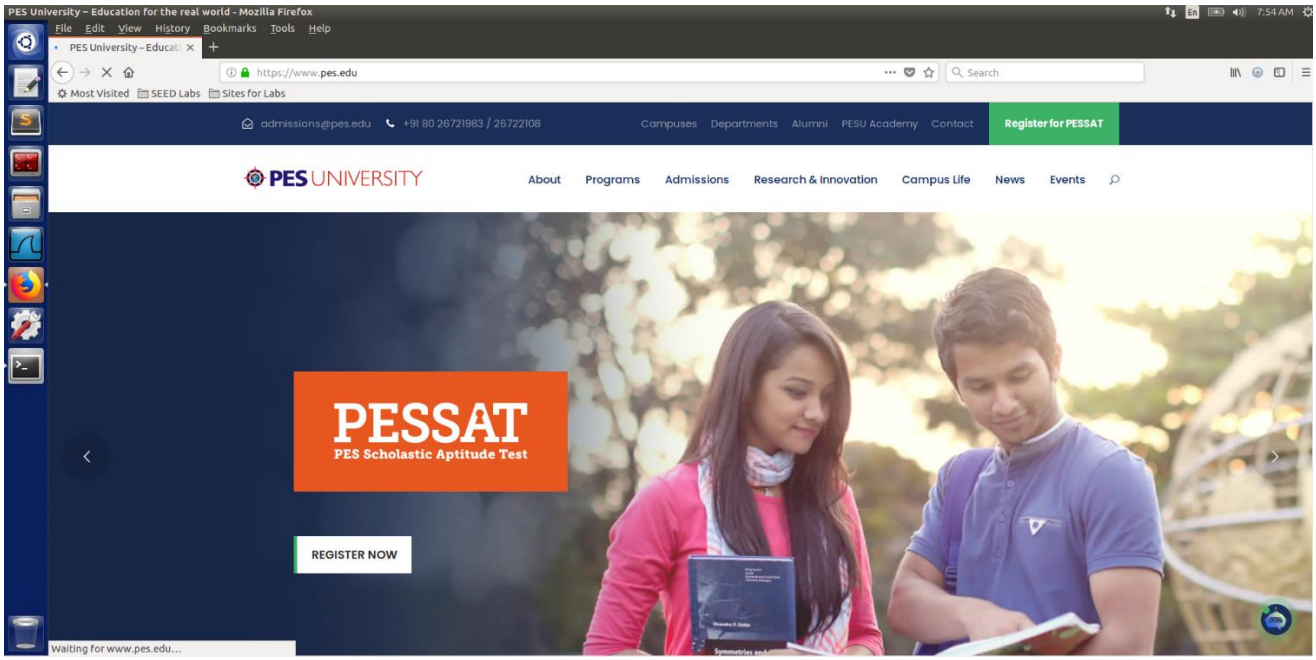


Blocking VM1 from accessing a website

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



We are able to ping www.pes.edu and we obtain the ip address as 13.71.123.138 which we will use for setting firewall rules later. We also make sure that we can access the webpage in the browser and then clear the



browser cache afterwards to prevent cached memory from reloading on the next access to the page.

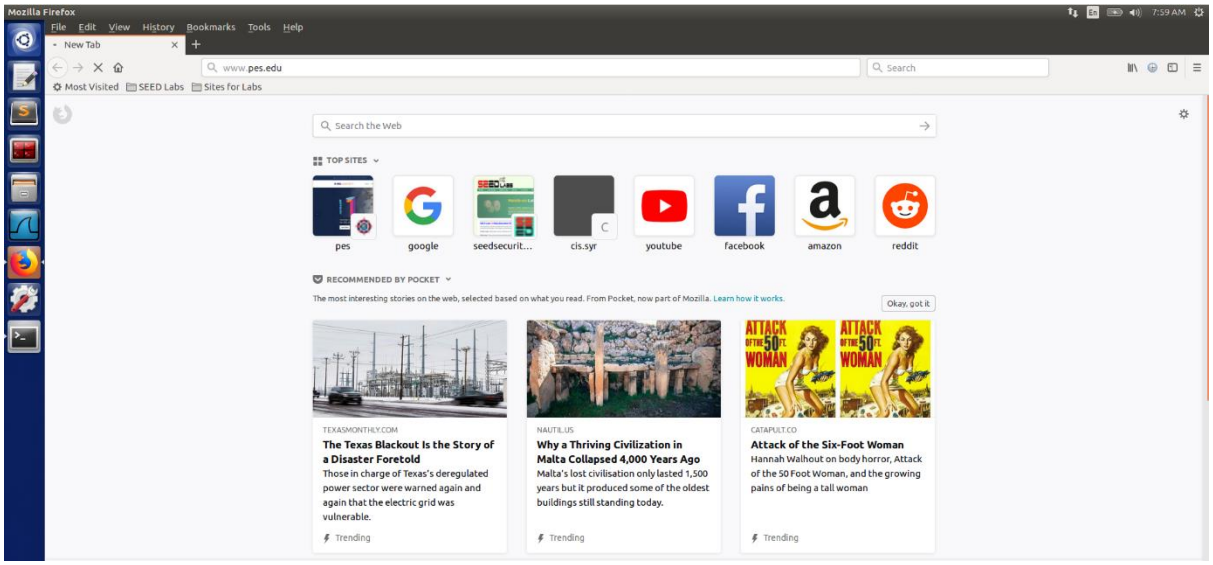
Once the connectivity is confirmed, a firewall rule is added to deny all traffic to the website's IP address (obtained from output of ping).

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.

```
Terminal
seed@Ankitha_PES1201801491_VM1:~$sudo ufw deny out to 13.71.123.138
Rule added
seed@Ankitha_PES1201801491_VM1:~$sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
13.71.123.138              DENY OUT    Anywhere

seed@Ankitha_PES1201801491_VM1:~$
seed@Ankitha_PES1201801491_VM1:~$
seed@Ankitha_PES1201801491_VM1:~$
seed@Ankitha_PES1201801491_VM1:~$ping www.pes.edu
PING www.pes.edu (13.71.123.138) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- www.pes.edu ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4087ms

seed@Ankitha_PES1201801491_VM1:~$
```

We can confirm this by accessing www.pes.edu in the browser as well and we can see that the page does not load and eventually shows Connection Timed Out.

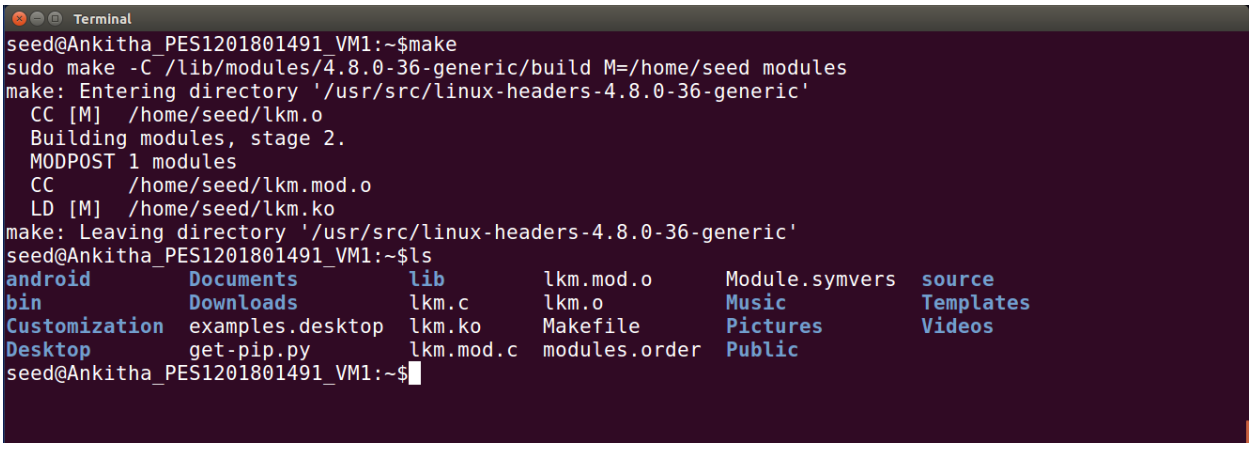
Hence the firewall works successfully and VM1 is prevented from accessing a website as shown below

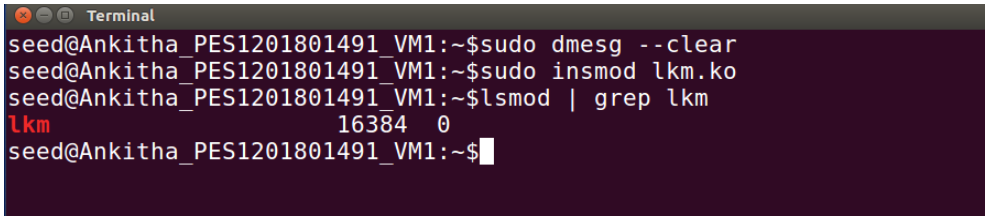


Task 2 : How Firewall works

We use the lkm.c code provided to implement rules in the firewall to block telnet, ssh and external website access from our VMs and we verify each one in this task.

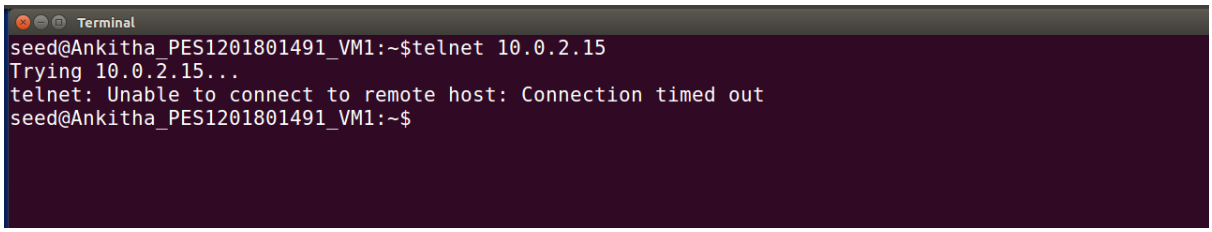
We compile the above code using a Makefile and load the compiler lkm module into the kernel using insmod as shown below.



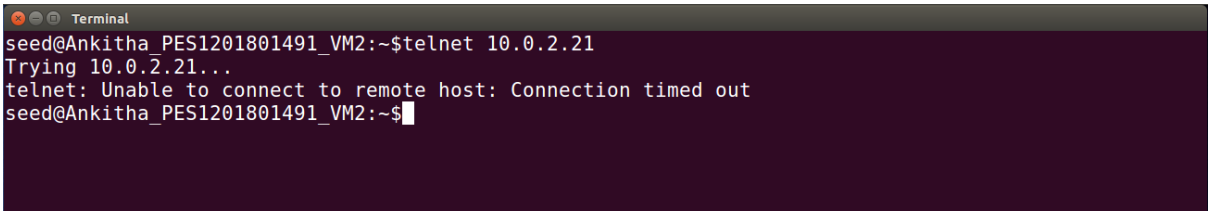


Testing our firewall rules:

1. Telnet from VM1 to VM2

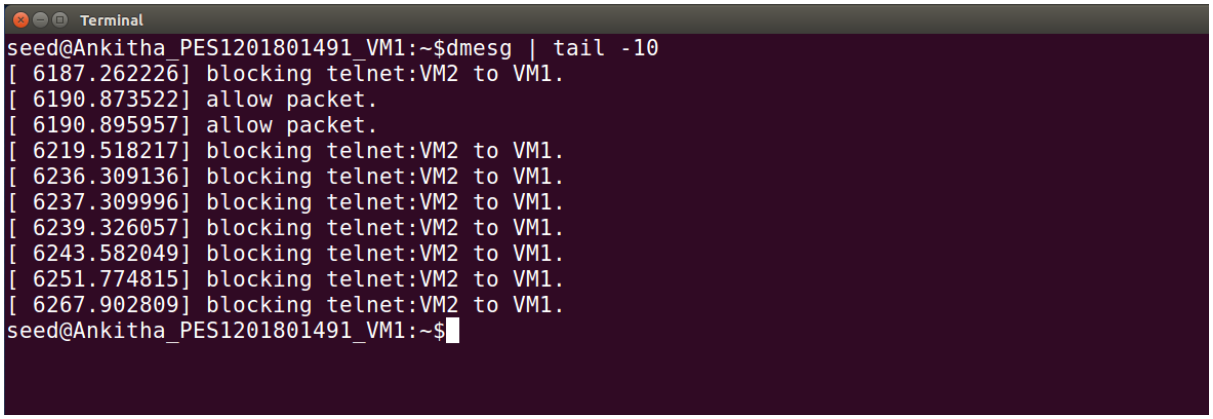


2. Telnet from VM2 to VM1

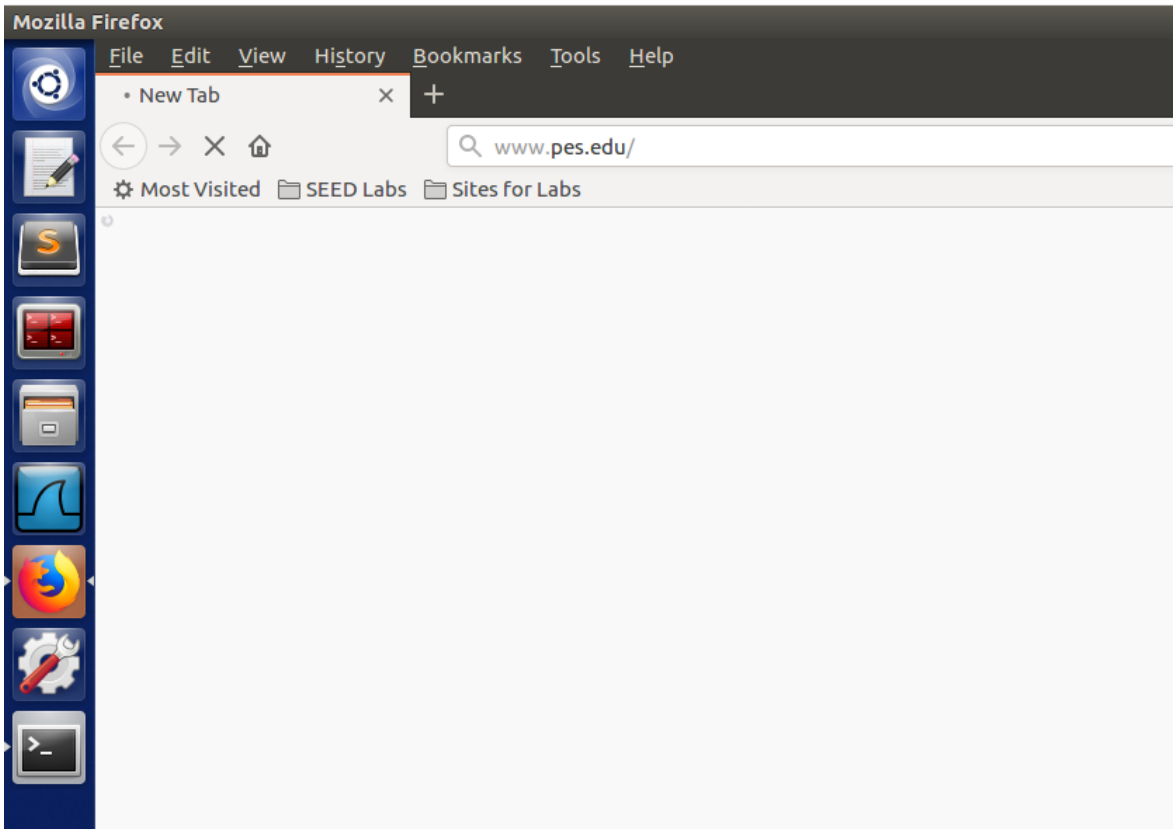


It is observed that we are unsuccessful in sending a telnet request from VM2 to VM1 and vice versa. Hence our firewall rules work correctly

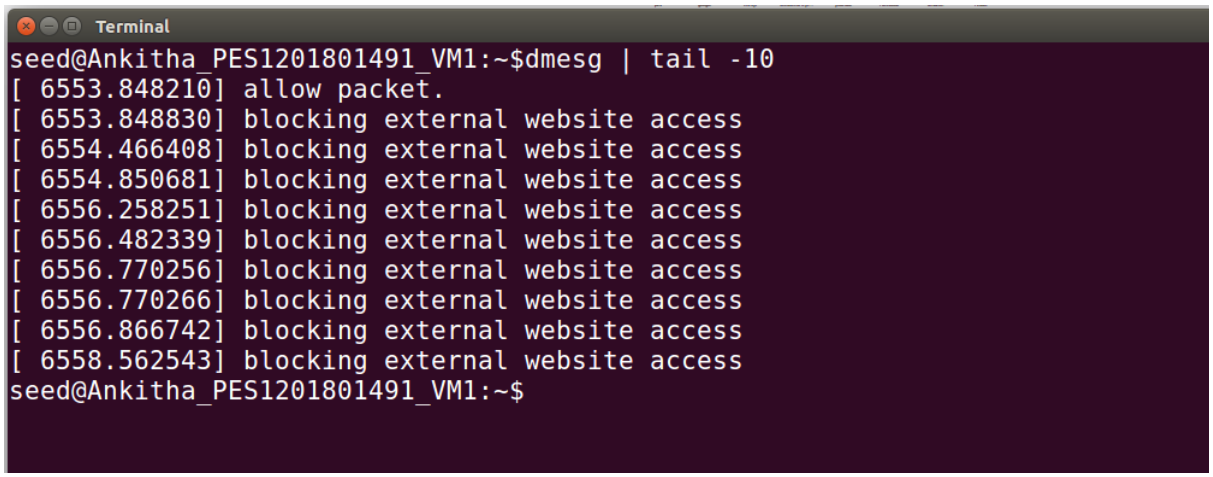
This can be confirmed by viewing details of the packets dropped by the firewall as shown in the screenshot below. We can see the firewall blocking telnet packets from VM2 -> VM1.



3. Block external website access from VM1 We try to access www.pes.edu (browser cache is cleared) on VM1. The website does not load and when we run the dmesg command we can see packets from VM1 at port 80 are getting dropped and the message “blocking external website access” is displayed.

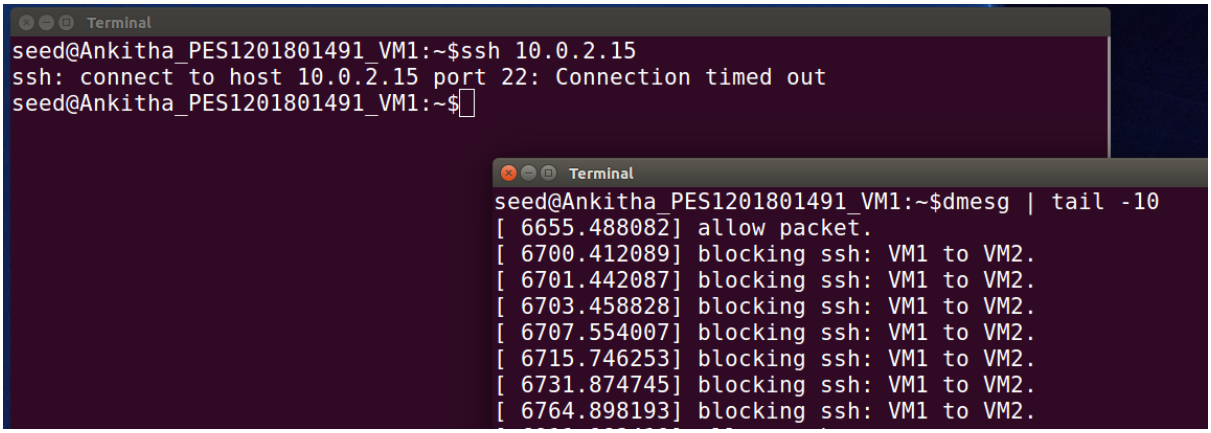


The site is blocked due to the firewall rules that have been added, denying packets on port 80 (for HTTP).



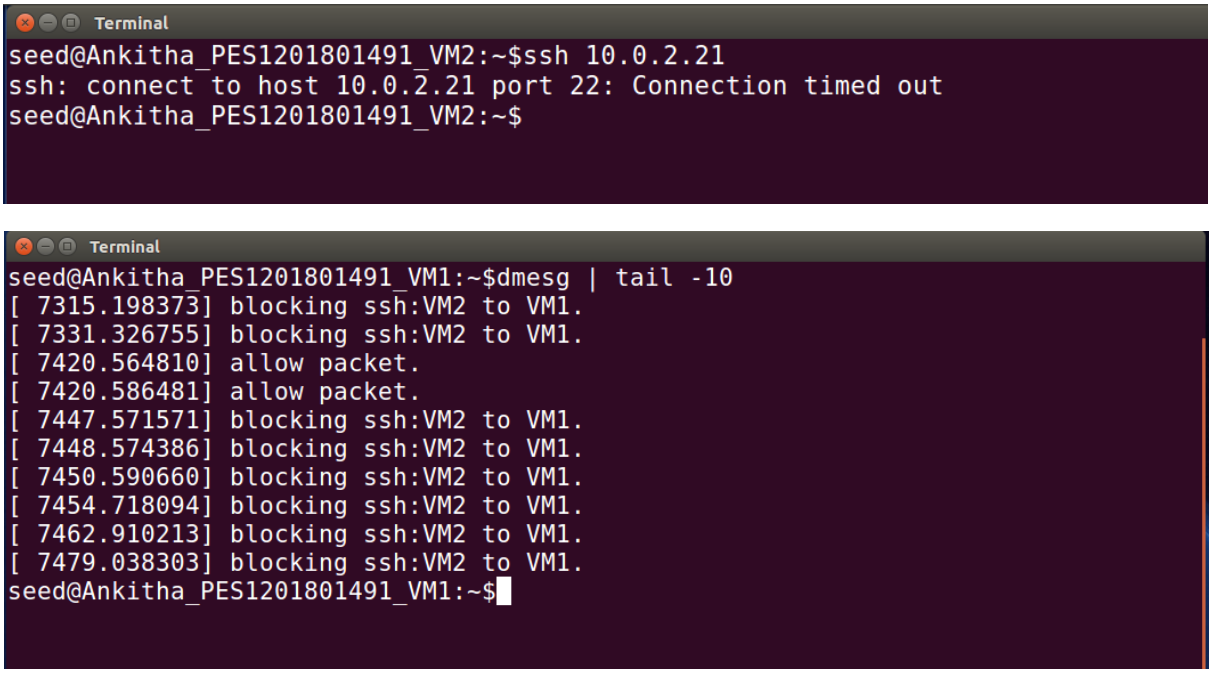
4. Block ssh from VM1 to VM2

The firewall rules also blocks SSH between VM1 and VM2. This is confirmed by sending a SSH request to VM2 from VM1. . We observe the message “blocking ssh: VM1 to VM2” on running the dmesg command meaning that the SSH packets which were sent from VM1 to VM2 were blocked by the firewall rule and dropped.



5. Ssh from VM2 to VM1

We observe the message “blocking ssh: VM1 to VM2” on running the dmesg command meaning that the SSH packets which were sent from VM1 to VM2 were blocked by the firewall rule and dropped.



Task 3: Evading Egress Filtering

VM1 is blocked from being able to telnet to any machine. In order to access VM3 from VM1, we setup a SSH tunnel via VM2. The below screenshot shows the updation of firewall rules to block all telnet requests from VM1.

```
Terminal
seed@Ankitha_PES1201801491_VM1:~$sudo ufw deny out from 10.0.2.21 to any port 23
Rule added
seed@Ankitha_PES1201801491_VM1:~$sudo ufw status verbose
Status: active
Logging: on (low)
Default: allow (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
23                         DENY OUT    10.0.2.21

seed@Ankitha_PES1201801491_VM1:~$
```

We try to access machine B(VM3) from VM1 using telnet but we are unsuccessful because of the firewall rule set and we get a Connection Timed Out error

```
Terminal
seed@Ankitha_PES1201801491_VM1:~$telnet 10.0.2.15
Trying 10.0.2.15...
telnet: Unable to connect to remote host: Connection timed out
seed@Ankitha_PES1201801491_VM1:~$
```

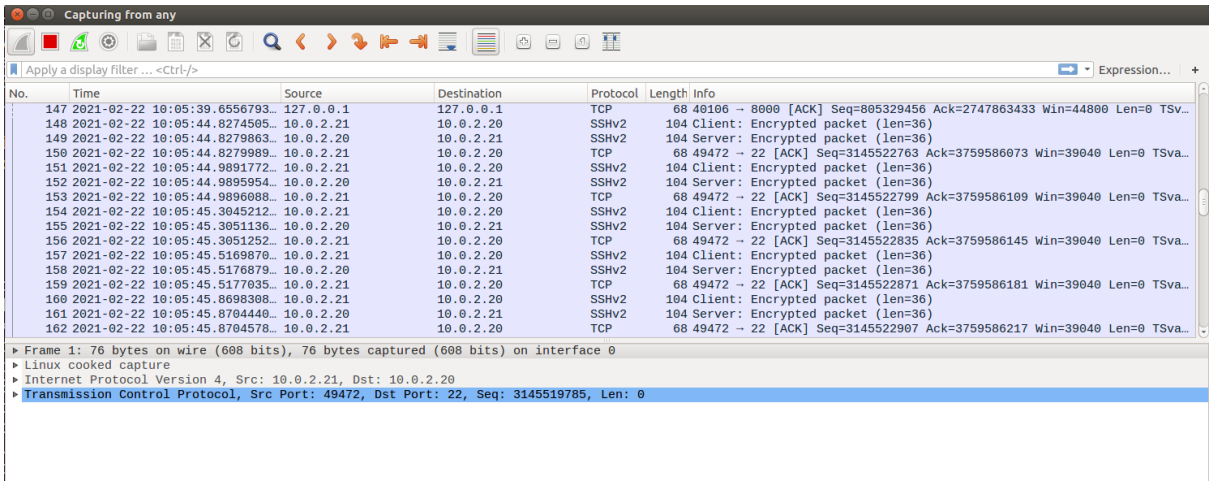
To overcome this, we setup a ssh tunnel between VM1 and VM2 to allow VM1 to telnet to VM3 via VM2, evading the firewall on VM1. The ssh command below allows VM1 to use its local port 8000 to telnet to VM3 via VM2 as shown below. With the ssh tunnel setup, we can now telnet from VM1 to VM3 on another terminal even though the firewall policy on VM1 denies outgoing telnet.

```
Terminal
seed@Ankitha_PES1201801491_VM1:~$ssh -L 8000:10.0.2.15:23 seed@10.0.2.20
The authenticity of host '10.0.2.20 (10.0.2.20)' can't be established.
ECDSA key fingerprint is SHA256:p1zAio6c1bI+8HDp5xa+eKRi561aFDaPE1/xq1eYzCI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.20' (ECDSA) to the list of known hosts.
seed@10.0.2.20's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

Last login: Mon Feb 22 06:02:08 2021
seed@Ankitha_1491_VM2:~$ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:38:4e:73
          inet addr:10.0.2.20  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::f08b:5116:45f6:271f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:243 errors:0 dropped:0 overruns:0 frame:0
          TX packets:173 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:31366 (31.3 KB)  TX bytes:21040 (21.0 KB)

lo        Link encap:Local Loopback
```

```
Terminal
seed@Ankitha_PES1201801491_VM1:~$telnet localhost 8000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
Ankitha_PES1201801491 login: seed
Password:
Last login: Mon Feb 22 10:03:04 EST 2021 from 10.0.2.15 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

seed@Ankitha_PES1201801491_VM3:~$ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:55:68:89
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::140e:4fed:4c2a:b80/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

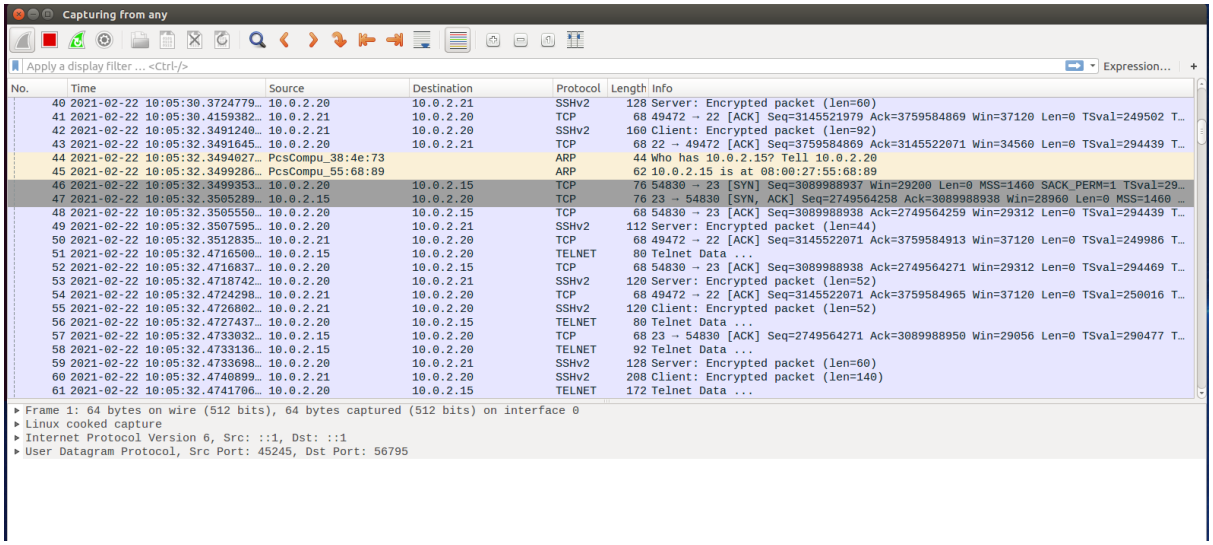
Wireshark capture in VM1 (10.0.2.21)

Wireshark capture in VM1 shows exchange of packets between VM1 and VM2 only. This is because VM1 does not exchange any direct packets with VM3. Telnet packets from VM1 are sent to VM3 via the SSH tunnel in VM2.



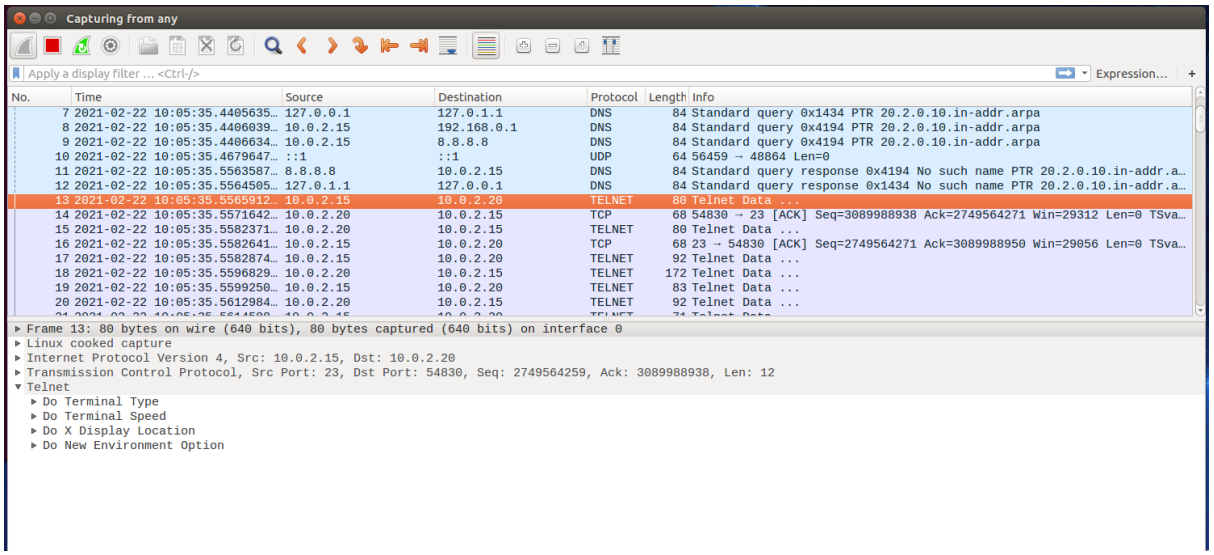
Wireshark capture in VM2 (10.0.2.20)

Wireshark capture in VM2 below shows exchange of packets between VM1 & VM2 and VM2 & VM3. Telnet requests from VM1 and received by VM2. These packets are then forwarded from VM2 to VM3. The reply packets follow a similar route from VM3 to VM1.



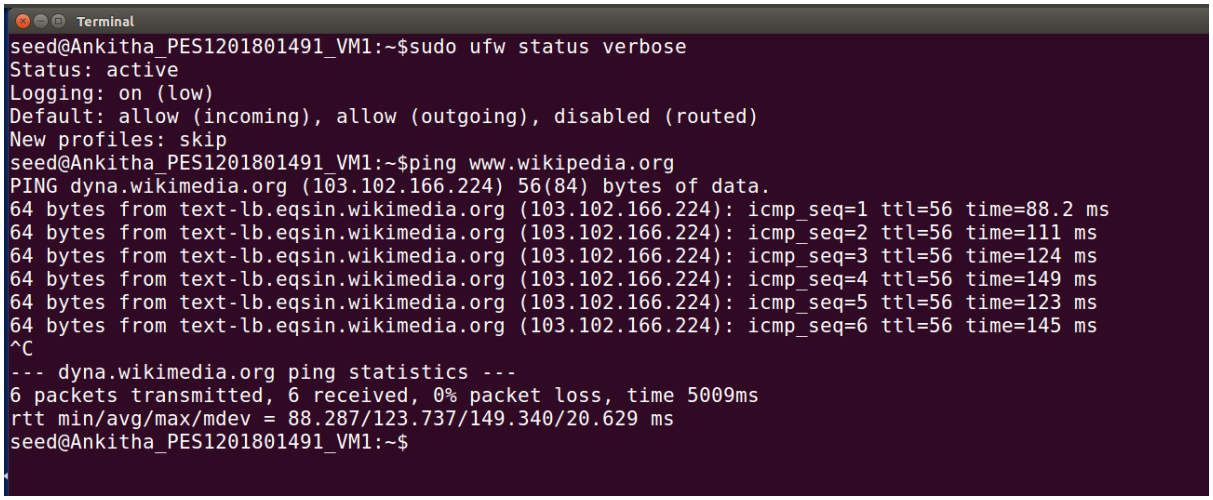
Wireshark capture in VM3 (10.0.2.15)

Wireshark capture in VM3 below shows exchange of packets between VM2 and VM3. VM3 does not exchange any packets directly with VM1. All the data sent from VM1 via telnet is forwarded by VM2 to VM3.

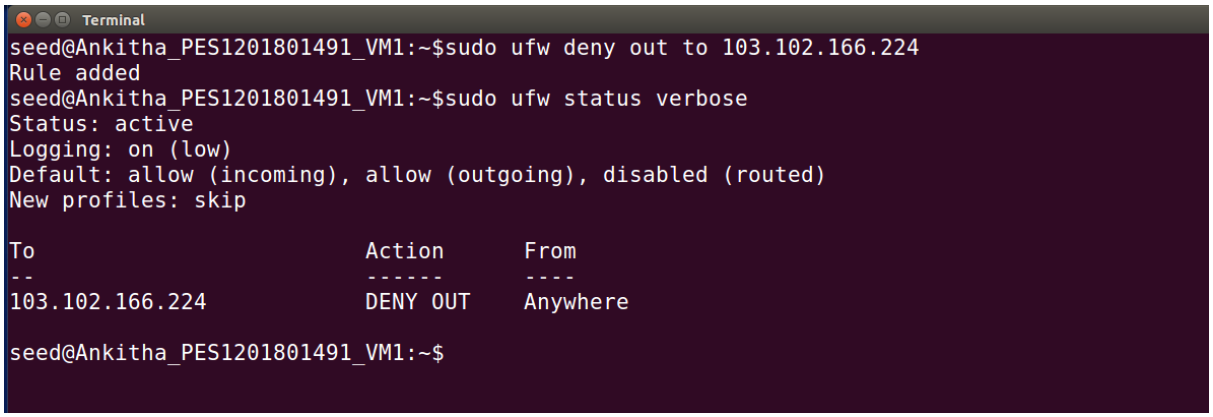


Task 3b : Connecting to Google using SSH tunnel

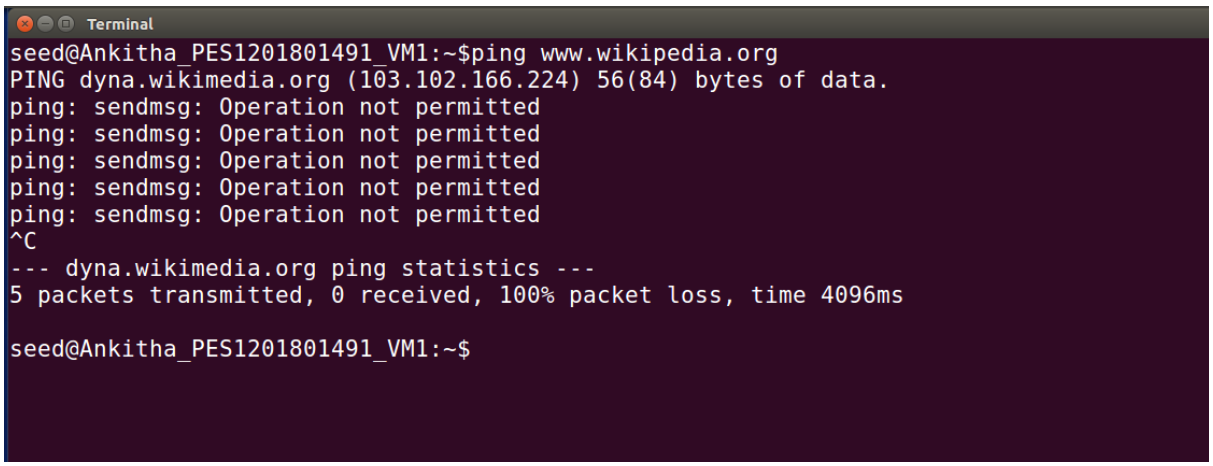
We delete all previous firewall rules and ping www.wikipedia.org to find out which ip address it uses and make a firewall rule to block access to it.



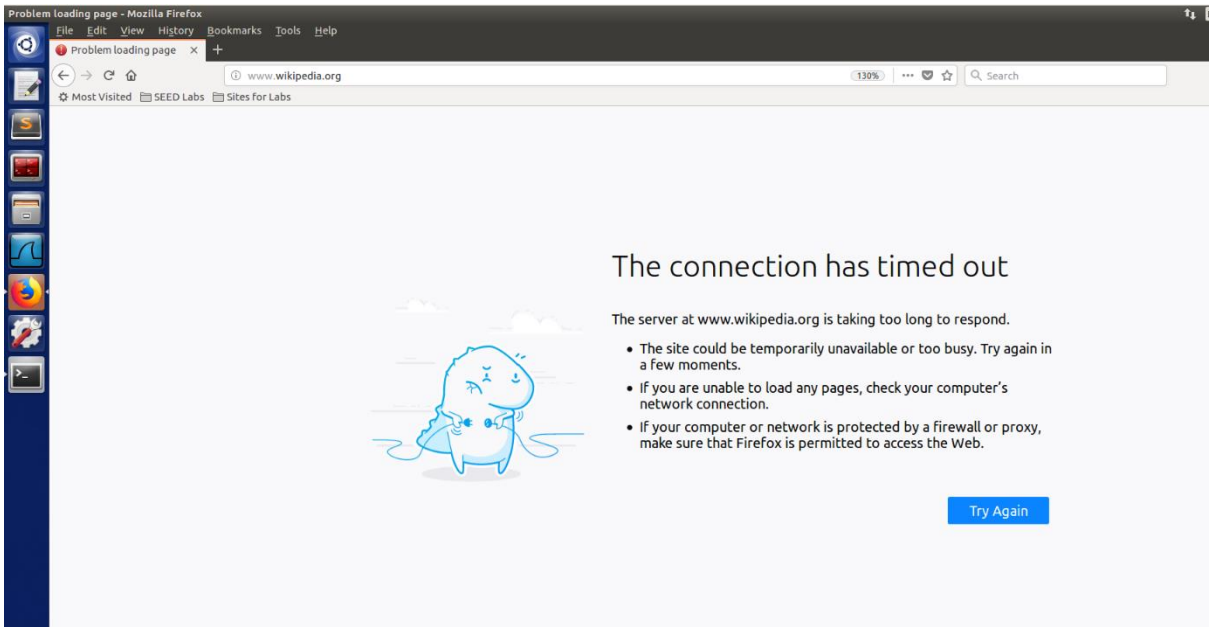
Using 103.102.166.224, the IP address for www.wikipedia.org, we set a firewall rule to block access to it.



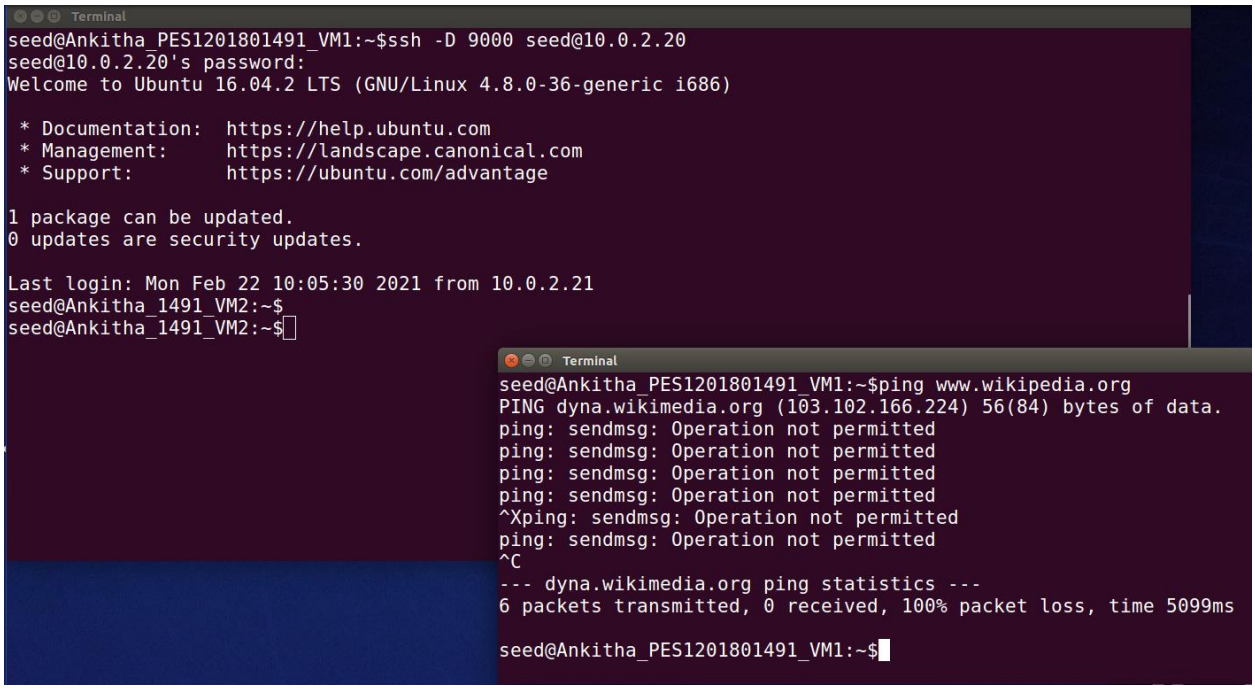
Once the firewall rule is set, pinging to www.wikipedia.org is not permitted (Operation not permitted) since the firewall blocks all the packets.



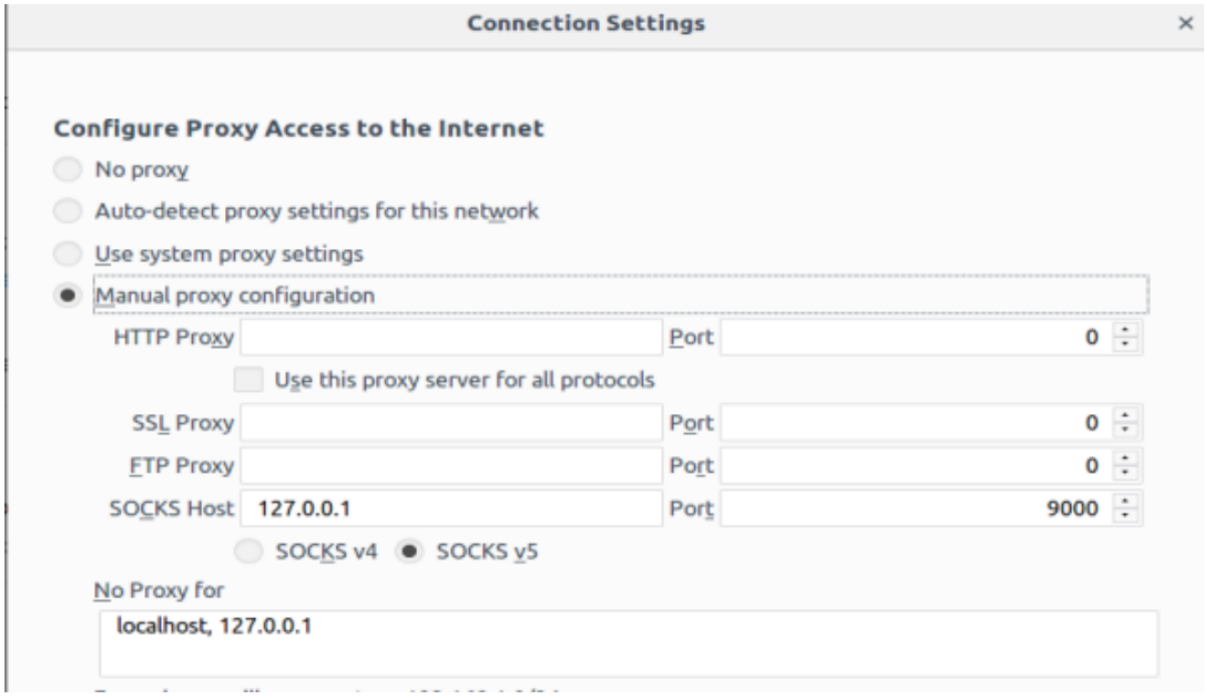
Loading the webpage from web browser also results in connection time out.



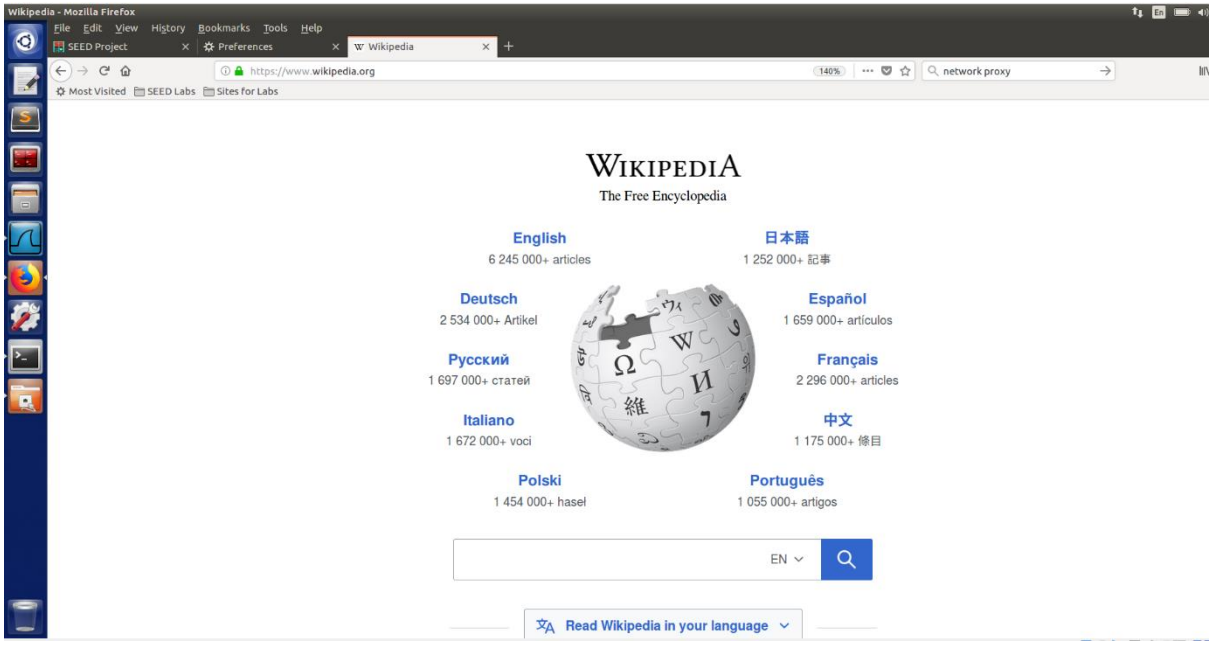
A SSH tunnel with dynamic port forwarding from VM1 to VM2 is setup by forwarding requests to port localhost@9000 to VM2. Localhost at 9000 is used as web proxy server. Ping requests will still be blocked since the ssh server blocks the SSH client's requests in opening a separate side channel as TCP port forwarding is disabled in VM2.



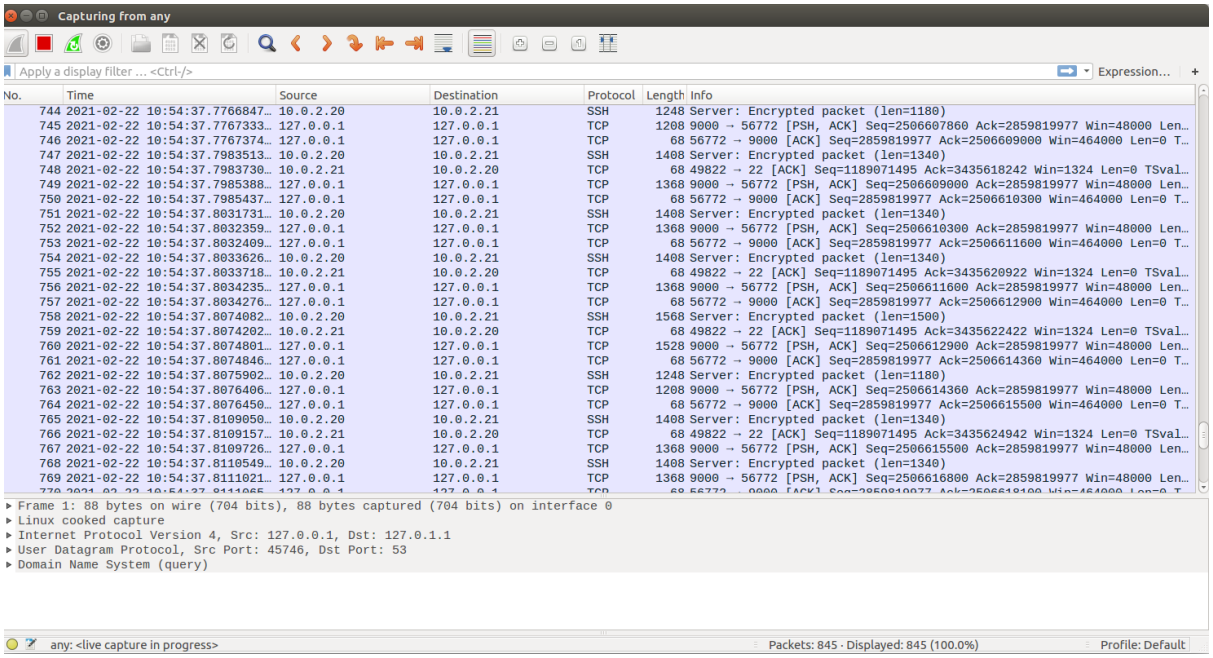
Now to be able to access the website through the established tunnel, we need to set the proxy settings in the firefox browser as shown below.



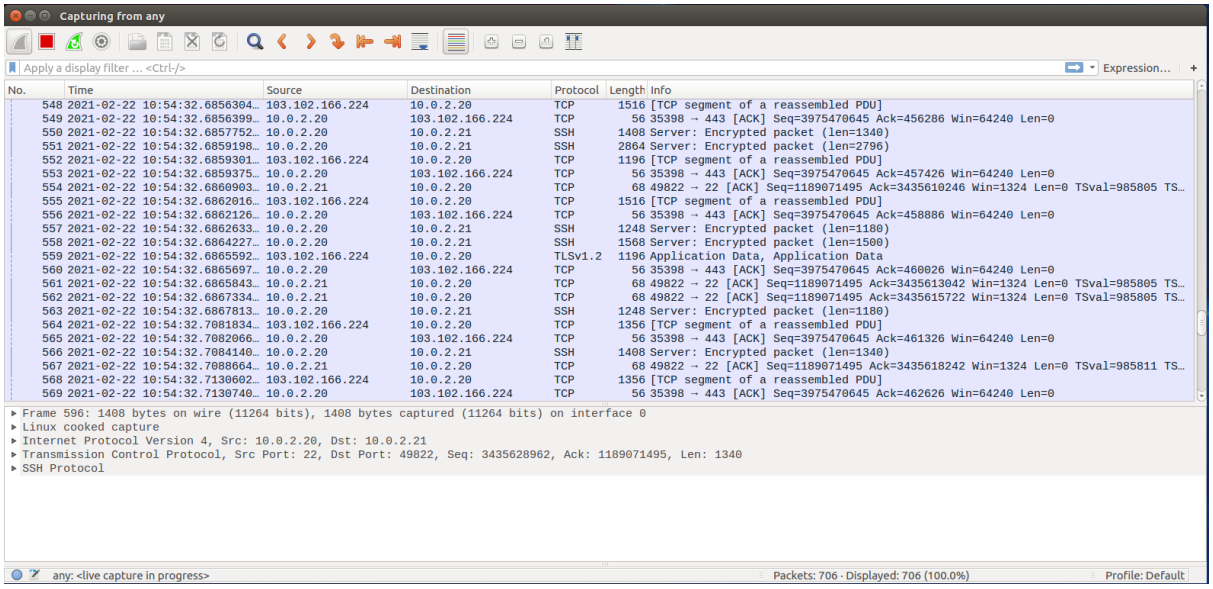
www.wikipedia.org is accessible from web browser through the proxy server invading the firewall



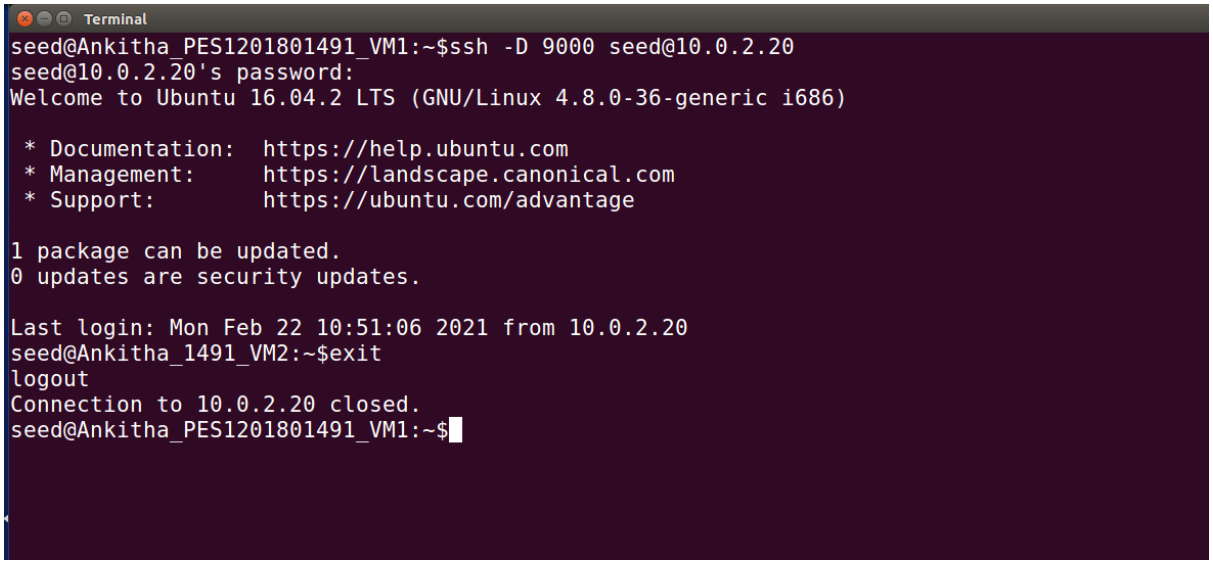
The Wireshark Capture on VM1 shows the connection made to www.wikipedia.org through localhost (127.0.0.1) as set by the proxy server.



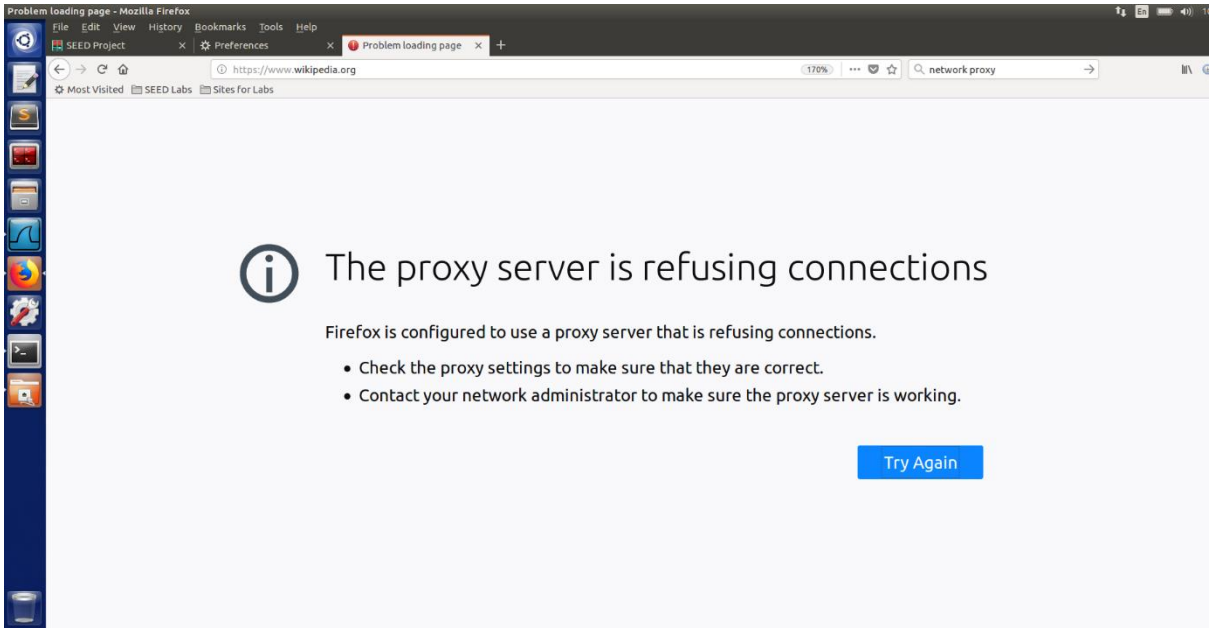
The Wireshark Capture on VM2 shows the SSH tunnel formed between VM1 and VM2 being used by VM1 to access www.wikipedia.org



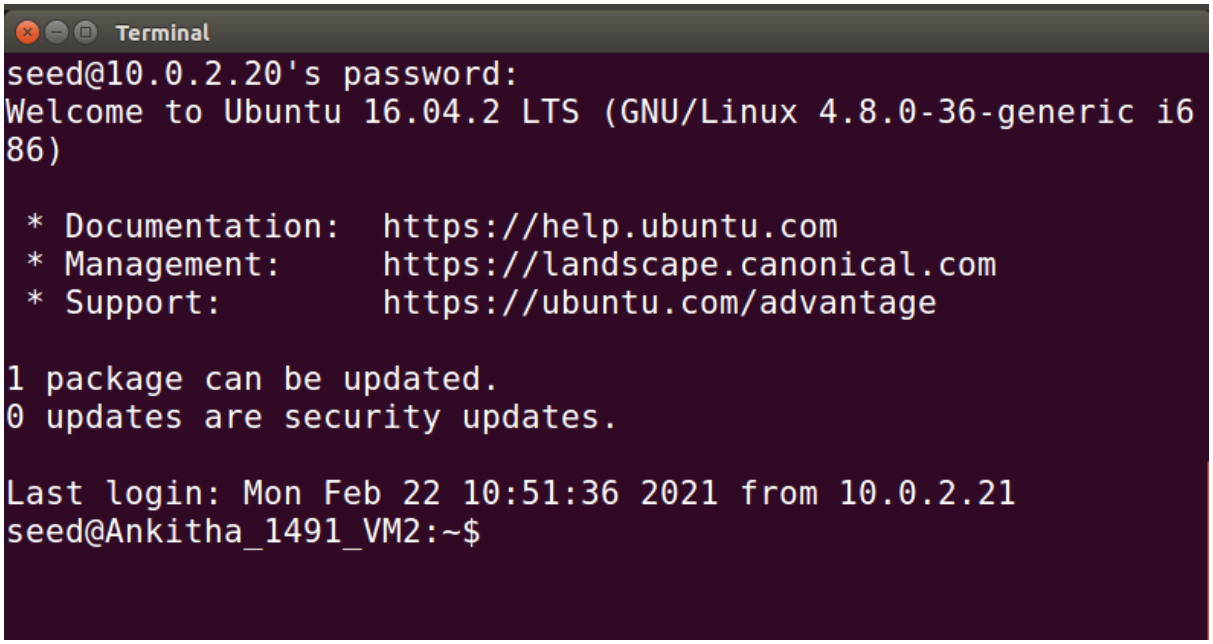
We disable the ssh tunnel using the `exit` command and clear the browser cache and try to access the site again.



We are unable to because the browser is still configured to use proxy and with the tunnel not running the browser cannot use local port 9000 to access the site.

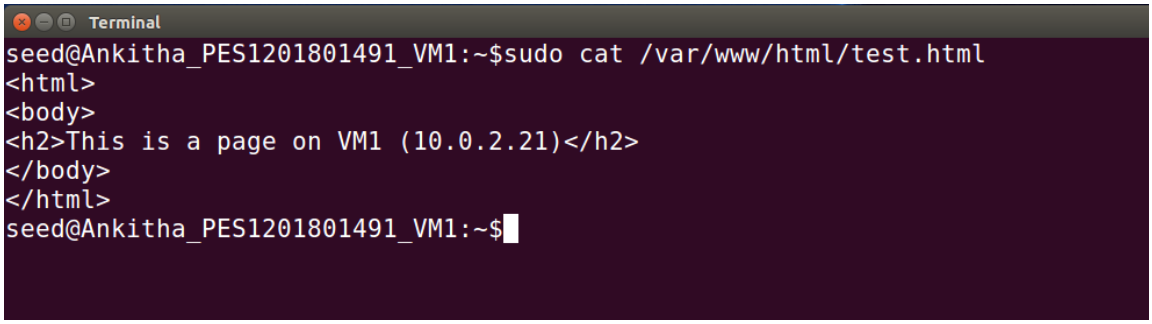


If we re-enable the ssh tunnel using ssh -D 9000 seed@10.0.2.13 command, we are able to access the website evading the firewall rule which is set as shown below.

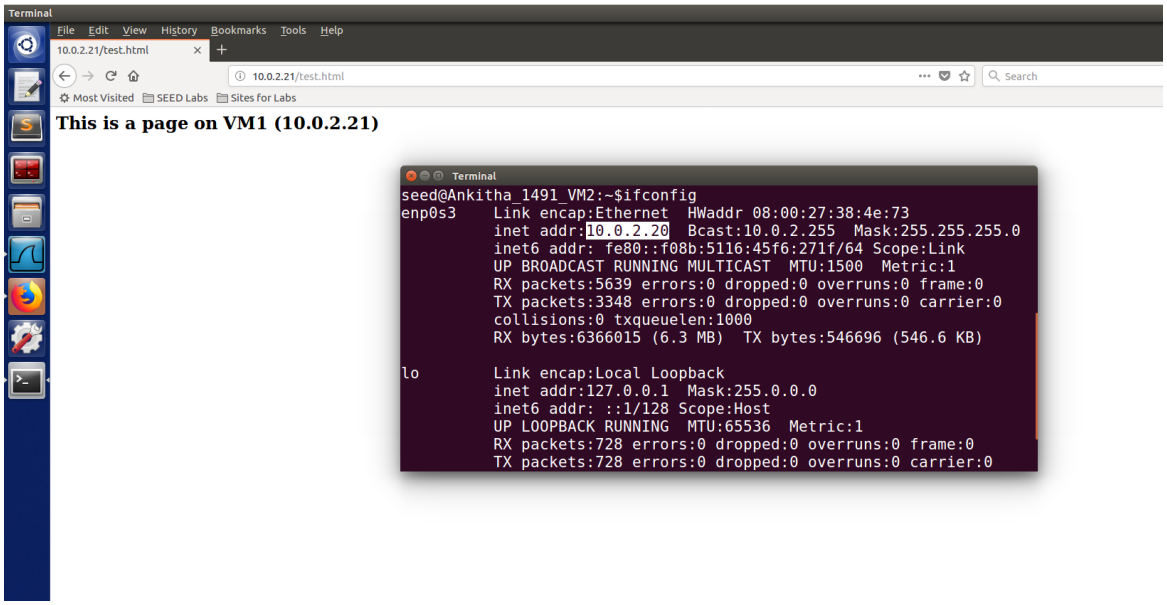


Task 4 : Evade Ingress Filtering

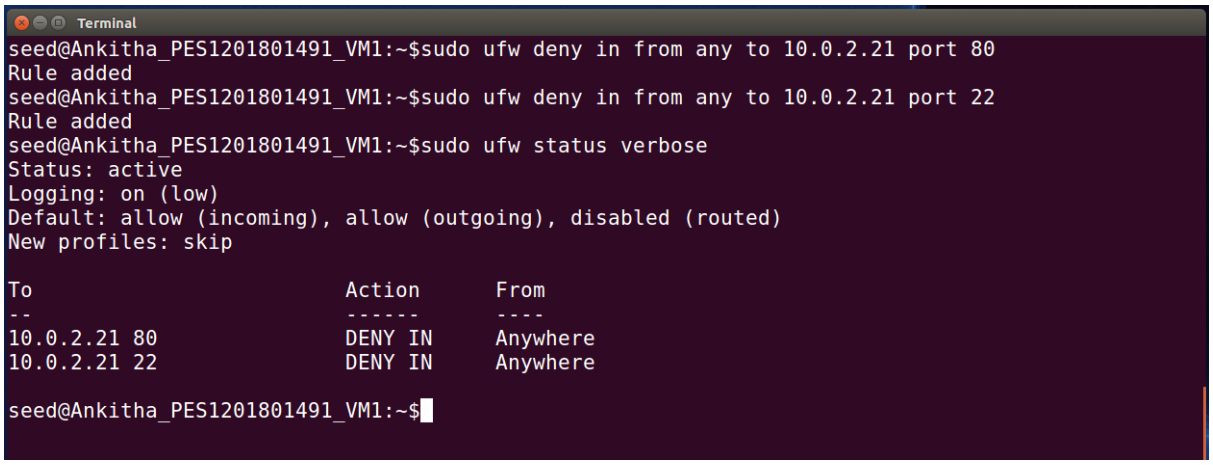
We create a html file called test.html on VM1.



The below screenshot shows that VM2 can access the secret page before updating the firewall rules in VM1.

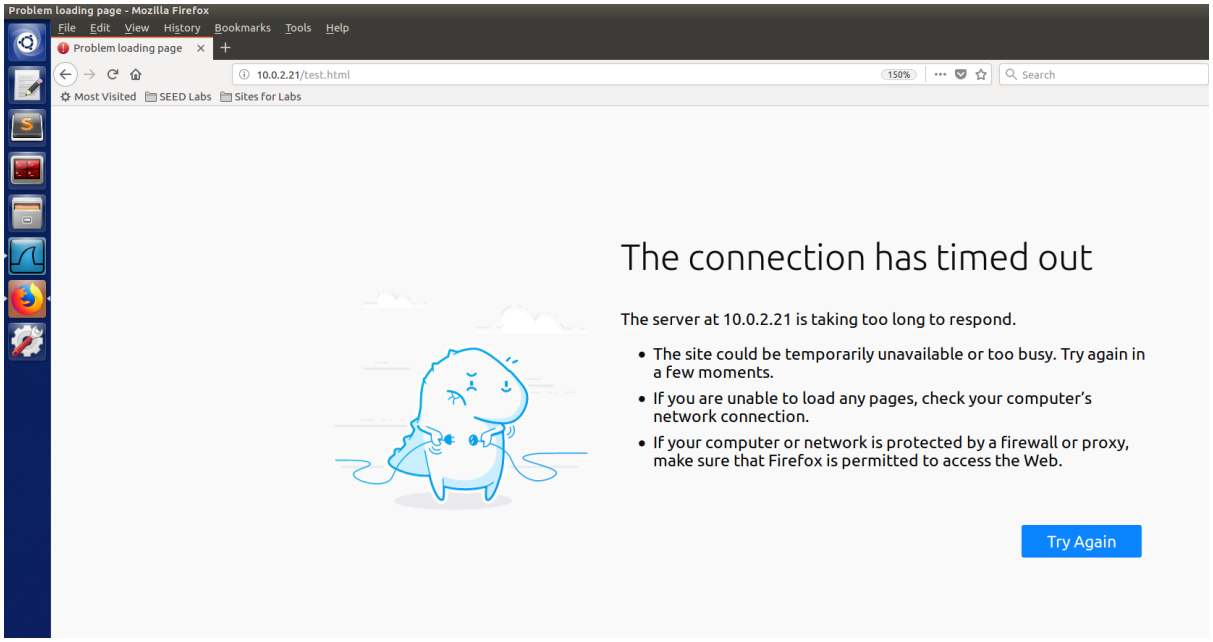


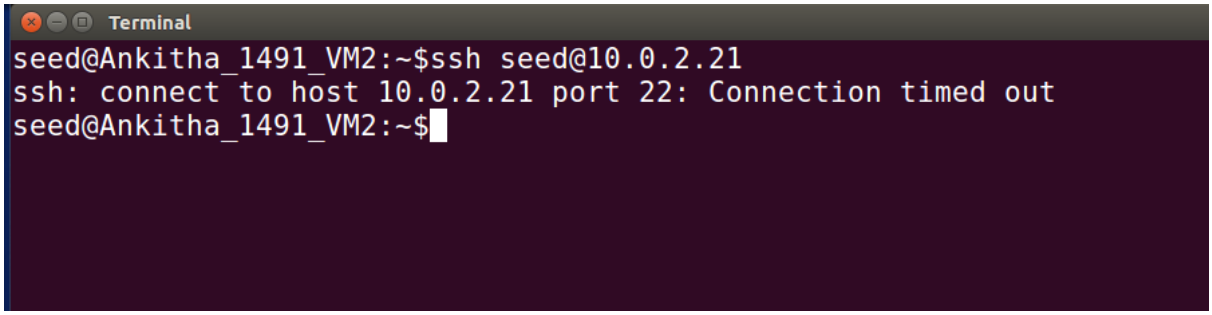
We next block incoming requests on port 80 and port 22 on VM1.



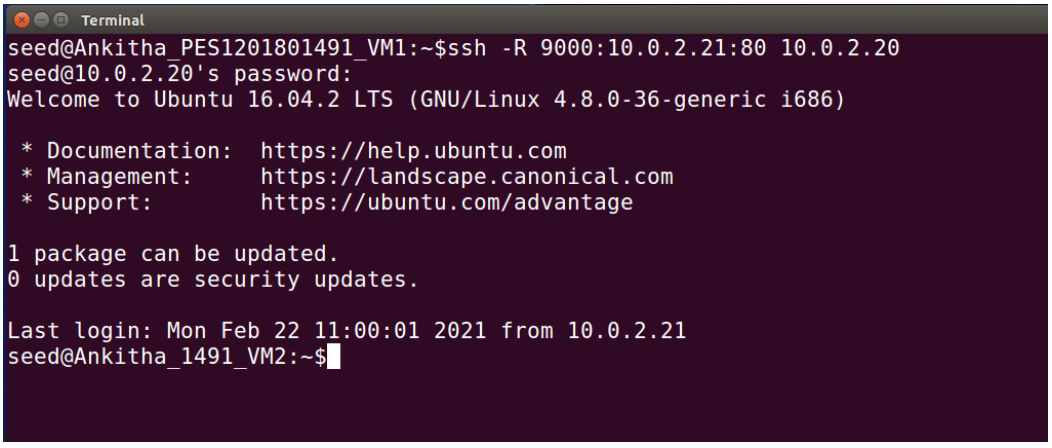
Once the firewall rules are updated, the secret page does not load in VM2 since the firewall blocks the web page requests.

Similarly, we cannot ssh from VM2 to VM1 either as we are blocked by the firewall(port 22). So we cannot access test.html from VM2.

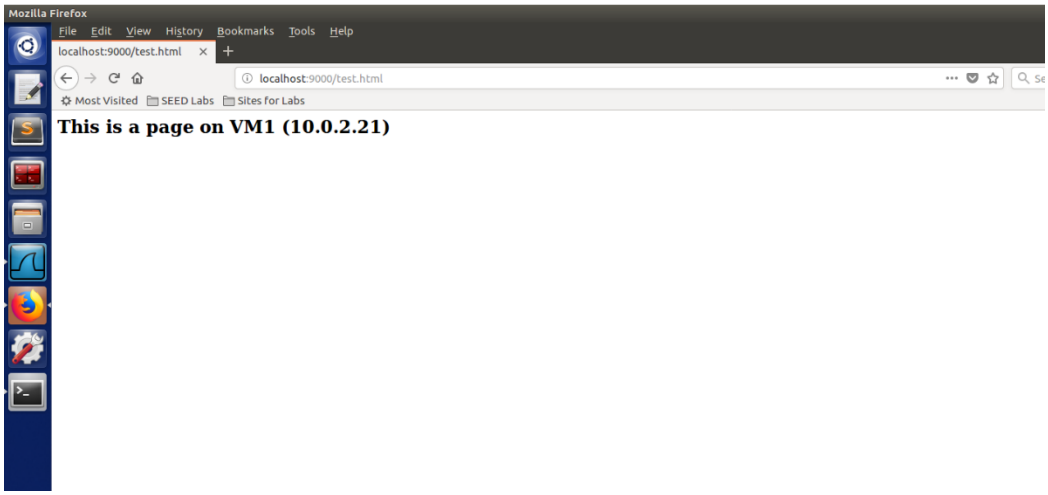




A reverse shell is setup in VM1 which directs requests to VM2 (10.0.2.20 @ 9000) .Once the reverse shell is setup, the web page loads again in the web browser.



With the tunnel setup, we test whether we can access the webpage using port 9000 on VM2 via the browser. The page is successfully retrieved.



Once the reverse shell is stopped and web browser is reloaded, the webpage fails to load due to the broken tunnel.