Lab 2

CSE-644 INTERNET SECURITY

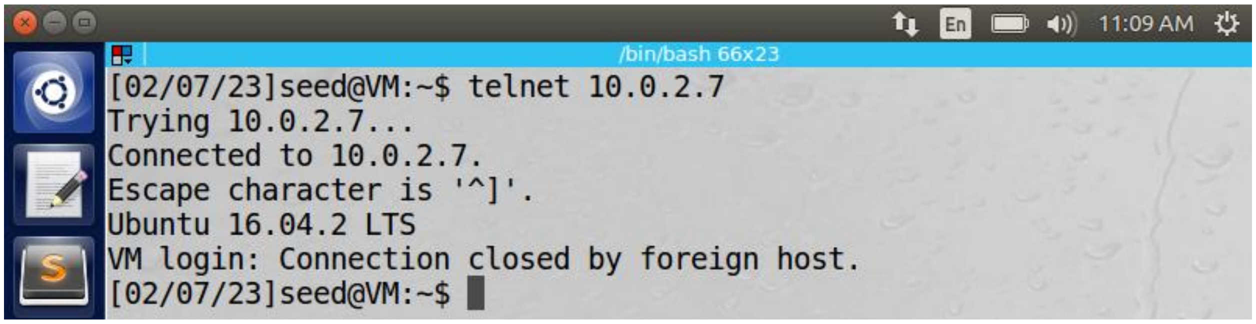
DR. SYED SHAZLI

2/8/2023

Anthony Redamonti
SYRACUSE UNIVERSITY

Task 1: Using Firewall

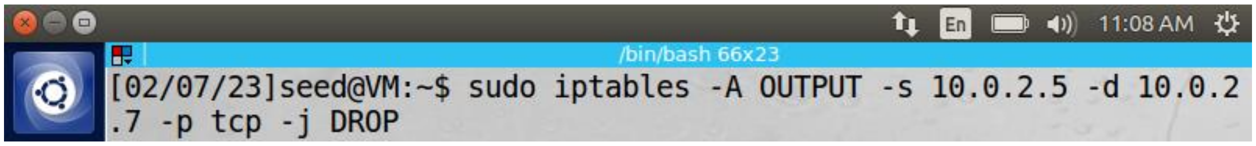
Machine A (IP 10.0.2.5) is on the same network as machine B (10.0.2.7). Below demonstrates a successful telnet command from machine A to machine B.



```
[02/07/23]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: Connection closed by foreign host.
[02/07/23]seed@VM:~$
```

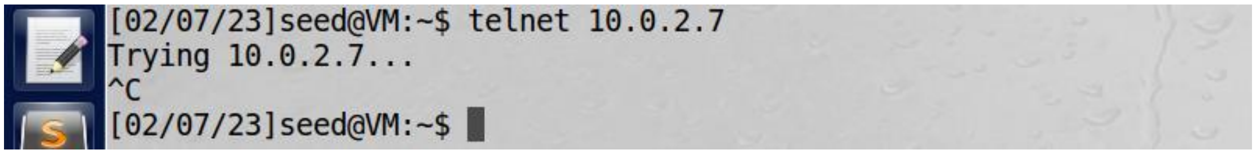
The following command was used to add a rule to the Linux iptables of machine A:

"sudo iptables -A OUTPUT -s 10.0.2.5 -d 10.0.2.7"



```
[02/07/23]seed@VM:~$ sudo iptables -A OUTPUT -s 10.0.2.5 -d 10.0.2
.7 -p tcp -j DROP
```

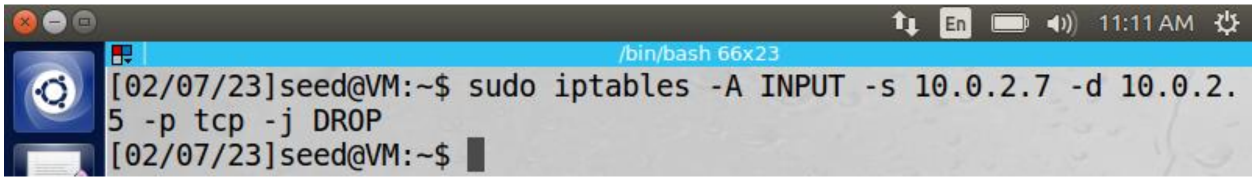
The rule prevented machine A from telnetting into machine B (egress).



```
[02/07/23]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
^C
[02/07/23]seed@VM:~$
```

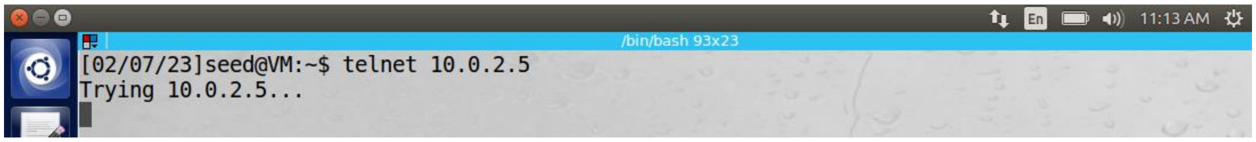
The following command was used to add a rule to the Linux iptables of machine A:

"sudo iptables -A INPUT -s 10.0.2.7 -d 10.0.2.5 -p tcp -j DROP"



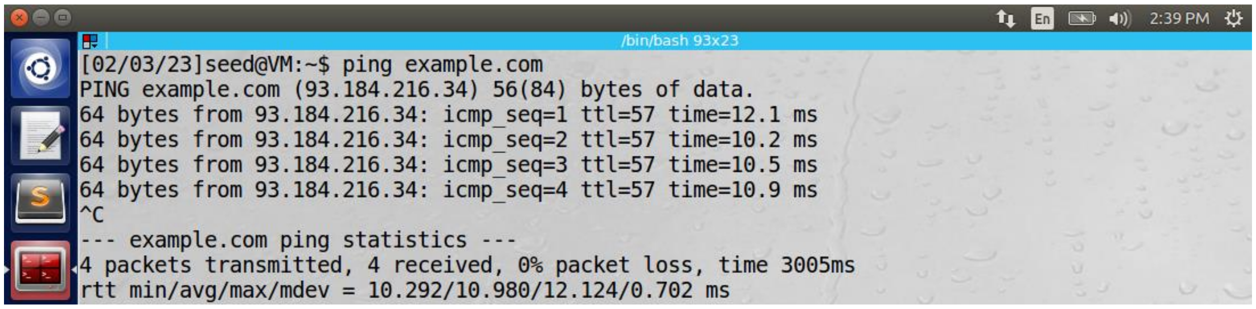
```
[02/07/23]seed@VM:~$ sudo iptables -A INPUT -s 10.0.2.7 -d 10.0.2.
5 -p tcp -j DROP
[02/07/23]seed@VM:~$
```

The rule prevents machine B from telnetting into machine A (ingress).



```
[02/07/23]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
```

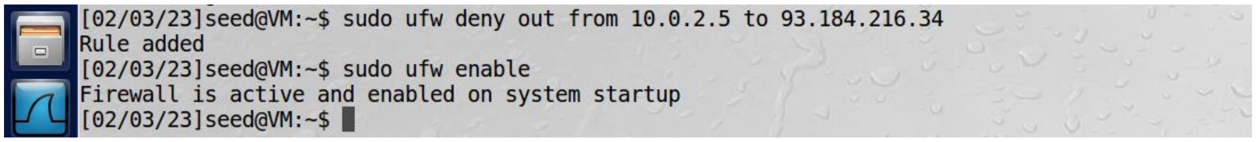
Machine A could successfully send a PING request to www.example.com.



```
[02/03/23]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=12.1 ms
64 bytes from 93.184.216.34: icmp_seq=2 ttl=57 time=10.2 ms
64 bytes from 93.184.216.34: icmp_seq=3 ttl=57 time=10.5 ms
64 bytes from 93.184.216.34: icmp_seq=4 ttl=57 time=10.9 ms
^C
--- example.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 10.292/10.980/12.124/0.702 ms
```

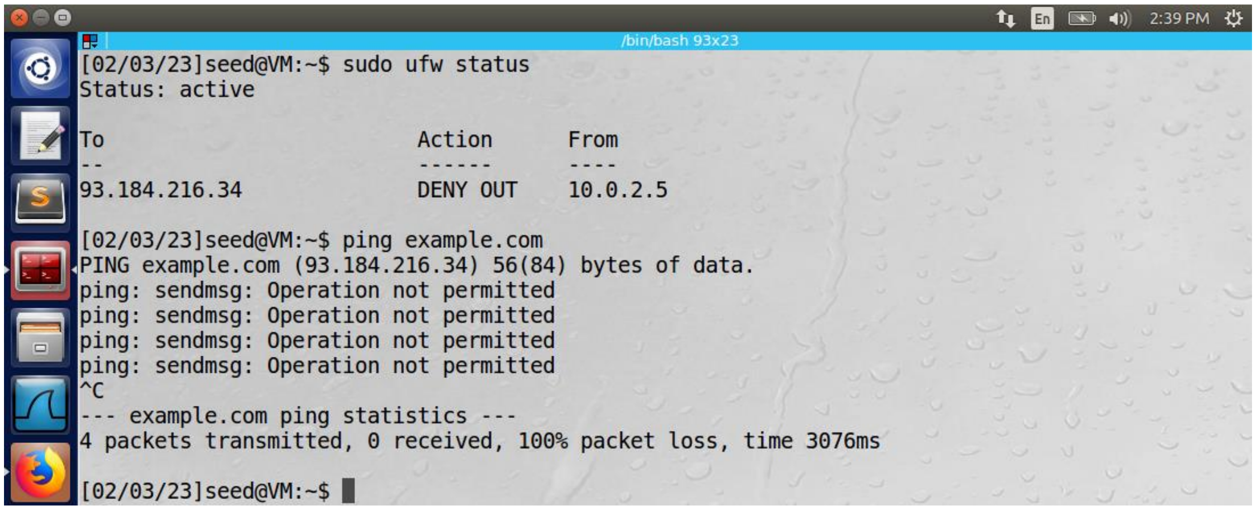
The following command was used to add a rule to the Linux iptables of machine A:

“sudo ufw deny out from 10.0.2.5 to 93.184.216.34” followed by “sudo ufw enable”



```
[02/03/23]seed@VM:~$ sudo ufw deny out from 10.0.2.5 to 93.184.216.34
Rule added
[02/03/23]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[02/03/23]seed@VM:~$
```

The PING command from machine A to www.example.com is no longer permitted.



```
[02/03/23]seed@VM:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
93.184.216.34              DENY OUT    10.0.2.5

[02/03/23]seed@VM:~$ ping example.com
PING example.com (93.184.216.34) 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
^C
--- example.com ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3076ms

[02/03/23]seed@VM:~$
```

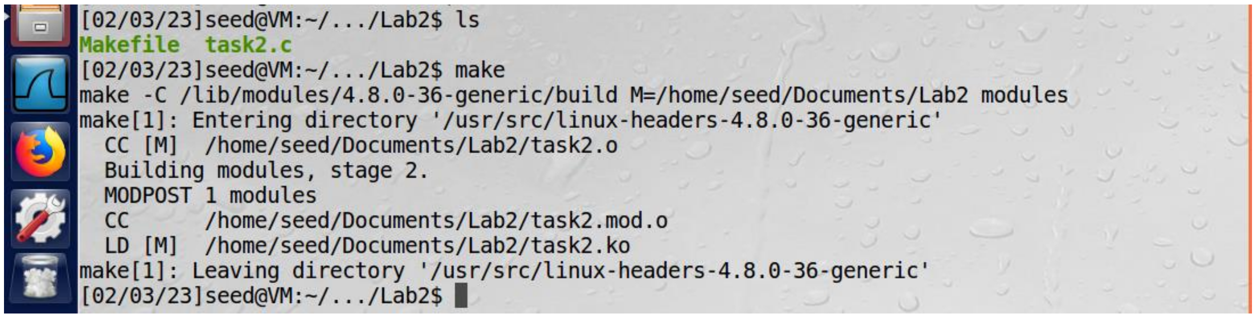
Observation: After the rules were added to the iptables, the Linux kernel used them to block ingress or egress traffic. The INPUT keyword is used to specify ingress traffic, and the OUTPUT keyword is used to specify egress traffic.
Explanation: The sudo keyword must be used to grant access to the iptables. The destination and source IP addresses were specified in each command. The UFW resides in the front-end of the iptables. It was used to block the egress traffic to www.example.com. UFW uses a slightly different syntax than the “iptables” commands. It must be activated using the “sudo ufw enable” command.

Task 2: Implementing a Simple Firewall

The file, “task2.c” is in the appendix section of the lab. It contains the C code to implement a loadable kernel module (LKM) using the Netfilter mechanism. The LKM allows the addition of firewall rules at specific points in the operating system without needing to recompile the kernel.

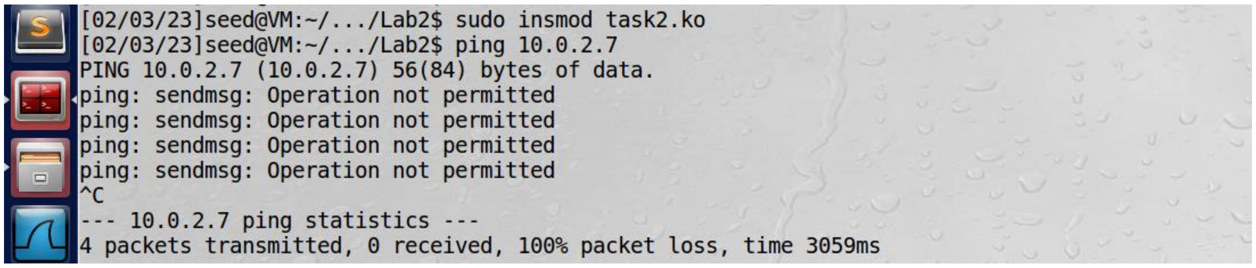
Rule 1: Block all ICMP echo requests to 10.0.2.7.
Rule 2: Block all ICMP echo requests from 10.0.2.7.
Rule 3: Block all telnet commands to 10.0.2.7.
Rule 4: Block all telnet commands from 10.0.2.7.
Rule 5: Block all telnet commands from 10.0.2.6.

The make file “Makefile” was used to compile the task2.c file, build the module, and create the task2.ko file.



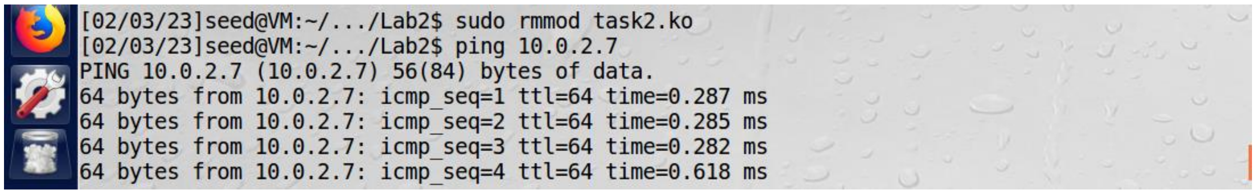
```
[02/03/23]seed@VM:~/.../Lab2$ ls
Makefile  task2.c
[02/03/23]seed@VM:~/.../Lab2$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/Documents/Lab2 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/Documents/Lab2/task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/Documents/Lab2/task2.mod.o
  LD [M]  /home/seed/Documents/Lab2/task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[02/03/23]seed@VM:~/.../Lab2$
```

After running the make file, the module is ready to be inserted into the kernel using the “sudo insmod task2.ko” command.



```
[02/03/23]seed@VM:~/.../Lab2$ sudo insmod task2.ko
[02/03/23]seed@VM:~/.../Lab2$ ping 10.0.2.7
PING 10.0.2.7 (10.0.2.7) 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
^C
--- 10.0.2.7 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3059ms
```

Notice that pinging 10.0.2.7 is not permitted (rule 1). To remove the module from the kernel, use the “sudo rmmod task2.ko” command.

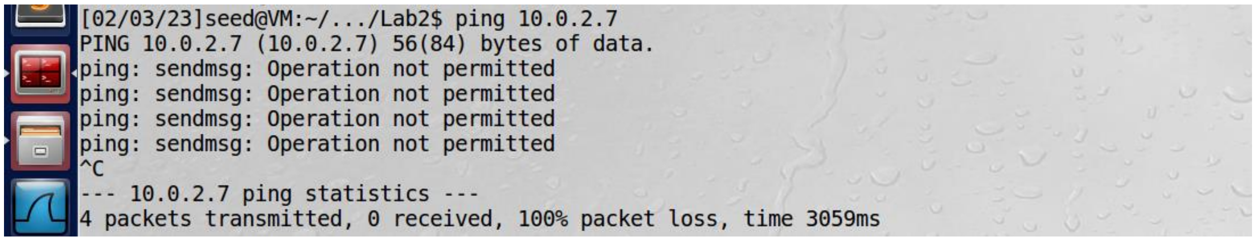


```
[02/03/23]seed@VM:~/.../Lab2$ sudo rmmod task2.ko
[02/03/23]seed@VM:~/.../Lab2$ ping 10.0.2.7
PING 10.0.2.7 (10.0.2.7) 56(84) bytes of data.
64 bytes from 10.0.2.7: icmp_seq=1 ttl=64 time=0.287 ms
64 bytes from 10.0.2.7: icmp_seq=2 ttl=64 time=0.285 ms
64 bytes from 10.0.2.7: icmp_seq=3 ttl=64 time=0.282 ms
64 bytes from 10.0.2.7: icmp_seq=4 ttl=64 time=0.618 ms
```

Notice that pinging 10.0.2.7 is now permitted because the module is not inserted into the kernel.

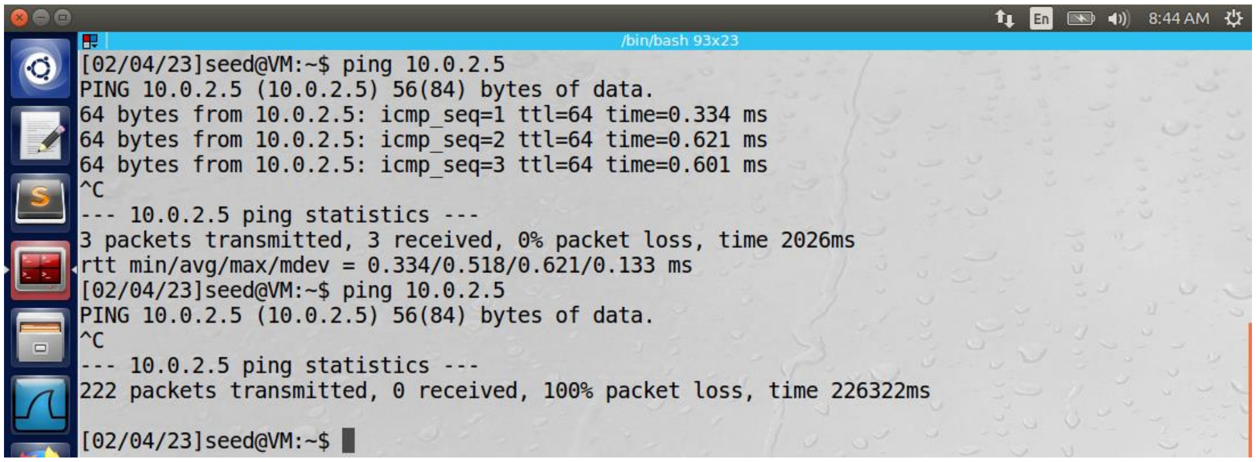
After inserting the module into the kernel, all five of the firewall rules were enforced.

Rule 1: Block all ICMP echo requests to 10.0.2.7.



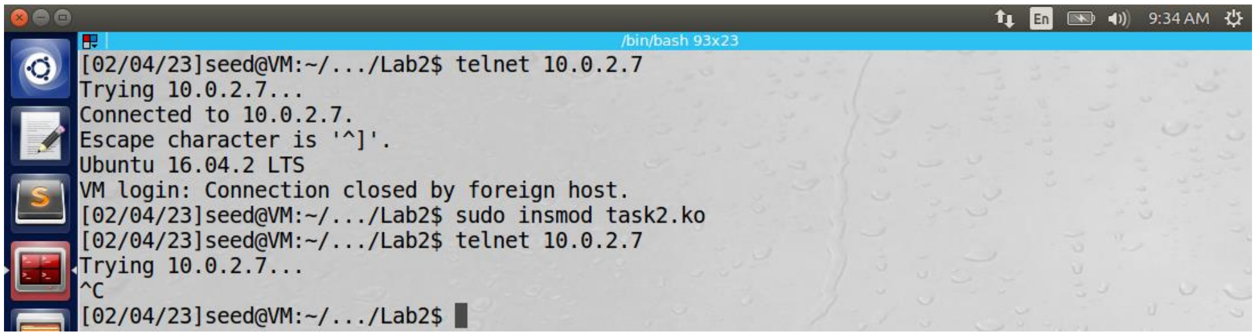
```
[02/03/23]seed@VM:~/.../Lab2$ ping 10.0.2.7
PING 10.0.2.7 (10.0.2.7) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 10.0.2.7 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3059ms
```

Rule 2: Block all ICMP echo requests from 10.0.2.7.



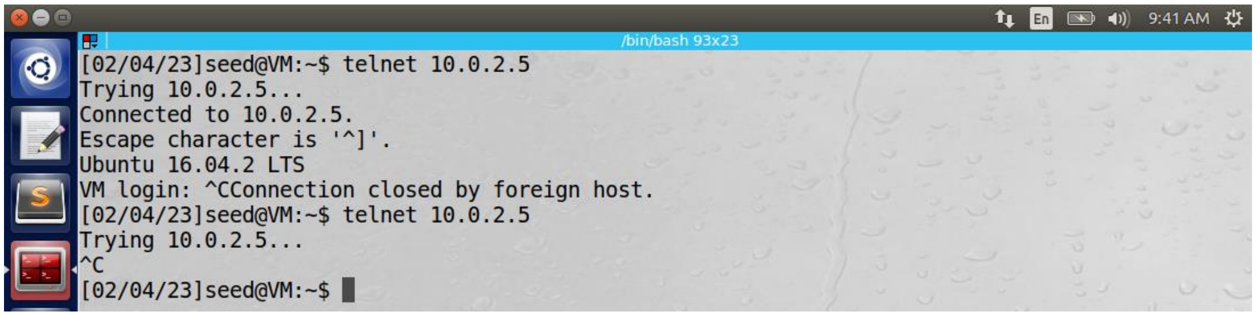
```
[02/04/23]seed@VM:~$ ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.334 ms
64 bytes from 10.0.2.5: icmp_seq=2 ttl=64 time=0.621 ms
64 bytes from 10.0.2.5: icmp_seq=3 ttl=64 time=0.601 ms
^C
--- 10.0.2.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.334/0.518/0.621/0.133 ms
[02/04/23]seed@VM:~$ ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
^C
--- 10.0.2.5 ping statistics ---
222 packets transmitted, 0 received, 100% packet loss, time 226322ms

[02/04/23]seed@VM:~$ 
```

Rule 3: Block all telnet commands to 10.0.2.7.



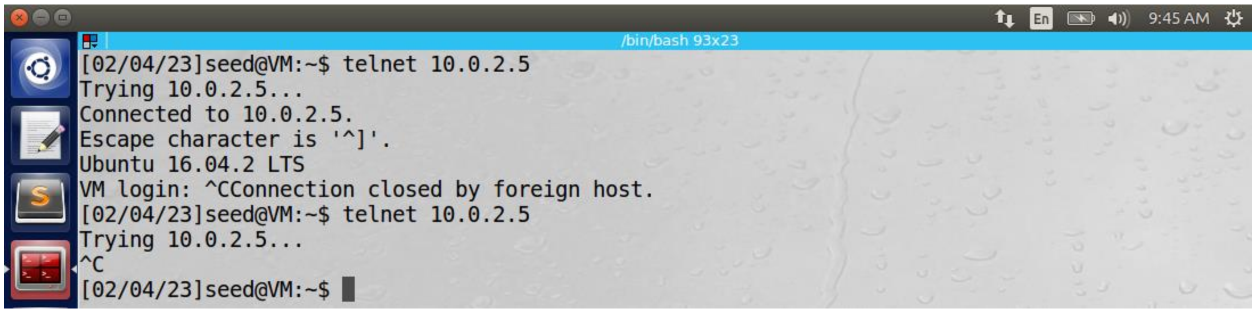
```
[02/04/23]seed@VM:~/.../Lab2$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: Connection closed by foreign host.
[02/04/23]seed@VM:~/.../Lab2$ sudo insmod task2.ko
[02/04/23]seed@VM:~/.../Lab2$ telnet 10.0.2.7
Trying 10.0.2.7...
^C
[02/04/23]seed@VM:~/.../Lab2$ 
```

Rule 4: Block all telnet commands from 10.0.2.7.



```
[02/04/23]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: ^CConnection closed by foreign host.
[02/04/23]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
^C
[02/04/23]seed@VM:~$ 
```

Rule 5: Block all telnet commands from 10.0.2.6.



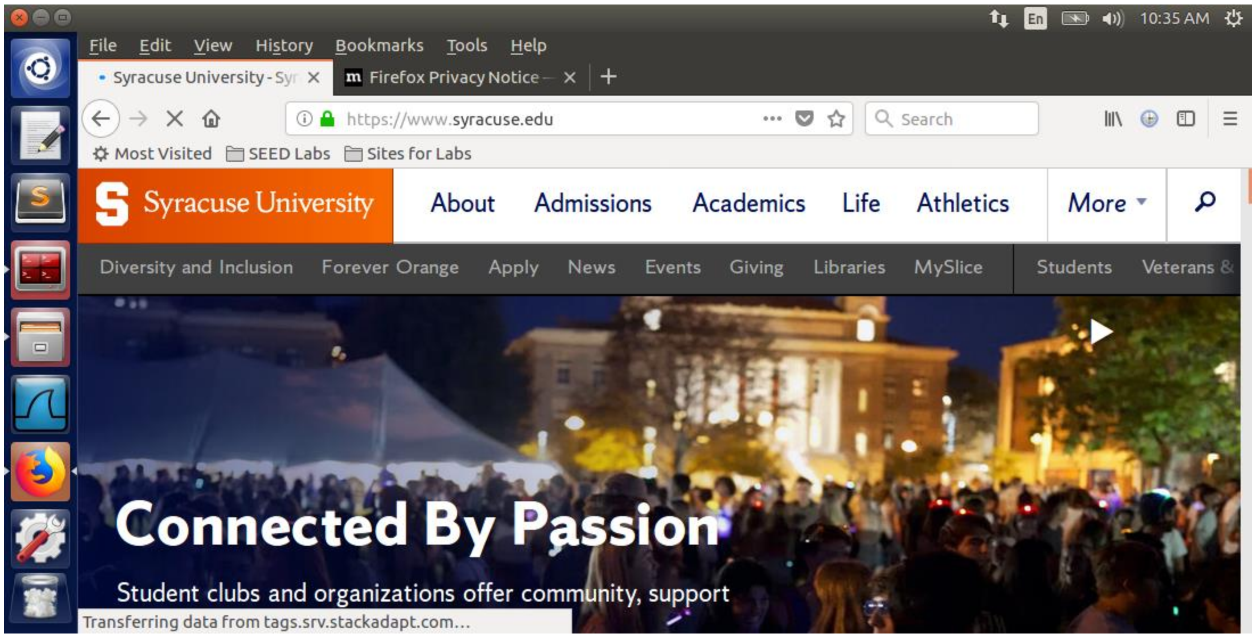
```
[02/04/23]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: ^CConnection closed by foreign host.
[02/04/23]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
^C
[02/04/23]seed@VM:~$
```

Observation: The use of hooks is used to insert these rules into specific places along the packet paths. The firewall can run the packets through the firewall after the module has been built and inserted into the kernel.

Explanation: A Loadable Kernel Module (LKM) is very convenient when creating firewall rules. The convenience is that the kernel does not need to be recompiled after the rules are inserted. In essence, the Linux kernel does not need to be modified to implement new firewall rules. Note: at the beginning of the Netfilter C application, each hook must be registered using the nf_register_net_hook() function, which registers the hook and all its contents into the module. At the end of the Netfilter C application, the nf_unregister_net_hook() function needs to be called to remove the hooks from the module. In the hooknum data member of the hook class, use NF_INET_LOCAL_OUT to specify egress traffic and NF_INET_LOCAL_IN to specify ingress traffic. Use the hook.hook() method to hook a custom function that parses the packet and returns either NF_ACCEPT or NF_DROP. The iph->saddr is used in the custom function to specify the source address of the IP header and iph->daddr is used to specify the destination address in the IP header.

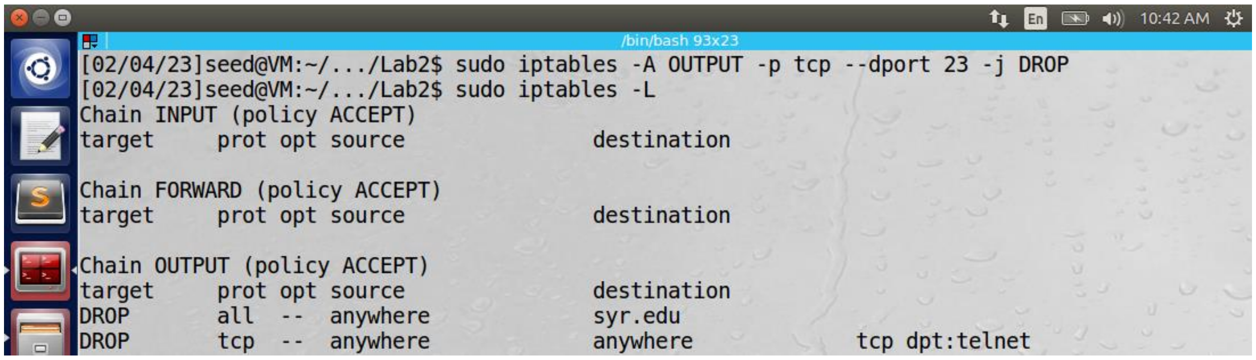
Task 3: Evading Egress Filter

Before implementing the firewall rules, machine A could telnet to other machines on its network and reach “www.syr.edu” via the internet.



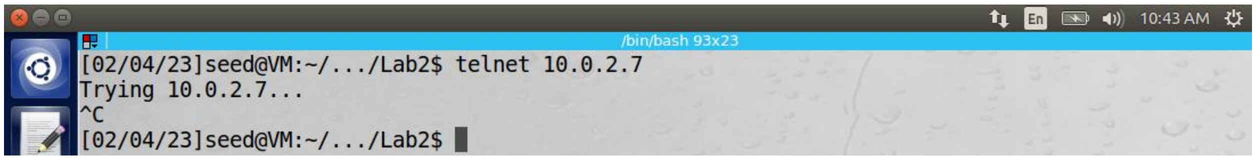
The following rule was implemented on machine A to block all outgoing traffic to telnet servers:

“sudo iptables -A OUTPUT -p tcp --dport 23 -j DROP”



```
[02/04/23]seed@VM:~/.../Lab2$ sudo iptables -A OUTPUT -p tcp --dport 23 -j DROP
[02/04/23]seed@VM:~/.../Lab2$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             syr.edu
DROP       tcp  --  anywhere             anywhere             tcp dpt:telnet
```

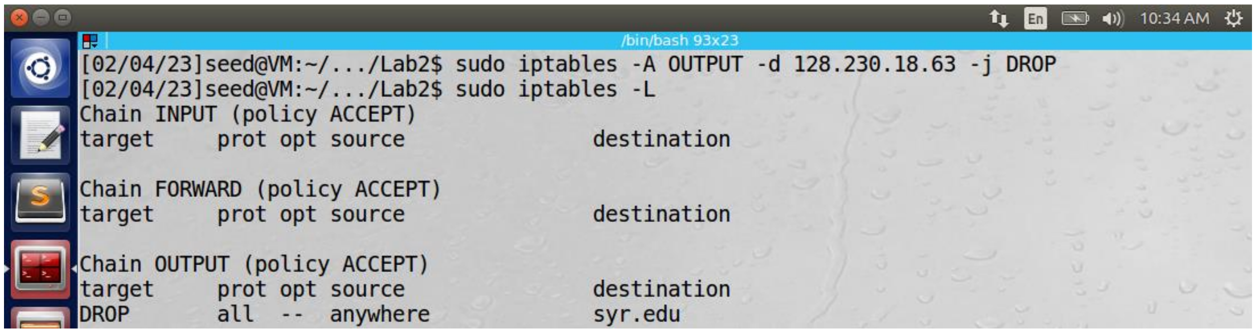
The rule prevented the machine from telnetting to machine B.



```
[02/04/23]seed@VM:~/.../Lab2$ telnet 10.0.2.7
Trying 10.0.2.7...
^C
[02/04/23]seed@VM:~/.../Lab2$ 
```

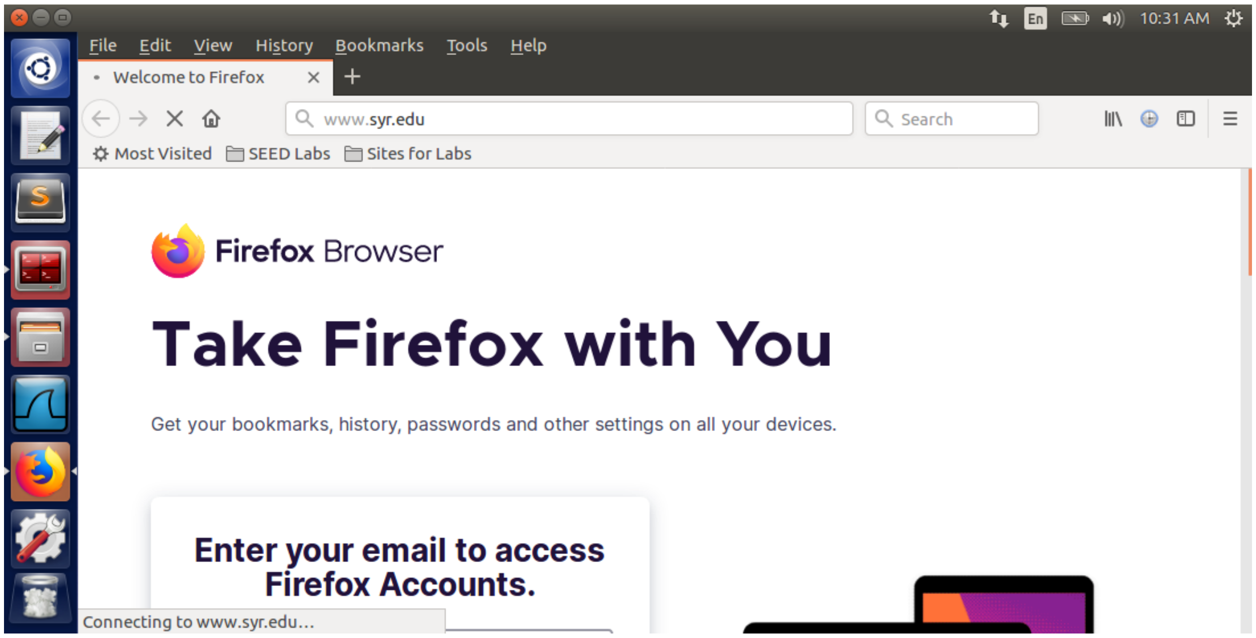
The following rule was implemented to prevent machine A from accessing “www.syr.edu”:

“sudo iptables -A OUTPUT -d 128.230.18.63 -j DROP”



```
[02/04/23]seed@VM:~/.../Lab2$ sudo iptables -A OUTPUT -d 128.230.18.63 -j DROP
[02/04/23]seed@VM:~/.../Lab2$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             syr.edu
```

The dig command was used to retrieve the IP address of www.syr.edu. Then the IP address was used as the destination address to target in the firewall rule blocking egress traffic. After the rule is inserted into the iptables, www.syr.edu was no longer reachable via the internet.


File Edit View History Bookmarks Tools Help
Welcome to Firefox
www.syr.edu
Search
Most Visited
SEED Labs
Sites for Labs
Firefox Browser
Take Firefox with You
Get your bookmarks, history, passwords and other settings on all your devices.
Enter your email to access Firefox Accounts.
Connecting to www.syr.edu...

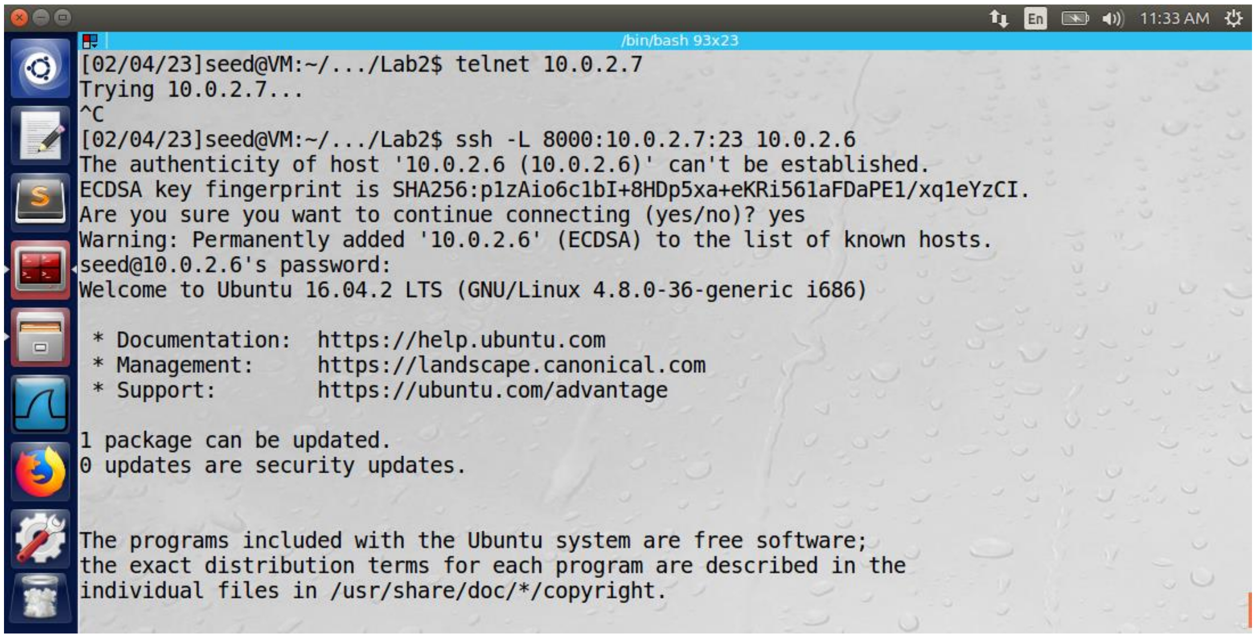
Part A: Telnet to Machine B through the firewall

To bypass the firewall, an SSH tunnel was established from machine A to machine B. The SSH tunnel encrypts the packets so that the firewall rules will no longer block TCP packets using port 23 (telnet commands).

The following command was used to set up the SSH tunnel:

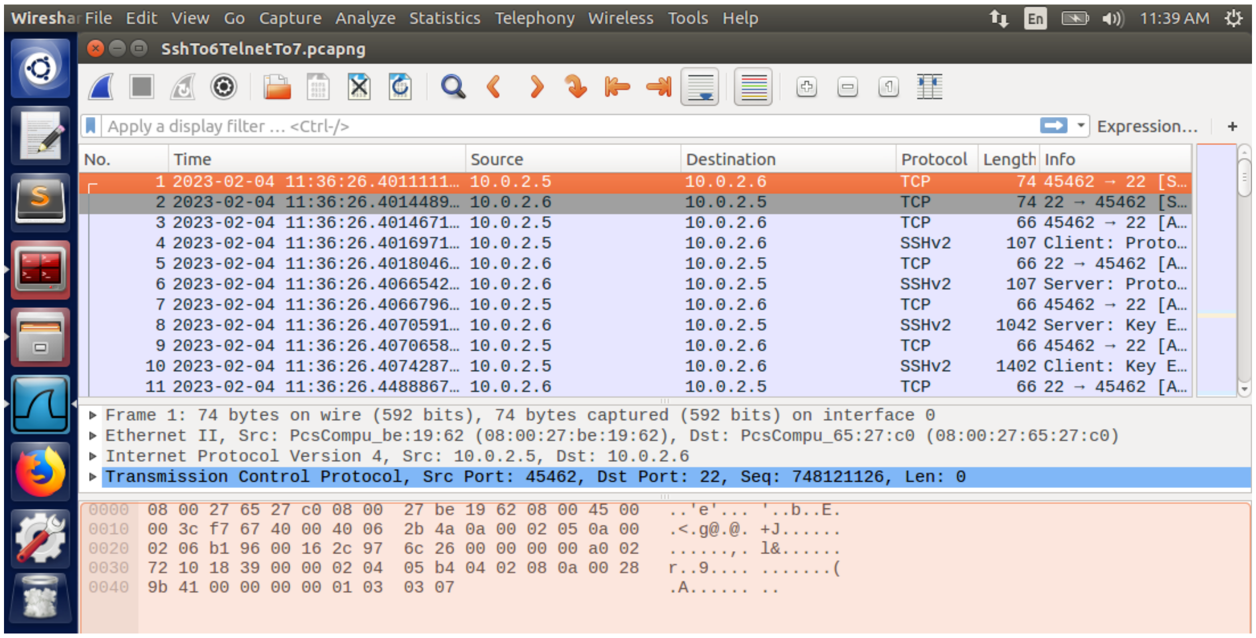
"ssh -L 8000:10.0.2.7:23 10.0.2.6"

An SSH tunnel is created with port 8000 of the localhost (machine A) and 10.0.2.6 (machine B). By creating an SSH tunnel with this port, all the TCP packets will be encrypted and sent to 10.0.2.6. The traffic is then transferred from machine B to machine C (10.0.2.7) via port 23 (telnet).



```
[02/04/23]seed@VM:~/.../Lab2$ telnet 10.0.2.7
Trying 10.0.2.7...
^C
[02/04/23]seed@VM:~/.../Lab2$ ssh -L 8000:10.0.2.7:23 10.0.2.6
The authenticity of host '10.0.2.6 (10.0.2.6)' 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 '10.0.2.6' (ECDSA) to the list of known hosts.
seed@10.0.2.6'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.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

A Wireshark log of the network traffic from machine A is below.


Wiresha File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
11:39 AM
SshTo6TelnetTo7.pcapng
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Time                         Source     Destination  Protocol  Length Info
   1 2023-02-04 11:36:26.4011111… 10.0.2.5   10.0.2.6     TCP          74 45462 → 22 [S…
   2 2023-02-04 11:36:26.4014489… 10.0.2.6   10.0.2.5     TCP          74 22 → 45462 [S…
   3 2023-02-04 11:36:26.4014671… 10.0.2.5   10.0.2.6     TCP          66 45462 → 22 [A…
   4 2023-02-04 11:36:26.4016971… 10.0.2.5   10.0.2.6     SSHv2       107 Client: Proto…
   5 2023-02-04 11:36:26.4018046… 10.0.2.6   10.0.2.5     TCP          66 22 → 45462 [A…
   6 2023-02-04 11:36:26.4066542… 10.0.2.6   10.0.2.5     SSHv2       107 Server: Proto…
   7 2023-02-04 11:36:26.4066796… 10.0.2.5   10.0.2.6     TCP          66 45462 → 22 [A…
   8 2023-02-04 11:36:26.4070591… 10.0.2.6   10.0.2.5     SSHv2      1042 Server: Key E…
   9 2023-02-04 11:36:26.4070658… 10.0.2.5   10.0.2.6     TCP          66 45462 → 22 [A…
  10 2023-02-04 11:36:26.4074287… 10.0.2.5   10.0.2.6     SSHv2      1402 Client: Key E…
  11 2023-02-04 11:36:26.4488867… 10.0.2.6   10.0.2.5     TCP          66 22 → 45462 [A…

▸ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_be:19:62 (08:00:27:be:19:62), Dst: PcsCompu_65:27:c0 (08:00:27:65:27:c0)
▸ Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.6
▸ Transmission Control Protocol, Src Port: 45462, Dst Port: 22, Seq: 748121126, Len: 0

0000  08 00 27 65 27 c0 08 00  27 be 19 62 08 00 45 00   ..'e'... '..b..E.
0010  00 3c f7 67 40 00 40 06  2b 4a 0a 00 02 05 0a 00   .<.g@.@. +J......
0020  02 06 b1 96 00 16 2c 97  6c 26 00 00 00 00 a0 02   ......,. l&......
0030  72 10 18 39 00 00 02 04  05 b4 04 02 08 0a 00 28   r..9.... .......(
0040  9b 41 00 00 00 00 01 03  03 07                     .A...... ..
```

Observation: The Wireshark log displays the packets using the SSHv2 protocol between the client (machine A) and the server (machine B). Between each SSHv2 message is an SSH tunneling packet (TCP) between port 22 or machine B and port 45462 of machine A.

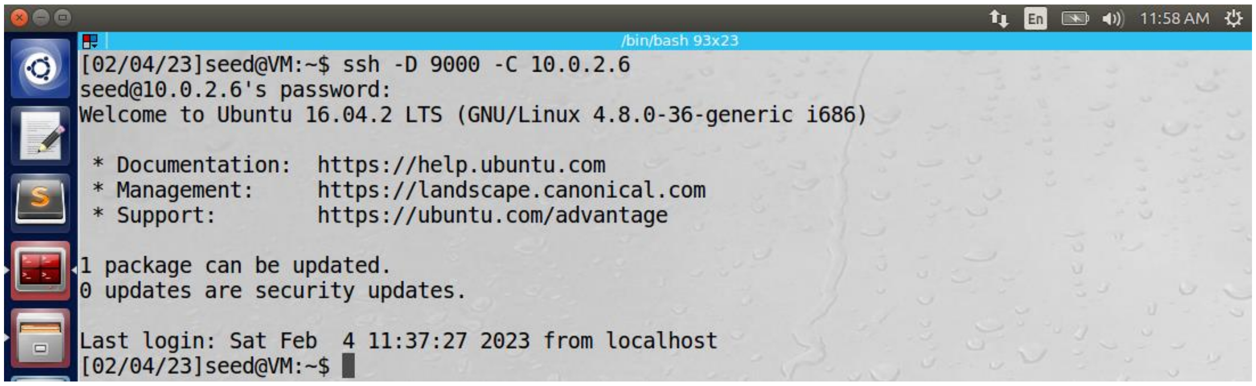
Explanation: By encrypting the packets using an SSH tunnel with machine B, machine A was able to successfully telnet into machine C from machine B.

Part B: Connect to WWW.SYR.EDU Using SSH Tunnel

The following command was used to establish an SSH tunnel between machine A and B to bypass the firewall on machine A:

"ssh -D 9000 -C 10.0.2.6"

The command sets up dynamic port forwarding to port 9000 on the localhost.

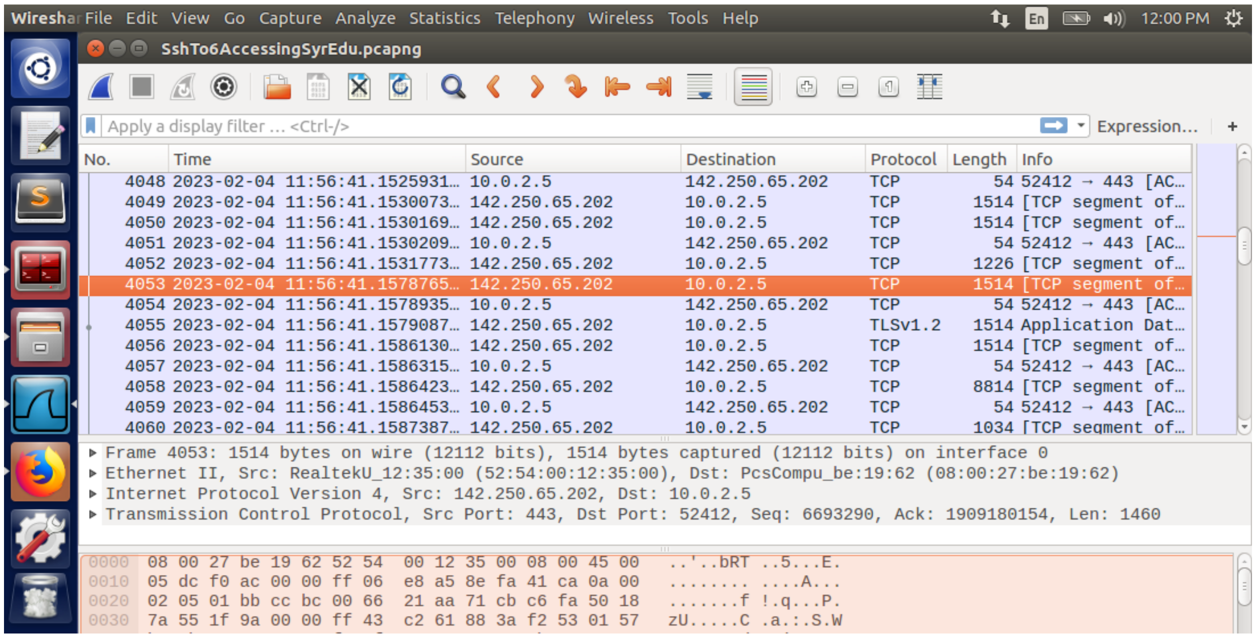


```
[02/04/23]seed@VM:~$ ssh -D 9000 -C 10.0.2.6
seed@10.0.2.6's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * 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: Sat Feb  4 11:37:27 2023 from localhost
[02/04/23]seed@VM:~$
```

Next, Firefox internet explorer needed to be configured to use port 9000 as its proxy server when attempting to connect to a web server. The lab instructions detail how to set up Firefox using the Edit -> Preferences menu.

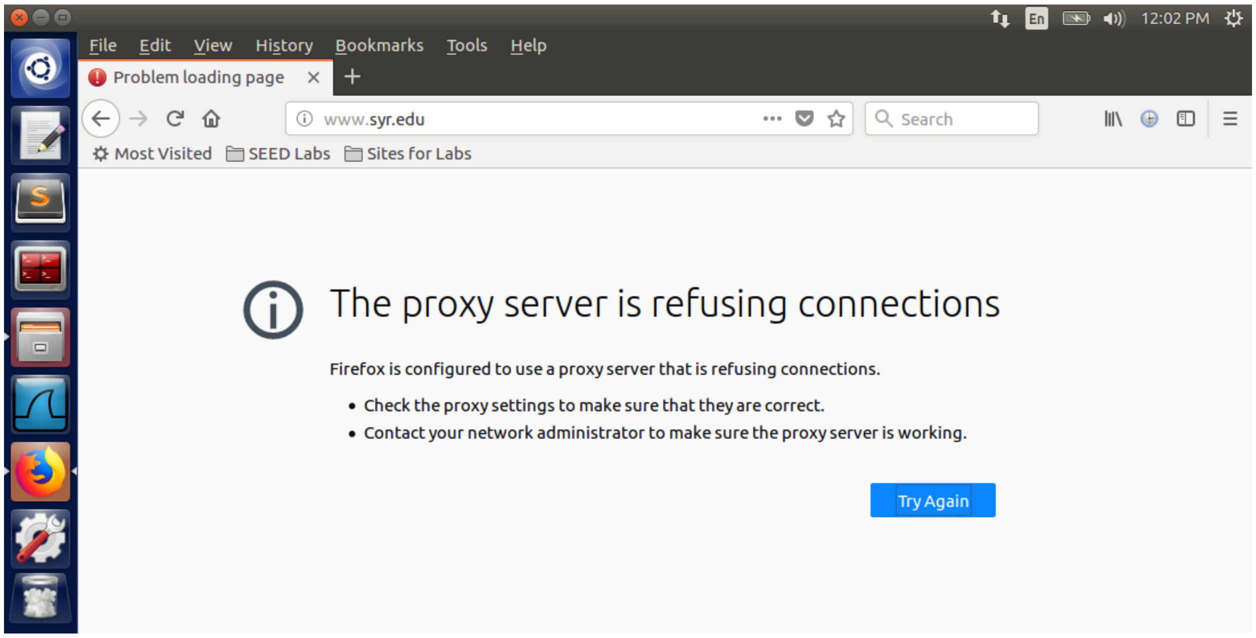
1. Firefox was used to successfully access www.syr.edu.

   A Wireshark log was created to display the packet content between machine A and the proxy server. The SSH tunnel establishes a connection between machine A (10.0.2.5) and machine B (IP 10.0.2.6) through the proxy server on port 9000. The SSH tunnel bypasses machine A's firewall. The Wireshark log below displays the TCP packets between the proxy server (IP 142.250.65.202) and machine A (IP 10.0.2.5).


Wireshar File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
12:00 PM
SshTo6AccessingSyrEdu.pcapng
Apply a display filter … <Ctrl-/>
Expression…

```
No.   Time                          Source          Destination     Protocol Length Info
 4048 2023-02-04 11:56:41.1525931… 10.0.2.5        142.250.65.202  TCP        54 52412 → 443 [AC…
 4049 2023-02-04 11:56:41.1530073… 142.250.65.202  10.0.2.5        TCP      1514 [TCP segment of…
 4050 2023-02-04 11:56:41.1530169… 142.250.65.202  10.0.2.5        TCP      1514 [TCP segment of…
 4051 2023-02-04 11:56:41.1530209… 10.0.2.5        142.250.65.202  TCP        54 52412 → 443 [AC…
 4052 2023-02-04 11:56:41.1531773… 142.250.65.202  10.0.2.5        TCP      1226 [TCP segment of…
 4053 2023-02-04 11:56:41.1578765… 142.250.65.202  10.0.2.5        TCP      1514 [TCP segment of…
 4054 2023-02-04 11:56:41.1578935… 10.0.2.5        142.250.65.202  TCP        54 52412 → 443 [AC…
 4055 2023-02-04 11:56:41.1579087… 142.250.65.202  10.0.2.5        TLSv1.2  1514 Application Dat…
 4056 2023-02-04 11:56:41.1586130… 142.250.65.202  10.0.2.5        TCP      1514 [TCP segment of…
 4057 2023-02-04 11:56:41.1586315… 10.0.2.5        142.250.65.202  TCP        54 52412 → 443 [AC…
 4058 2023-02-04 11:56:41.1586423… 142.250.65.202  10.0.2.5        TCP      8814 [TCP segment of…
 4059 2023-02-04 11:56:41.1586453… 10.0.2.5        142.250.65.202  TCP        54 52412 → 443 [AC…
 4060 2023-02-04 11:56:41.1587387… 142.250.65.202  10.0.2.5        TCP      1034 [TCP segment of…

▸ Frame 4053: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▸ Ethernet II, Src: RealtekU_12:35:00 (52:54:00:12:35:00), Dst: PcsCompu_be:19:62 (08:00:27:be:19:62)
▸ Internet Protocol Version 4, Src: 142.250.65.202, Dst: 10.0.2.5
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 52412, Seq: 6693290, Ack: 1909180154, Len: 1460

0000  08 00 27 be 19 62 52 54  00 12 35 00 08 00 45 00   ..'..bRT ..5...E.
0010  05 dc f0 ac 00 00 ff 06  e8 a5 8e fa 41 ca 0a 00   ........ ....A...
0020  02 05 01 bb cc bc 00 66  21 aa 71 cb c6 fa 50 18   .......f !.q...P.
0030  7a 55 1f 9a 00 00 ff 43  c2 61 88 3a f2 53 01 57   zU.....C .a.:.S.W
```

2. The SSH tunnel was then broken, and the Firefox cache was cleared. The connection with www.syr.edu was unsuccessful.

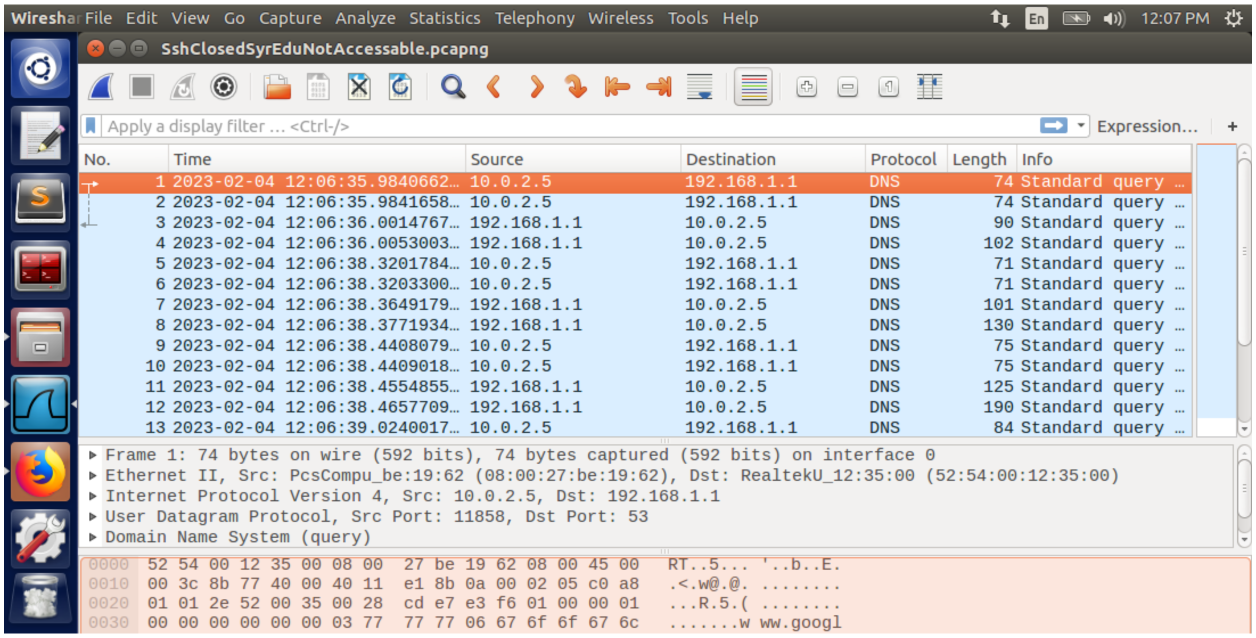


The proxy server is refusing connections

Firefox is configured to use a proxy server that is refusing connections.

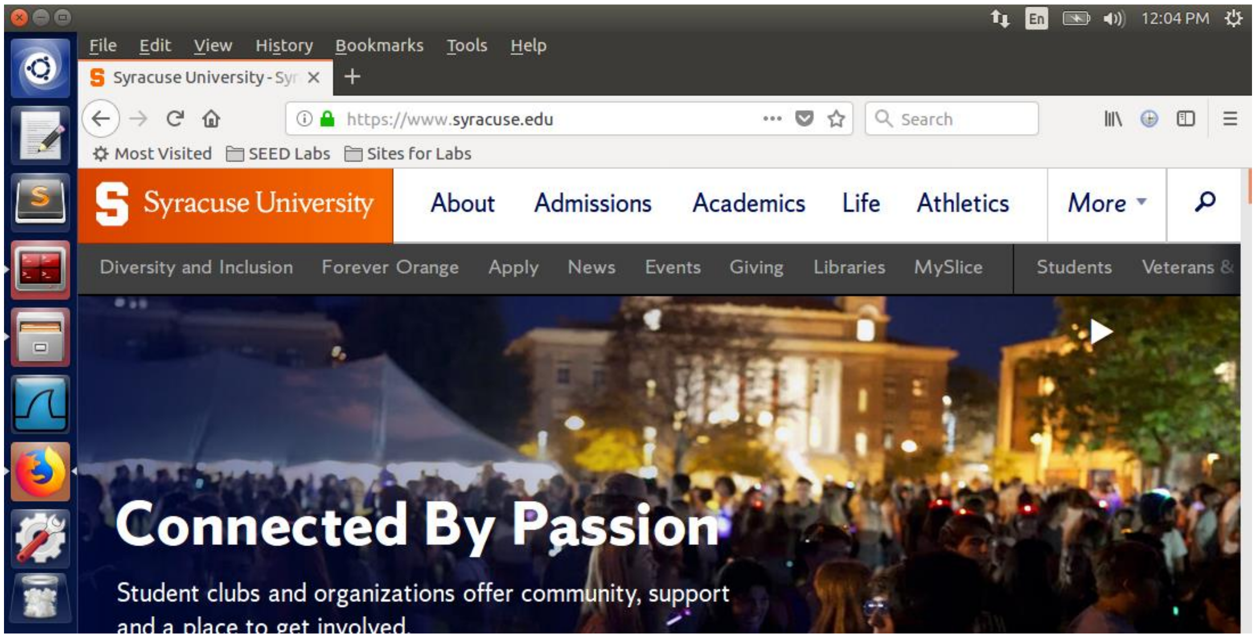
- Check the proxy settings to make sure that they are correct.
- Contact your network administrator to make sure the proxy server is working.

The error message that appeared was “The proxy server is refusing connections.” The Wireshark log below displays the standard DNS packets between machine A and IP 192.168.1.1. Notice that the packets directed to IP 128.230.18.63 (www.syr.edu) have been dropped and are thus not present in the log.


SshClosedSyrEduNotAccessable.pcapng
Apply a display filter … <Ctrl-/>
Expression…

```
No. Time                          Source       Destination  Protocol Length Info
  1 2023-02-04 12:06:35.9840662…  10.0.2.5     192.168.1.1  DNS       74 Standard query …
  2 2023-02-04 12:06:35.9841658…  10.0.2.5     192.168.1.1  DNS       74 Standard query …
  3 2023-02-04 12:06:36.0014767…  192.168.1.1  10.0.2.5     DNS       90 Standard query …
  4 2023-02-04 12:06:36.0053003…  192.168.1.1  10.0.2.5     DNS      102 Standard query …
  5 2023-02-04 12:06:38.3201784…  10.0.2.5     192.168.1.1  DNS       71 Standard query …
  6 2023-02-04 12:06:38.3203300…  10.0.2.5     192.168.1.1  DNS       71 Standard query …
  7 2023-02-04 12:06:38.3649179…  192.168.1.1  10.0.2.5     DNS      101 Standard query …
  8 2023-02-04 12:06:38.3771934…  192.168.1.1  10.0.2.5     DNS      130 Standard query …
  9 2023-02-04 12:06:38.4408079…  10.0.2.5     192.168.1.1  DNS       75 Standard query …
 10 2023-02-04 12:06:38.4409018…  10.0.2.5     192.168.1.1  DNS       75 Standard query …
 11 2023-02-04 12:06:38.4554855…  192.168.1.1  10.0.2.5     DNS      125 Standard query …
 12 2023-02-04 12:06:38.4657709…  192.168.1.1  10.0.2.5     DNS      190 Standard query …
 13 2023-02-04 12:06:39.0240017…  10.0.2.5     192.168.1.1  DNS       84 Standard query …

▸ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_be:19:62 (08:00:27:be:19:62), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
▸ Internet Protocol Version 4, Src: 10.0.2.5, Dst: 192.168.1.1
▸ User Datagram Protocol, Src Port: 11858, Dst Port: 53
▸ Domain Name System (query)

0000  52 54 00 12 35 00 08 00  27 be 19 62 08 00 45 00   RT..5... '..b..E.
0010  00 3c 8b 77 40 00 40 11  e1 8b 0a 00 02 05 c0 a8   .<.w@.@. ........
0020  01 01 2e 52 00 35 00 28  cd e7 e3 f6 01 00 00 01   ...R.5.( ........
0030  00 00 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   .......w ww.googl
```

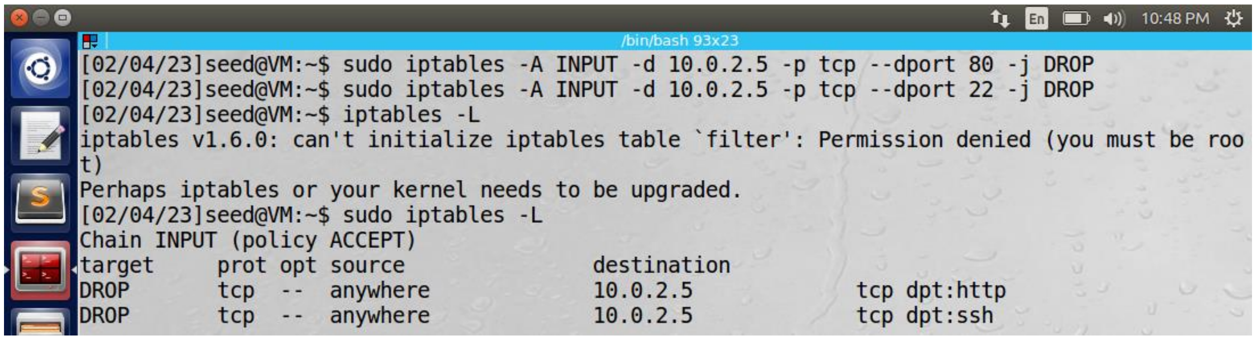
3. The SSH tunnel was reestablished, and Firefox could successfully access www.syr.edu again.



4. The Wireshark logs illustrate the packet traffic with and without the SSH tunnel. The SSH tunnel is established between machine A and machine B through port 9000. The tunnel sends encrypted traffic to machine B, which does not have an egress filter. In this way, the SSH allows the bypassing of the egress filter in machine A's firewall.

Task 4:

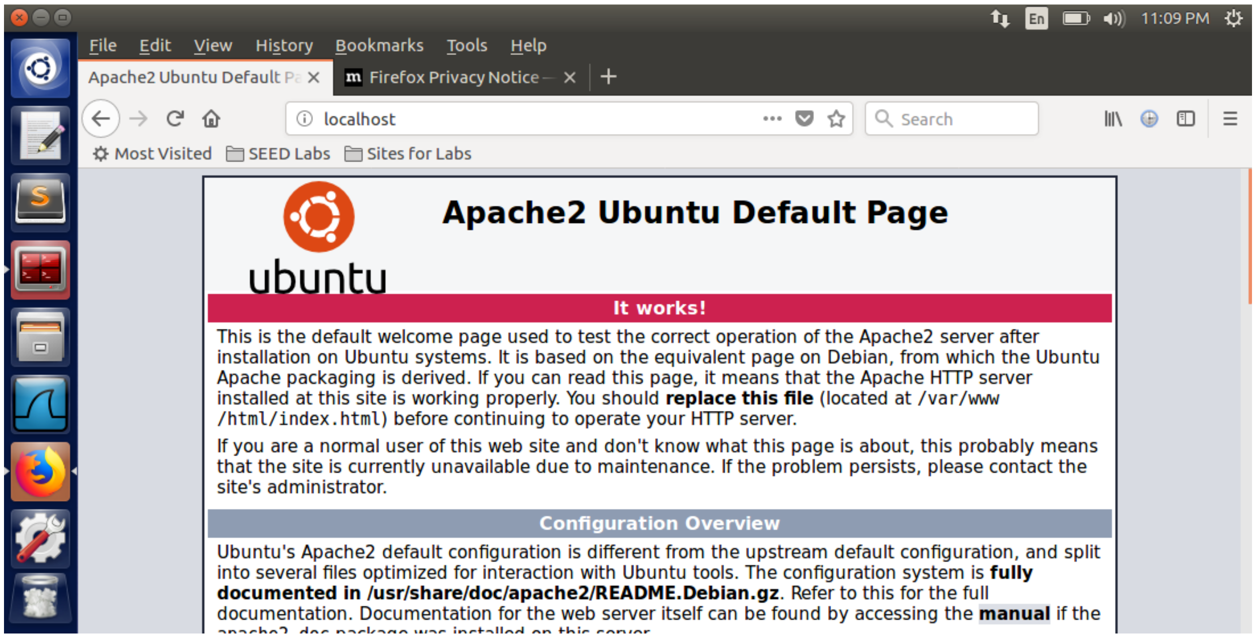
The following firewall rules were used to establish the rules specified in the lab requirements on machine A:



```
[02/04/23]seed@VM:~$ sudo iptables -A INPUT -d 10.0.2.5 -p tcp --dport 80 -j DROP
[02/04/23]seed@VM:~$ sudo iptables -A INPUT -d 10.0.2.5 -p tcp --dport 22 -j DROP
[02/04/23]seed@VM:~$ iptables -L
iptables v1.6.0: can't initialize iptables table `filter': Permission denied (you must be roo
t)
Perhaps iptables or your kernel needs to be upgraded.
[02/04/23]seed@VM:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             10.0.2.5             tcp dpt:http
DROP       tcp  --  anywhere             10.0.2.5             tcp dpt:ssh
```

The rules will drop any ingress TCP packets using port 80 (webserver) or port 22 (ssh tunnel). Therefore, no external machines will be able to access machine A's internal webserver (port 80) or establish an SSH connection using ingress SSH commands (sent to machine A).

Below is machine A's internal webserver.



Apache2 Ubuntu Default Page

It works!

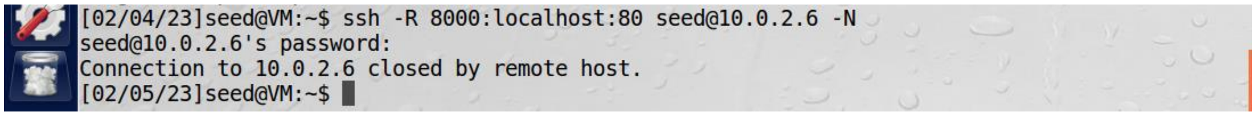
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the

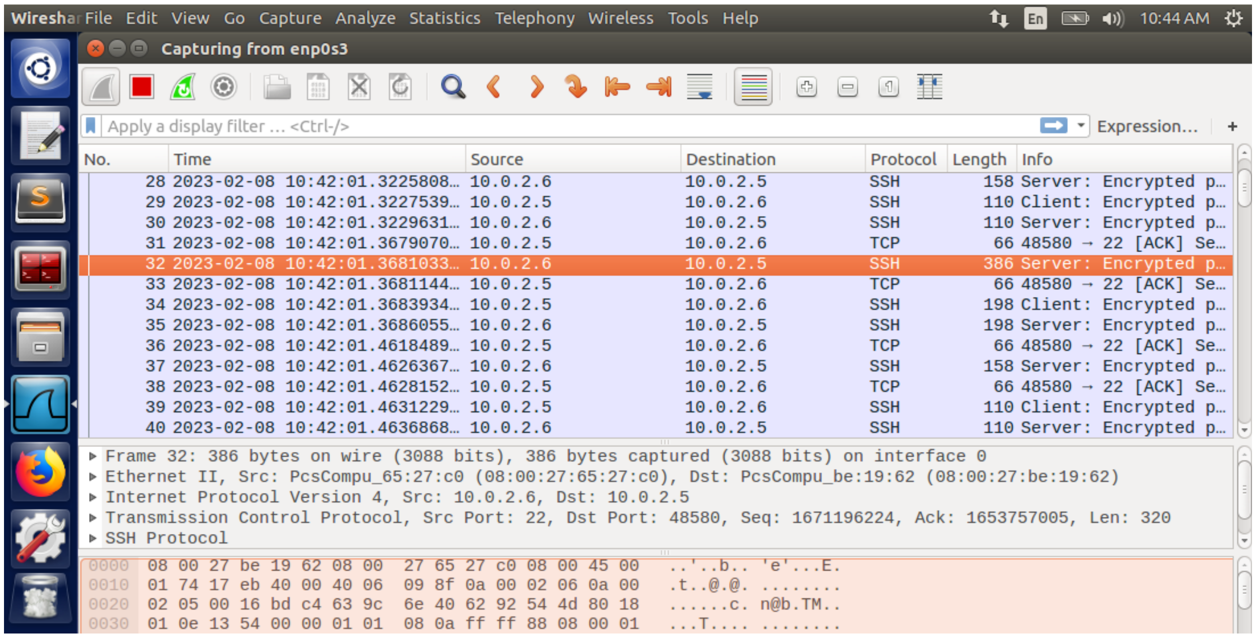
How would machine B access this server on machine A? One method is to use a reverse tunnel established by machine A to machine B. The SSH firewall rule only blocked ingress SSH commands, not egress. The following rule was used to establish the reverse tunnel between machines A and B:



```
[02/04/23]seed@VM:~$ ssh -R 8000:localhost:80 seed@10.0.2.6 -N
seed@10.0.2.6's password:
Connection to 10.0.2.6 closed by remote host.
[02/05/23]seed@VM:~$ 
```

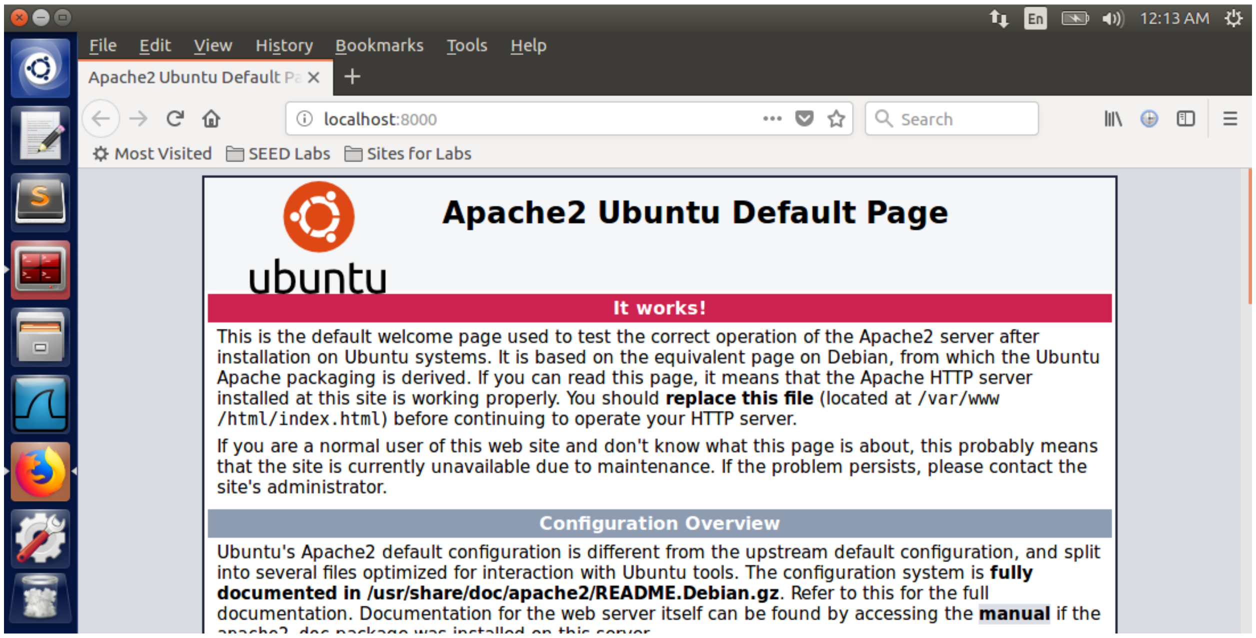
The localhost port 8000 was used to establish the reverse tunnel with seed@10.0.2.6 (machine B). The tunnel forwarded traffic from the localhost server on port 80 of machine A.

Below is a Wireshark log of the traffic taken from enp0s3 on machine A. The SSH traffic is encrypted, so the firewall rule allows the sending of information from the localhost server on port 80 of machine A.


Wireshar File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
10:44 AM
Capturing from enp0s3
Apply a display filter … <Ctrl-/>
Expression…

```
No.  Time                          Source     Destination  Protocol Length Info
  28 2023-02-08 10:42:01.3225808… 10.0.2.6   10.0.2.5     SSH       158 Server: Encrypted p…
  29 2023-02-08 10:42:01.3227539… 10.0.2.5   10.0.2.6     SSH       110 Client: Encrypted p…
  30 2023-02-08 10:42:01.3229631… 10.0.2.6   10.0.2.5     SSH       110 Server: Encrypted p…
  31 2023-02-08 10:42:01.3679070… 10.0.2.5   10.0.2.6     TCP        66 48580 → 22 [ACK] Se…
  32 2023-02-08 10:42:01.3681033… 10.0.2.6   10.0.2.5     SSH       386 Server: Encrypted p…
  33 2023-02-08 10:42:01.3681144… 10.0.2.5   10.0.2.6     TCP        66 48580 → 22 [ACK] Se…
  34 2023-02-08 10:42:01.3683934… 10.0.2.5   10.0.2.6     SSH       198 Client: Encrypted p…
  35 2023-02-08 10:42:01.3686055… 10.0.2.6   10.0.2.5     SSH       198 Server: Encrypted p…
  36 2023-02-08 10:42:01.4618489… 10.0.2.5   10.0.2.6     TCP        66 48580 → 22 [ACK] Se…
  37 2023-02-08 10:42:01.4626367… 10.0.2.6   10.0.2.5     SSH       158 Server: Encrypted p…
  38 2023-02-08 10:42:01.4628152… 10.0.2.5   10.0.2.6     TCP        66 48580 → 22 [ACK] Se…
  39 2023-02-08 10:42:01.4631229… 10.0.2.5   10.0.2.6     SSH       110 Client: Encrypted p…
  40 2023-02-08 10:42:01.4636868… 10.0.2.6   10.0.2.5     SSH       110 Server: Encrypted p…

▸ Frame 32: 386 bytes on wire (3088 bits), 386 bytes captured (3088 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_65:27:c0 (08:00:27:65:27:c0), Dst: PcsCompu_be:19:62 (08:00:27:be:19:62)
▸ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.5
▸ Transmission Control Protocol, Src Port: 22, Dst Port: 48580, Seq: 1671196224, Ack: 1653757005, Len: 320
▸ SSH Protocol

0000  08 00 27 be 19 62 08 00  27 65 27 c0 08 00 45 00   ..'..b.. 'e'...E.
0010  01 74 17 eb 40 00 40 06  09 8f 0a 00 02 06 0a 00   .t..@.@. ........
0020  02 05 00 16 bd c4 63 9c  6e 40 62 92 54 4d 80 18   ......c. n@b.TM..
0030  01 0e 13 54 00 00 01 01  08 0a ff ff 88 08 00 01   ...T.... ........
```

Below is a screenshot of machine B accessing machine A's localhost server using port 8000.



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the

Observation: Machine B was able to access machine A's localhost server via a reverse tunnel established on port 8000. The reverse tunnel encrypted and forwarded data from the localhost server on port 80 of machine A.

Explanation: By encrypting the packets using an SSH tunnel with machine A, machine B was able to successfully access machine A's localhost server. Therefore, a reverse SSH tunnel was used in place of a VPN.

Appendix:

Task 2: task2.c

```
// CSE-644: Internet Security
// Dr. Syed Shazli
// Anthony Redamonti
// 2-7-2023
//
// task2.c

#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>
#include <linux/udp.h>
#include <linux/icmp.h>
#include <linux/if_ether.h>
#include <linux/inet.h>

// 6 hooks are needed for 5 rules because one of the rules is used
// to print out the packet information
static struct nf_hook_ops hook1, hook2, hook3, hook4, hook5, hook6;

// blocking ping to destination IP 10.0.2.7
unsigned int blockIcmpTo7(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct icmphdr *icmph;
   char destIp[16] = "10.0.2.7";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);

   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(destIp, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_ICMP) {
       icmph = icmp_hdr(skb);
       if (iph->daddr == ip_addr && icmph->type == ICMP_ECHO){
            printk(KERN_WARNING "*** Dropping %pI4 (ICMP) \n", &(iph->daddr));
            return NF_DROP;
        }
    }
    return NF_ACCEPT;
}

// blocking ping from destination IP 10.0.2.7
unsigned int blockIcmpFrom7(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct icmphdr *icmph;
   char srcIp[16] = "10.0.2.7";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);

   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(srcIp, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_ICMP) {
       icmph = icmp_hdr(skb);
       if (iph->saddr == ip_addr && icmph->type == ICMP_ECHO){
            printk(KERN_WARNING "*** Dropping %pI4 (ICMP) \n", &(iph->daddr));
            return NF_DROP;
        }
    }
    return NF_ACCEPT;
}

// blocking telnet to 10.0.2.7 : 23
unsigned int blockTelnetTo7(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct tcphdr *tcph;

   u16  port   = 23;
   char ip[16] = "10.0.2.7";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;
   iph = ip_hdr(skb);

   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_TCP) {
       tcph = tcp_hdr(skb);
       if (iph->daddr == ip_addr && ntohs(tcph->dest) == port){
            printk(KERN_WARNING "*** Dropping %pI4 (TCP), port %d\n", &(iph-
>daddr), port);
            return NF_DROP;
        }
   }
   return NF_ACCEPT;
}

// blocking telnet from 10.0.2.7 : 23
unsigned int blockTelnetFrom7(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct tcphdr *tcph;

   u16  port   = 23;
   char ip[16] = "10.0.2.7";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);

   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_TCP) {
       tcph = tcp_hdr(skb);
       if (iph->saddr == ip_addr && ntohs(tcph->dest) == port){
            printk(KERN_WARNING "*** Dropping %pI4 (TCP), port %d\n", &(iph-
>daddr), port);
            return NF_DROP;
        }
   }
   return NF_ACCEPT;
}
```

```
// blocking telnet from 10.0.2.6 : 23
unsigned int blockTelnetFrom6(void *priv, struct sk_buff *skb,
                       const struct nf_hook_state *state)
{
   struct iphdr *iph;
   struct tcphdr *tcph;

   u16  port   = 23;
   char ip[16] = "10.0.2.6";
   u32  ip_addr;

   if (!skb) return NF_ACCEPT;

   iph = ip_hdr(skb);

   // Convert the IPv4 address from dotted decimal to 32-bit binary
   in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

   if (iph->protocol == IPPROTO_TCP) {
       tcph = tcp_hdr(skb);
       if (iph->saddr == ip_addr && ntohs(tcph->dest) == port){
            printk(KERN_WARNING "*** Dropping %pI4 (TCP), port %d\n", &(iph-
>daddr), port);
            return NF_DROP;
        }
   }
   return NF_ACCEPT;
}

// print the packet information to the log file.
unsigned int printInfo(void *priv, struct sk_buff *skb,
                 const struct nf_hook_state *state)
{
   struct iphdr *iph;
   char *hook;
   char *protocol;

   switch (state->hook){
     case NF_INET_LOCAL_IN:     hook = "LOCAL_IN";     break;
     case NF_INET_LOCAL_OUT:    hook = "LOCAL_OUT";    break;
     case NF_INET_PRE_ROUTING:  hook = "PRE_ROUTING";  break;
     case NF_INET_POST_ROUTING: hook = "POST_ROUTING"; break;
     case NF_INET_FORWARD:      hook = "FORWARD";      break;
     default:                   hook = "IMPOSSIBLE";   break;
   }
   printk(KERN_INFO "*** %s\n", hook);

   iph = ip_hdr(skb);
   switch (iph->protocol){
     case IPPROTO_UDP:  protocol = "UDP";   break;
     case IPPROTO_TCP:  protocol = "TCP";   break;
     case IPPROTO_ICMP: protocol = "ICMP";  break;
     default:           protocol = "OTHER"; break;
   }

   // Print out the IP addresses and protocol
   printk(KERN_INFO "    %pI4  --> %pI4 (%s)\n",
                    &(iph->saddr), &(iph->daddr), protocol);

   return NF_ACCEPT;
}

int registerFilter(void) {
   printk(KERN_INFO "Registering filters.\n");

   // hook 1 = print packet information
   hook1.hook = printInfo;
   hook1.hooknum = NF_INET_LOCAL_OUT;
   hook1.pf = PF_INET;
   hook1.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook1);

   // hook 2 = block ICMP to 10.0.2.7
   hook2.hook = blockIcmpTo7;
   hook2.hooknum = NF_INET_LOCAL_OUT;
   hook2.pf = PF_INET;
   hook2.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook2);

   // hook 3 = block ICMP from 10.0.2.7
   hook3.hook = blockIcmpFrom7;
   hook3.hooknum = NF_INET_LOCAL_IN;
   hook3.pf = PF_INET;
   hook3.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook3);

   // hook 4 = block telnet to 10.0.2.7
   hook4.hook = blockTelnetTo7;
   hook4.hooknum = NF_INET_LOCAL_OUT;
   hook4.pf = PF_INET;
   hook4.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook4);

   // hook 5 = block telnet from 10.0.2.7
   hook5.hook = blockTelnetFrom7;
   hook5.hooknum = NF_INET_LOCAL_IN;
   hook5.pf = PF_INET;
   hook5.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook5);

   // hook 6 = block telnet from 10.0.2.6
   hook6.hook = blockTelnetFrom6;
   hook6.hooknum = NF_INET_LOCAL_IN;
   hook6.pf = PF_INET;
   hook6.priority = NF_IP_PRI_FIRST;
   nf_register_net_hook(&init_net, &hook6);

   return 0;
}

// remove the hooks from the module
void removeFilter(void) {
   printk(KERN_INFO "The filters are being removed.\n");
   nf_unregister_net_hook(&init_net, &hook1);
   nf_unregister_net_hook(&init_net, &hook2);
   nf_unregister_net_hook(&init_net, &hook3);
   nf_unregister_net_hook(&init_net, &hook4);
   nf_unregister_net_hook(&init_net, &hook5);
   nf_unregister_net_hook(&init_net, &hook6);
}

module_init(registerFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

Task 2: Makefile

```
obj-m += seedBlock.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

ins:
    sudo dmesg -C
    sudo insmod seedFilter.ko

rm:
    sudo rmmod seedFilter
```