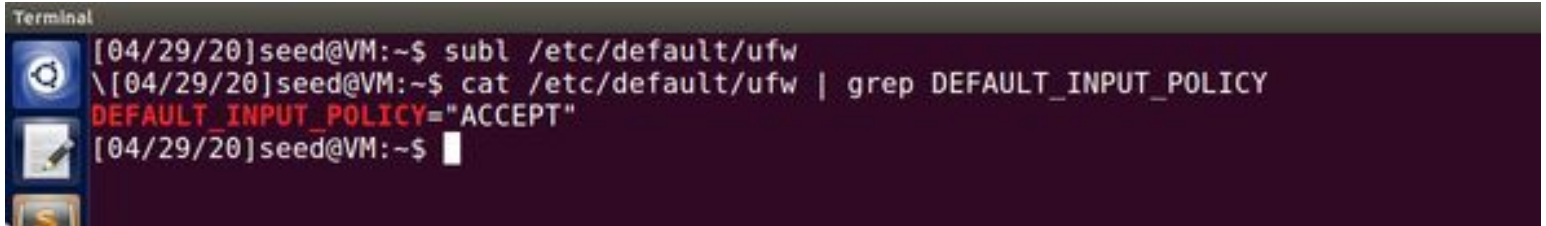


Task 1 : Using Firewall:

In this task we'll make use of linux's inbuilt tool iptables which also uses a frontend tool called ufw to setup some firewall policies and observe our system's behaviour when the policies come into effect. Before performing the tasks, we first enable the firewall to accept the incoming traffic. Here is what we do for that, I changed the default value of **DEFAULT_INPUT_POLICY** from drop to accept.



```

[04/29/20]seed@VM:~$ subl /etc/default/ufw
\[04/29/20]seed@VM:~$ cat /etc/default/ufw | grep DEFAULT_INPUT_POLICY
DEFAULT_INPUT_POLICY="ACCEPT"
[04/29/20]seed@VM:~$

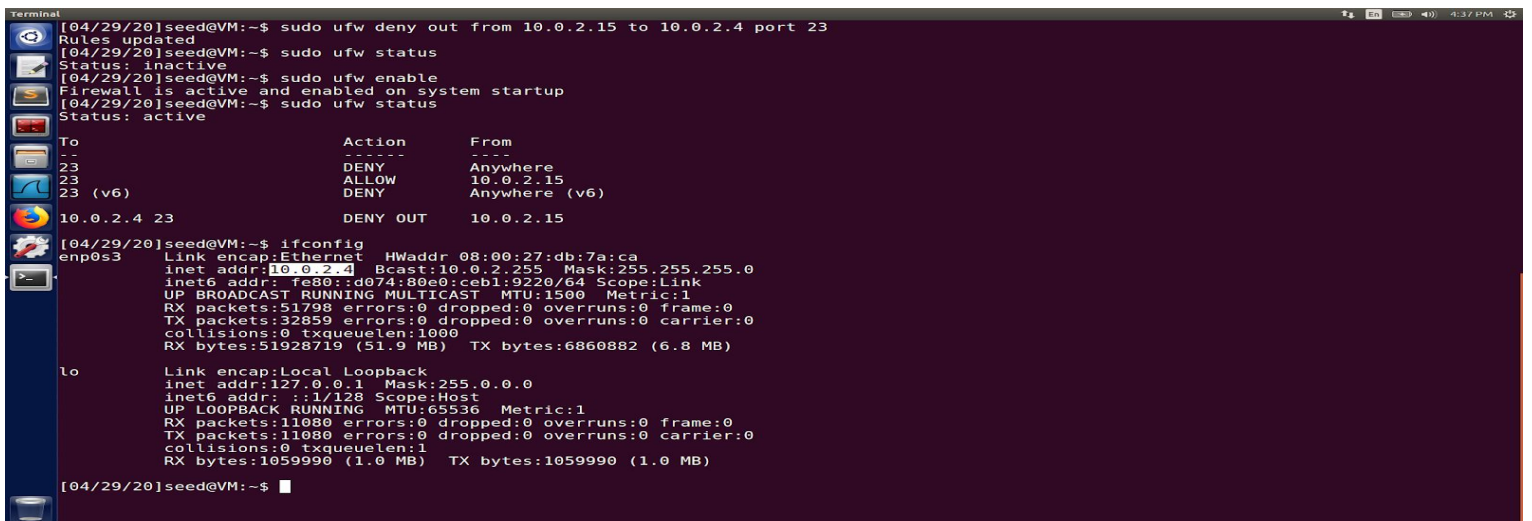
```

We know by default the telnet runs on port 23. Making use of ufw's deny with the port number 23 as an argument, we can stop other machines accessing the host through telnet as the port 23 is denied access. Below is the screenshot of that being performed.

Prevent A from doing telnet to Machine B:

The two machines **B** and **A** I'm using for this attack have IP's **10.0.2.4** and **10.0.2.15** respectively. In the below screenshot we can see I have disabled telnet running on port 23 of Machine A using ufw's deny to port 23.

We make use of the command **sudo ufw deny out from 10.0.2.15 to 10.0.2.4 port 23** Doing this denies access to Machine **B**'s Telnet from Machine **A**. And once we execute the command the rules get updated and to enable the updated rules, we run **sudo ufw enable**.



```

[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.15 to 10.0.2.4 port 23
Rules updated
[04/29/20]seed@VM:~$ sudo ufw status
Status: inactive
[04/29/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[04/29/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
23 DENY Anywhere
23 ALLOW 10.0.2.15
23 (v6) DENY Anywhere (v6)
10.0.2.4 23 DENY OUT 10.0.2.15

[04/29/20]seed@VM:~$ ifconfig
enp0s3 Link encap:Ethernet HWaddr 08:00:27:db:7a:ca
inet addr:10.0.2.4 Bcast:10.0.2.255 Mask:255.255.255.0
inet6 addr: fe80::d074:80e0:ceb1:9220/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:51798 errors:0 dropped:0 overruns:0 frame:0
TX packets:32859 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:51928719 (51.9 MB) TX bytes:6860882 (6.8 MB)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:11080 errors:0 dropped:0 overruns:0 frame:0
TX packets:11080 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:1059990 (1.0 MB) TX bytes:1059990 (1.0 MB)

[04/29/20]seed@VM:~$

```

Once the above commands are run, the firewall is now enabled.

In the below screenshot we can see how Machine **A** trying to access machine **B** is failing.

```

[04/29/20]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:53:14:6b
            inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
            inet6 addr: fe80::927:149:2255:cf70/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:56 errors:0 dropped:0 overruns:0 frame:0
            TX packets:61 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:7734 (7.7 KB)  TX bytes:6973 (6.9 KB)

lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:69 errors:0 dropped:0 overruns:0 frame:0
            TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:21500 (21.5 KB)  TX bytes:21500 (21.5 KB)

[04/29/20]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
  
```

Prevent B from doing telnet to Machine A:

The two machines **B** and **A** I'm using for this attack have IP's **10.0.2.4** and **10.0.2.15** respectively. In the below screenshot we can see I have disabled telnet running on port 23 of Machine A using ufw's deny to port 23.

We make use of the command **sudo ufw deny out from 10.0.2.4 to 10.0.2.15 port 23** Doing this denies access to Machine A's Telnet from Machine B. And once we execute the command the rules get updated and to enable the updated rules, we run **sudo ufw enable**.

```

[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.4 to 10.0.2.15 port 23
Rules updated
[04/29/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[04/29/20]seed@VM:~$ sudo ufw status
Status: active

To          Action      From
--          -
10.0.2.15 23      DENY OUT    10.0.2.4

[04/29/20]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:53:14:6b
            inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
            inet6 addr: fe80::927:149:2255:cf70/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:35939 errors:0 dropped:0 overruns:0 frame:0
            TX packets:22151 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:39915225 (39.9 MB)  TX bytes:4348510 (4.3 MB)

lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:3248 errors:0 dropped:0 overruns:0 frame:0
            TX packets:3248 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:330466 (330.4 KB)  TX bytes:330466 (330.4 KB)

[04/29/20]seed@VM:~$
  
```

Once the above commands are run, the firewall is now enabled.

In the below screenshot we can see how Machine **B** trying to access machine **A** is failing.

```

[04/29/20]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:db:7a:ca
            inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
            inet6 addr: fe80::d074:80e0:ceb1:9220/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:17951 errors:0 dropped:0 overruns:0 frame:0
            TX packets:9819 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:20430432 (20.4 MB)  TX bytes:2589849 (2.5 MB)

lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:1662 errors:0 dropped:0 overruns:0 frame:0
            TX packets:1662 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:152316 (152.3 KB)  TX bytes:152316 (152.3 KB)

[04/29/20]seed@VM:~$ telnet 10.0.2.15
Trying 10.0.2.15...

```

Prevent A from visiting an external web site. You can choose any website that you like to block, but keep in mind, some web servers have multiple IP addresses:

In this task, I'll try to ping iit.edu and we can see it's IP being **174.143.130.167**

Therefore to disable access to instagram.com from Machine **A**, we update the firewall using the below command. **sudo ufw deny out from 10.0.2.15 to 174.143.130.167**

And once the above command is executed, it updates all the rules and to enable it, we run **sudo ufw enable**. And now, once the firewall rules are enabled, when we try to ping **iit.edu**, we can see the ping being failed making out task successful.

```

[04/29/20]seed@VM:~$ clear

[04/29/20]seed@VM:~$ ping iit.edu
PING iit.edu (174.143.130.167) 56(84) bytes of data:
64 bytes from www-c.iit.edu (174.143.130.167): icmp_seq=1 ttl=51 time=47.4 ms
64 bytes from www-c.iit.edu (174.143.130.167): icmp_seq=2 ttl=51 time=39.6 ms
64 bytes from www-c.iit.edu (174.143.130.167): icmp_seq=3 ttl=51 time=48.7 ms
64 bytes from www-c.iit.edu (174.143.130.167): icmp_seq=4 ttl=51 time=44.4 ms
64 bytes from www-c.iit.edu (174.143.130.167): icmp_seq=5 ttl=51 time=40.8 ms
64 bytes from www-c.iit.edu (174.143.130.167): icmp_seq=6 ttl=51 time=42.0 ms
64 bytes from www-c.iit.edu (174.143.130.167): icmp_seq=7 ttl=51 time=40.2 ms
^C
--- iit.edu ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6014ms
rtt min/avg/max/mdev = 39.614/43.335/48.714/3.355 ms
[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.15 to 174.143.130.167
Rule added
[04/29/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[04/29/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
10.0.2.15 23 DENY OUT 10.0.2.4
34.225.190.8 DENY OUT 10.0.2.15
3.224.36.18 DENY OUT 10.0.2.15
216.239.32.21 DENY OUT 10.0.2.15
174.143.130.167 DENY OUT 10.0.2.15

[04/29/20]seed@VM:~$ ping iit.edu
PING iit.edu (174.143.130.167) 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

```


The reason for using iit.edu being it's IP did not change with each ping command. I tried using instagram.com... but each time it's IP kept changing making it hard to update the firewall each time.

Task 2: Implementing a Simple Firewall

In this task I extended the code given in the textbook code from [here](#) to perform various tasks defined in the handout.

Task 2.1: Prevent A from doing telnet to Machine B:

For this task, I have chosen Machine A with an IP 10.0.2.4 and Machine B that has an IP 10.0.2.5. The code that I'm using to prevent Machine A from doing telnet to Machine B is shown below.

```
telnetFilter.c  x  Makefile  x
1 #include <linux/kernel.h>
2 #include <linux/module.h>
3 #include <linux/netfilter.h>
4 #include <linux/netfilter_ipv4.h>
5 #include <linux/ip.h>
6 #include <linux/tcp.h>
7 #include <linux/inet.h>
8
9 static struct nf_hook_ops telnetFilterHook;
10
11 unsigned int telnetFilter(void *priv, struct sk_buff *skb,
12                          const struct nf_hook_state *state)
13 {
14     struct iphdr *iph;
15     struct tcphdr *tcph;
16
17     iph = ip_hdr(skb);
18     tcph = (void *)iph+iph->ihl*4;
19
20
21     if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23) && iph->saddr == in_aton("10.0.2.4") && iph->daddr == in_aton("10.0.2.5")) {
22         printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",
23                ((unsigned char *)iph->daddr)[0],
24                ((unsigned char *)iph->daddr)[1],
25                ((unsigned char *)iph->daddr)[2],
26                ((unsigned char *)iph->daddr)[3]);
27         return NF_DROP;
28     } else {
29         return NF_ACCEPT;
30     }
31 }
32
33
34 int setupFilter(void) {
35     printk(KERN_INFO "Registering a Telnet filter.\n");
36     telnetFilterHook.hook = telnetFilter;
37     telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
38     telnetFilterHook.pf = PF_INET;
39     telnetFilterHook.priority = NF_IP_PRI_FIRST;
40     nf_register_hook(&telnetFilterHook);
41     return 0;
42 }
43
44 void removeFilter(void) {
45     printk(KERN_INFO "Telnet filter is being removed.\n");
46     nf_unregister_hook(&telnetFilterHook);
47 }
48
49 module_init(setupFilter);
50 module_exit(removeFilter);
```

The content of the Makefile is given below

obj-m += telnetFilter.o

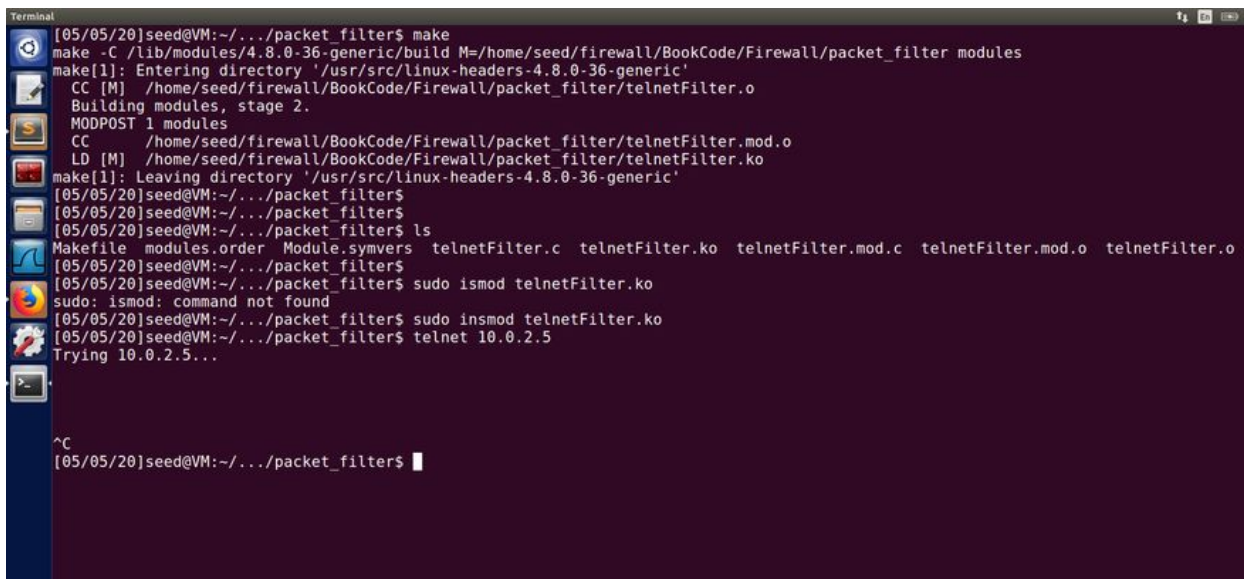
all:

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

clean:

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

In the screenshot below, we can see I compiled the code using make command and then I inserted the kernel module using the insmod command. Once the kernel module is inserted, I try to execute the telnet to Machine B with an IP 10.0.2.5. We can see, it fails to execute the telnet command making the task successful. That is because a tiny firewall is being implemented on executing the code that required us to load our code into the kernel using the hooks provided by the netfilter.



```
Terminal
[05/05/20]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/firewall/BookCode/Firewall/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
CC [M] /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.mod.o
LD [M] /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[05/05/20]seed@VM:~/.../packet_filter$
[05/05/20]seed@VM:~/.../packet_filter$
[05/05/20]seed@VM:~/.../packet_filter$ ls
Makefile modules.order Module.symvers telnetFilter.c telnetFilter.ko telnetFilter.mod.c telnetFilter.mod.o telnetFilter.o
[05/05/20]seed@VM:~/.../packet_filter$
[05/05/20]seed@VM:~/.../packet_filter$ sudo insmod telnetFilter.ko
sudo: insmod: command not found
[05/05/20]seed@VM:~/.../packet_filter$ sudo insmod telnetFilter.ko
[05/05/20]seed@VM:~/.../packet_filter$ telnet 10.0.2.5
Trying 10.0.2.5...
^C
[05/05/20]seed@VM:~/.../packet_filter$
```

Task 2.2: Prevent B from doing telnet to Machine A:

This task is just the opposite of **task 2.1**, I have chosen Machine A with an IP 10.0.2.4 and Machine B that has an IP 10.0.2.5. The code that I'm using to prevent Machine B from doing telnet to Machine A is shown below. Here the code too is very similar to the code of the

previous one except for that the if condition following from line 21 in the code screenshot of the previous task is changed to

```

if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23) && iph->saddr ==
in_aton("10.0.2.5") && iph->daddr == in_aton("10.0.2.4")) {
    printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",
        ((unsigned char *)&iph->daddr)[0],
        .....
        ((unsigned char *)&iph->daddr)[3]);
    return NF_DROP;
} else {
    return NF_ACCEPT;
}

```

This change employs the telnet is being prevented from Machine B with IP 10.0.2.5 to Machine A with IP 10.0.2.4. The Makefile remains the same. In the screenshot below, we can see from the highlighted section that the program is running from Machine B with an IP 10.0.2.15. I then compiled the code using make command and then I inserted the kernel module using the insmod command. Once the kernel module is inserted, I try to execute the telnet to Machine A with an IP 10.0.2.4. We can see, it fails to execute the telnet command making the task successful. That is because a tiny firewall is being implemented on executing the code that required us to load our code into the kernel using the hooks provided by the netfilter.

```

terminal
enp0s3  Link encap:Ethernet  HWaddr 08:00:27:be:04:6a
       inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
       inet6 addr: fe80::c794:8eeb:cac2:b9d4/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:10722 errors:0 dropped:0 overruns:0 frame:0
       TX packets:7900 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:11099099 (11.0 MB)  TX bytes:1474986 (1.4 MB)

lo      Link encap:Local Loopback
       inet addr:127.0.0.1  Mask:255.0.0.0
       inet6 addr: ::1/128 Scope:Host
       UP LOOPBACK RUNNING  MTU:65536  Metric:1
       RX packets:1218 errors:0 dropped:0 overruns:0 frame:0
       TX packets:1218 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1
       RX bytes:140841 (140.8 KB)  TX bytes:140841 (140.8 KB)

[05/05/20]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/firewall/BookCode/Firewall/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.mod.o
  LD [M]  /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[05/05/20]seed@VM:~/.../packet_filter$ ls
Makefile  modules.order  Module.symvers  telnetFilter.c  telnetFilter.ko  telnetFilter.mod.c  telnetFilter.mod.o  telnetFilter.o
[05/05/20]seed@VM:~/.../packet_filter$ sudo insmod telnetFilter.ko
[05/05/20]seed@VM:~/.../packet_filter$ telnet 10.0.2.4
Trying 10.0.2.4...
^C
[05/05/20]seed@VM:~/.../packet_filter$

```

Task 2.3: Prevent A from visiting an external web site. You can choose any website that you like to block, but keep in mind, some web servers have multiple IP addresses:

This task is very similar to **task 2.2**, I have chosen Machine A with an IP 10.0.2.4 and an external site as www.iit.edu with an IP 50.19.226.237. The code that I'm using to prevent Machine A from Accessing www.iit.edu is shown below. Here the code too is very similar to the code of the previous one except for that the if condition following from line 21 in the code screenshot of the previous task is changed to

```
if (iph->saddr == in_aton("10.0.2.4") && iph->daddr == in_aton("50.19.226.237")) {  
    printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",  
        ((unsigned char *)&iph->daddr)[0],  
        .....  
        ((unsigned char *)&iph->daddr)[3]);  
    return NF_DROP;  
} else {  
    return NF_ACCEPT;  
}
```

This change employs the access being denied to www.iit.edu from Machine A with IP 10.0.2.4. The Makefile remains the same.

In the screenshot below, we can see from the highlighted section that the IP of www.iit.edu is 50.19.226.237. I then compiled the code using make command and then I inserted the kernel module using the insmod command. Once the kernel module is inserted, I try to ping www.iit.edu. We can see, it fails to execute the ping command making the task successful. That is because a tiny firewall is being implemented on executing the code that required us to load our code into the kernel using the hooks provided by the netfilter.

```

[05/05/20]seed@VM:~/.../packet_filter$ ping www.iit.edu
PING www.iit.edu (50.19.226.237) 56(84) bytes of data.
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=1 ttl=49 time=39.6 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=2 ttl=49 time=32.9 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=3 ttl=49 time=33.4 ms
64 bytes from ec2-50-19-226-237.compute-1.amazonaws.com (50.19.226.237): icmp_seq=4 ttl=49 time=33.8 ms
^C
--- www.iit.edu ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3019ms
rtt min/avg/max/mdev = 32.919/34.959/39.676/2.751 ms
[05/05/20]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/firewall/BookCode/Firewall/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
CC [M] /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.mod.o
LD [M] /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[05/05/20]seed@VM:~/.../packet_filter$ sudo insmod telnetFilter.ko
[05/05/20]seed@VM:~/.../packet_filter$ ping www.iit.edu
PING www.iit.edu (50.19.226.237) 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
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- www.iit.edu ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6136ms

[05/05/20]seed@VM:~/.../packet_filters$

```

Task 2.4: Prevent A from sending ICMP to any machine

This task is very similar to **task 2.3**, I have chosen Machine A with an IP 10.0.2.4 and Machine B with an IP 10.0.2.5. The code that I'm using to prevent Machine A from sending ICMP packets to Machine B is shown below. Here the code too is very similar to the code of the previous one except for that the if condition following from line 21 in the code screenshot of the previous task is changed to

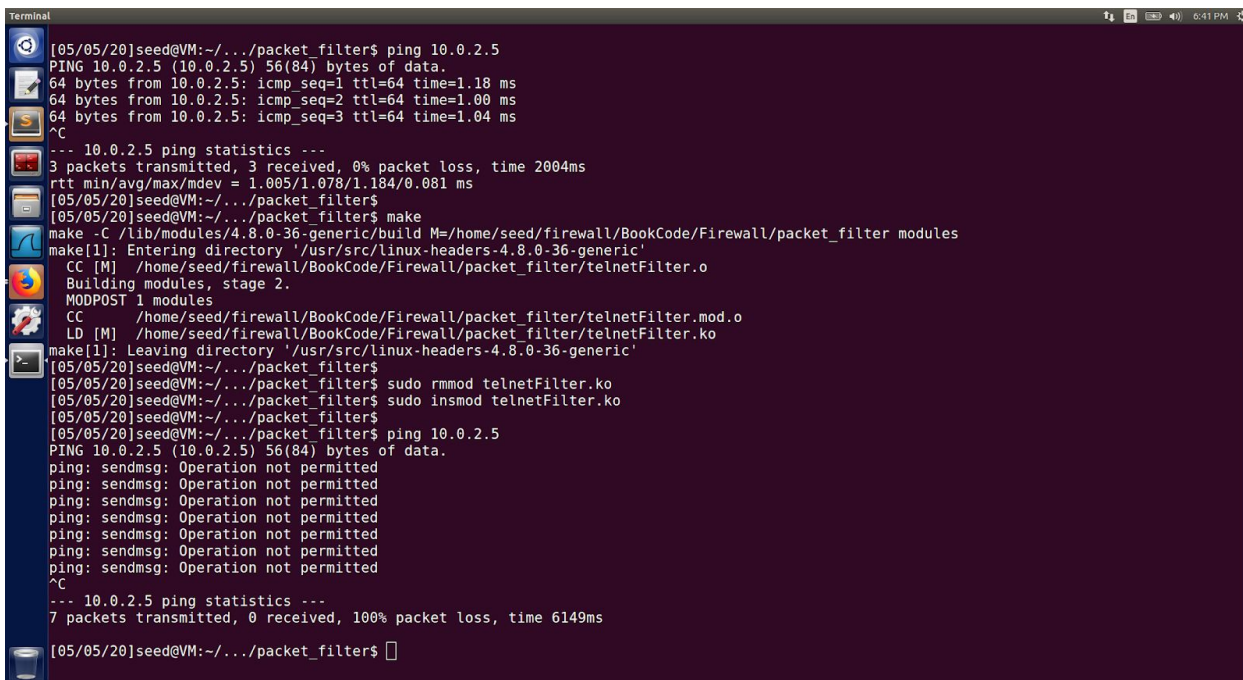
```

if (iph->protocol == IPPROTO_ICMP && iph->saddr == in_aton("10.0.2.4")) {
    printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",
        ((unsigned char *)&iph->daddr)[0],
        ((unsigned char *)&iph->daddr)[1],
        ((unsigned char *)&iph->daddr)[2],
        ((unsigned char *)&iph->daddr)[3]);
    return NF_DROP;
} else {
    return NF_ACCEPT;
}

```


This change employs that ping to any machine from Machine A with IP 10.0.2.4 is denied. The Makefile remains the same.

In the screenshot below, we can see the ping to machine B with IP 10.0.0.5 is successful in the beginning. I then compiled the code using make command and then I inserted the kernel module using the insmod command. Once the kernel module is inserted, I try to ping the same Machine B. We can see, it fails to execute the ping command making the task successful. That is because a tiny firewall is being implemented on executing the code that required us to load our code into the kernel using the hooks provided by the netfilter.



```
[05/05/20]seed@VM:~/.../packet_filter$ 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=1.18 ms
64 bytes from 10.0.2.5: icmp_seq=2 ttl=64 time=1.00 ms
64 bytes from 10.0.2.5: icmp_seq=3 ttl=64 time=1.04 ms
^C
--- 10.0.2.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 1.005/1.078/1.184/0.081 ms
[05/05/20]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/firewall/BookCode/Firewall/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
CC [M] /home/seed/firewall/BookCode/Firewall/packet_filter/telneterFilter.o
Building modules, stage 2.
MODPOST 1 modules
CC      /home/seed/firewall/BookCode/Firewall/packet_filter/telneterFilter.mod.o
LD [M] /home/seed/firewall/BookCode/Firewall/packet_filter/telneterFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[05/05/20]seed@VM:~/.../packet_filter$
[05/05/20]seed@VM:~/.../packet_filter$ sudo rmmod telneterFilter.ko
[05/05/20]seed@VM:~/.../packet_filter$ sudo insmod telneterFilter.ko
[05/05/20]seed@VM:~/.../packet_filter$
[05/05/20]seed@VM:~/.../packet_filter$ ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 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
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 10.0.2.5 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6149ms
[05/05/20]seed@VM:~/.../packet_filter$
```

Task 2.5: Prevent B from sending ICMP to any machine

This task is very similar to **task 2.4**, I have chosen Machine A with an IP 10.0.2.4 and Machine B with an IP 10.0.2.5. The code that I'm using to prevent Machine B from sending ICMP packets to Machine A is shown below. Here the code too is very similar to the code of the previous one except for that the if condition following from line 21 in the code screenshot of the previous task is changed to

```
if (iph->protocol == IPPROTO_ICMP && iph->saddr == in_aton("10.0.2.5")) {
```

```

printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",
((unsigned char *)&iph->daddr)[0],
((unsigned char *)&iph->daddr)[1],
((unsigned char *)&iph->daddr)[2],
((unsigned char *)&iph->daddr)[3]);
return NF_DROP;
} else {
    return NF_ACCEPT;
}

```

This change employs that ping to any machine from Machine **A** with IP 10.0.2.5 is denied. The Makefile remains the same. In the screenshot below, we can see the ping to machine A with IP 10.0.0.5 is successful in the beginning. I then compiled the code using make command and then I inserted the kernel module using the insmod command. Once the kernel module is inserted, I try to ping the same Machine A. We can see, it fails to execute the ping command making the task successful. That is because a tiny firewall is being implemented on executing the code that required us to load our code into the kernel using the hooks provided by the netfilter.

```

[05/05/20]seed@VM:~/.../packet_filter$ ping 10.0.0.5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=248 time=42.4 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=248 time=35.2 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=248 time=35.6 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=248 time=36.2 ms
^C
--- 10.0.0.5 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4015ms
rtt min/avg/max/mdev = 35.265/37.420/42.488/2.950 ms
[05/05/20]seed@VM:~/.../packet_filter$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/firewall/BookCode/Firewall/packet_filter modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
CC [M] /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.o
Building modules, stage 2.
MODPOST 1 modules
CC      /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.mod.o
LD [M]  /home/seed/firewall/BookCode/Firewall/packet_filter/telnetFilter.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[05/05/20]seed@VM:~/.../packet_filter$
[05/05/20]seed@VM:~/.../packet_filter$ sudo insmod telnetFilter.ko
[05/05/20]seed@VM:~/.../packet_filter$
[05/05/20]seed@VM:~/.../packet_filter$ ping 10.0.0.5
PING 10.0.0.5 (10.0.0.5) 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
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 10.0.0.5 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7157ms
[05/05/20]seed@VM:~/.../packet_filter$

```

With this I have specified 5 different rules including the ones we've performed in task 1.

Task 3 : Evading egress Filtering

In this task we make use of tunnels to bypass egress filtering implemented. Firstly even before we proceed further with the tasks below, we setup the firewalls to make sure we're ready to perform the tasks listed below. Here I'm using www.iit.edu instead of www.facebook.com as facebook has multiple static IP's which makes it difficult to perform the following attacks.

I'll be performing the tasks from Machine A which has an IP of 10.0.2.4. And here are the commands to set the firewall rules.

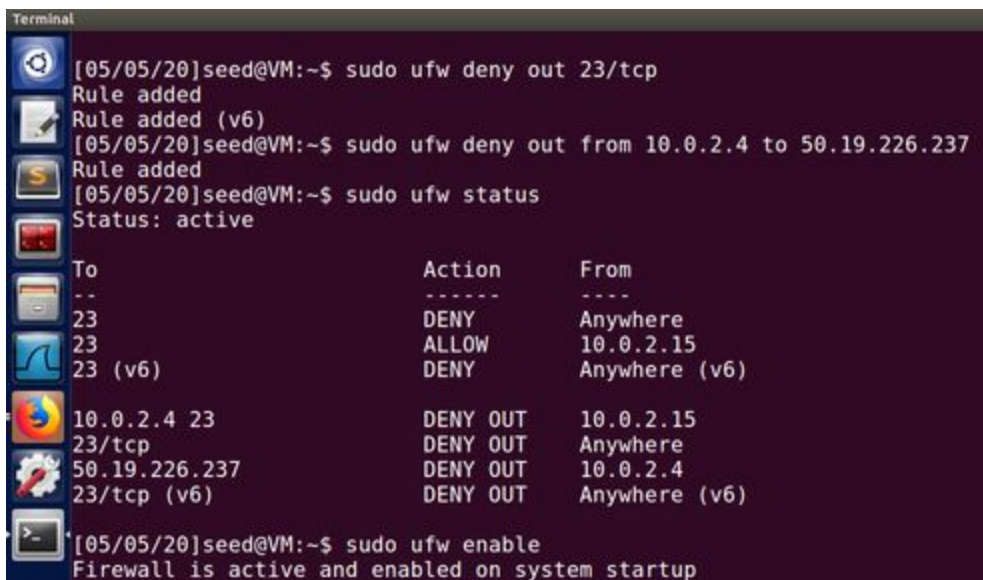
To block any telnet connections, we execute

sudo ufw deny out 23/tcp

To block access to www.iit.edu running on IP 50.19.226.237, we execute

sudo ufw deny out from 10.0.2.4 to 50.19.226.237

Here is the complete setup

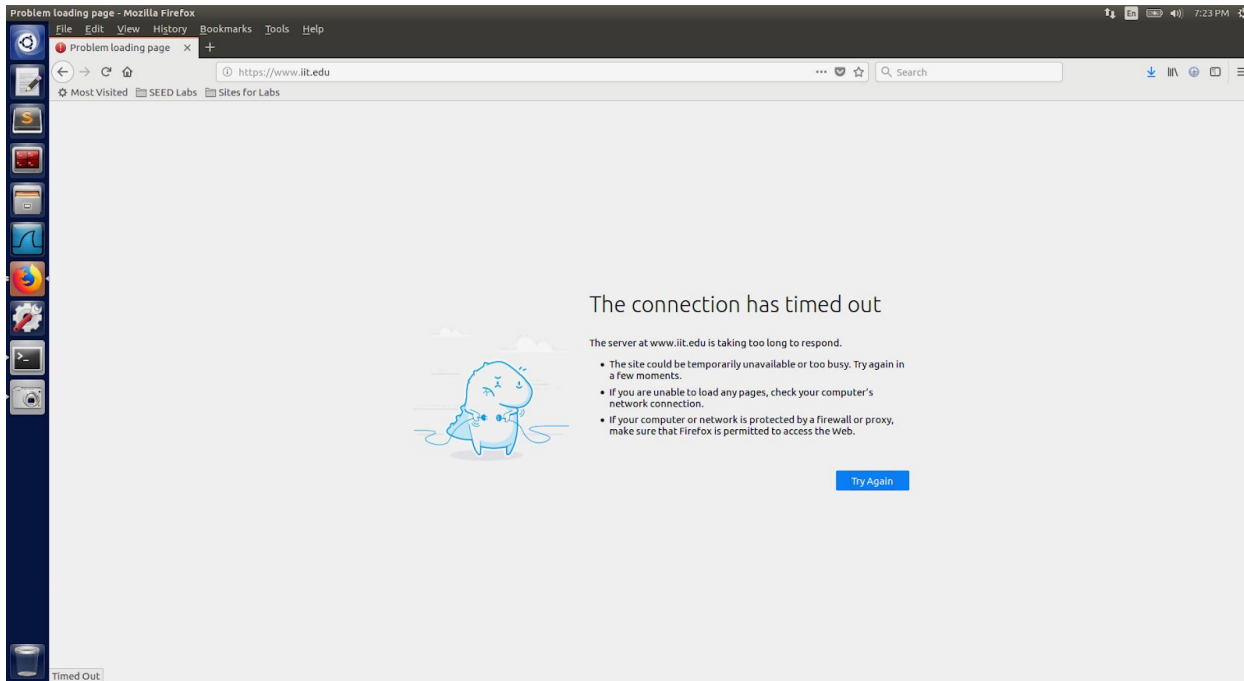


```
Terminal
[05/05/20]seed@VM:~$ sudo ufw deny out 23/tcp
Rule added
Rule added (v6)
[05/05/20]seed@VM:~$ sudo ufw deny out from 10.0.2.4 to 50.19.226.237
Rule added
[05/05/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
23 DENY Anywhere
23 ALLOW 10.0.2.15
23 (v6) DENY Anywhere (v6)
10.0.2.4 23 DENY OUT 10.0.2.15
23/tcp DENY OUT Anywhere
50.19.226.237 DENY OUT 10.0.2.4
23/tcp (v6) DENY OUT Anywhere (v6)

[05/05/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

Once the access to www.iit.edu is revoked by tweaking the firewalls, when tried to access www.iit.edu from the browser, we get the following message



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

I now setup the ssh connection to create tunnel from 10.0.2.4 to 10.0.2.5 using the below command **ssh -L 8000:10.0.2.4:23 10.0.2.4**

On executing the above command, here is how it looks

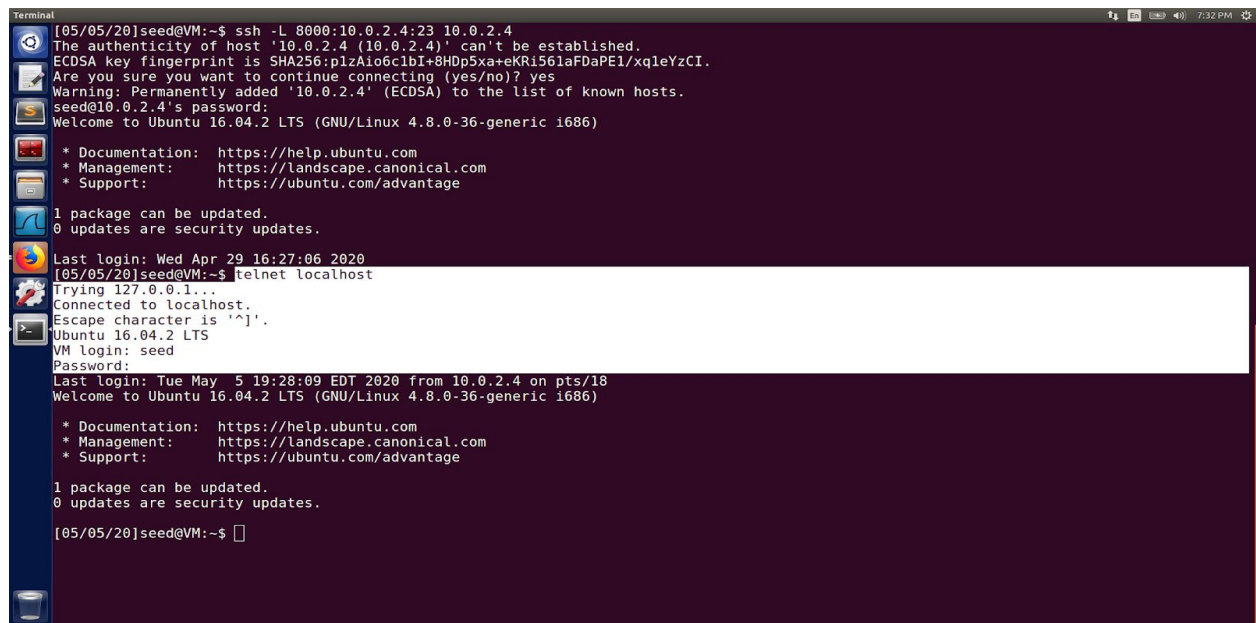
```
Terminal
[05/05/20]seed@VM:~$ ssh -L 8000:10.0.2.4:23 10.0.2.4
The authenticity of host '10.0.2.4 (10.0.2.4)' can't be established.
ECDSA key fingerprint is SHA256:plzAio6c1bI+8Hdp5xa+eKRi561aFDaPE1/xqlEYzCI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.4' (ECDSA) to the list of known hosts.
seed@10.0.2.4'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: Wed Apr 29 16:27:06 2020
[05/05/20]seed@VM:~$
```


Now, we have successfully established a connection to Machine B with an IP 10.0.2.4. Previously we have disabled telnet to any machine from Machine A by tweaking the Firewall rules using ufw. Now once we have ssh'ed into Machine B, we execute telnet to localhost. Which means we're executing the telnet to Machine B from Machine A through a tunnel created in the previous step.



```
[05/05/20]seed@VM:~$ ssh -L 8000:10.0.2.4:23 10.0.2.4
The authenticity of host '10.0.2.4 (10.0.2.4)' can't be established.
ECDSA key fingerprint is SHA256:plzAio6c1bI+8HDp5xa+eKRi561aFDaPEl/xqleYzCI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.4' (ECDSA) to the list of known hosts.
seed@10.0.2.4'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: Wed Apr 29 16:27:06 2020
[05/05/20]seed@VM:~$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue May  5 19:28:09 EDT 2020 from 10.0.2.4 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

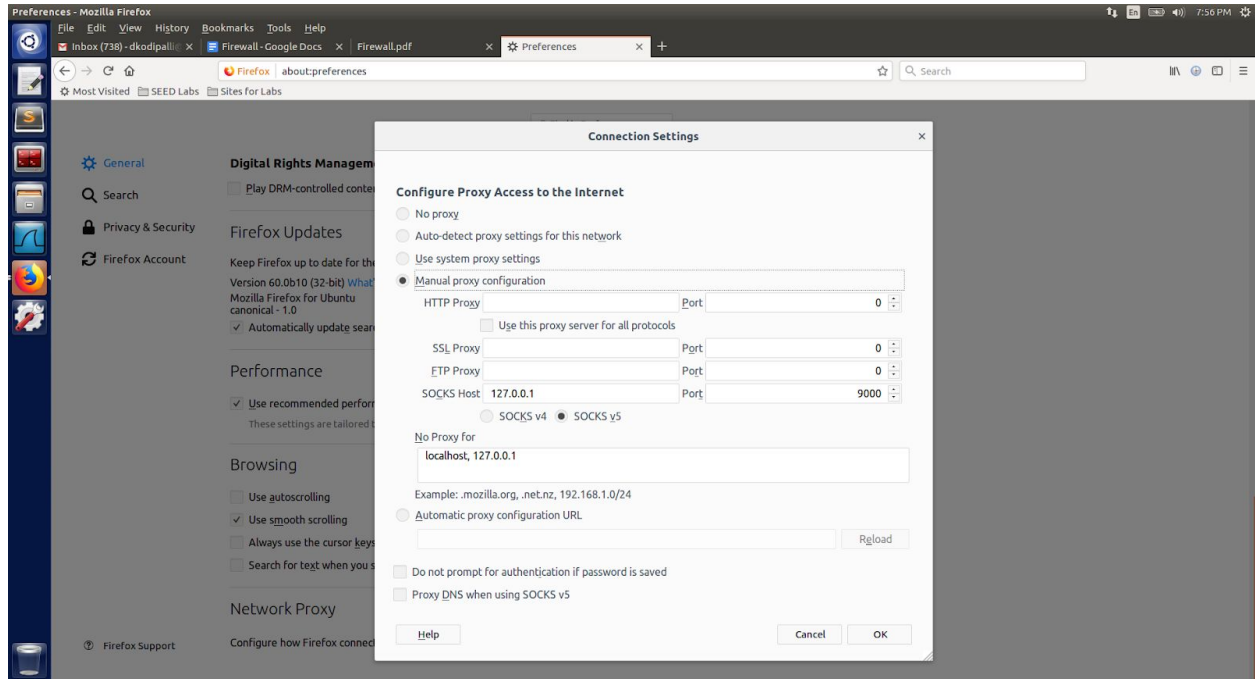
1 package can be updated.
0 updates are security updates.

[05/05/20]seed@VM:~$
```

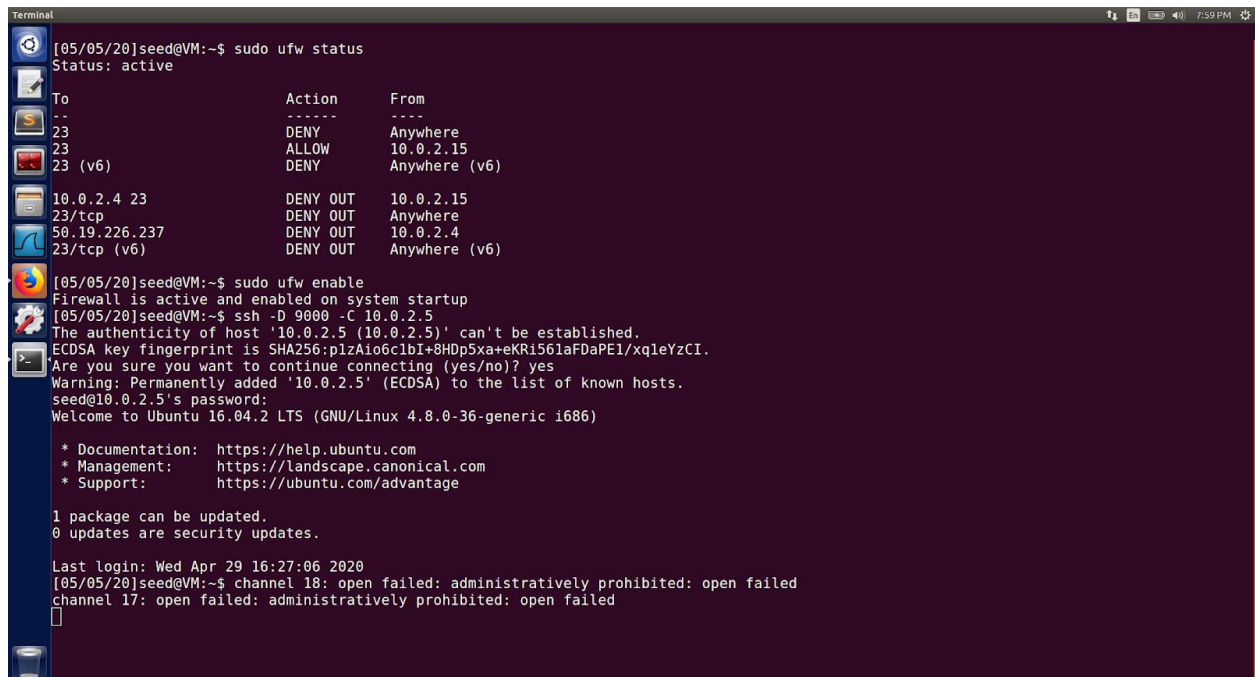
In the screenshot, the highlighted section shows how we telnet'ed to Machine B from Machine A after ssh'ing to Machine B from A. Once the ssh is done, we performed telnet to localhost which means telnetting to Machine B from A. This executed the task successfully. UFW basically blocks all telnet connection going out from Machine A, so to evade it we created an SSH connection between Machine A and Machine B. We could still perform telnet which means the Packet Filters usually don't look at the content inside the packets, they only look at packet level data like IP address and port information.

Task 3.b: Connect to Facebook/(www.iit.edu) using SSH Tunnel.

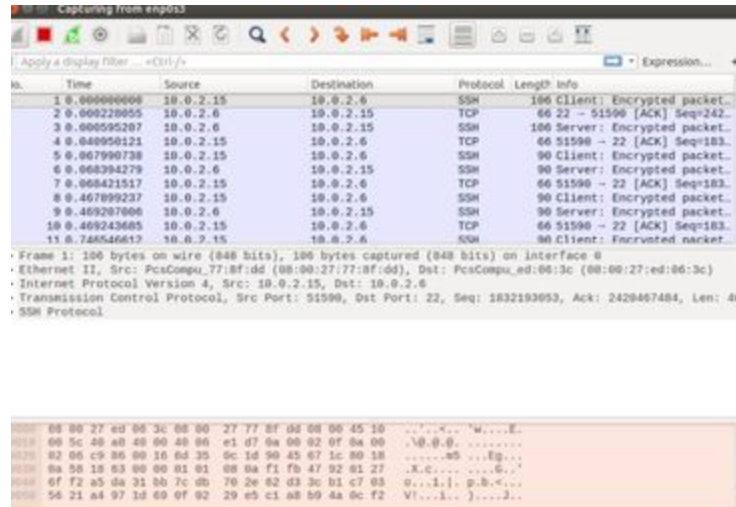
We now configure the firefox in the way shown below:



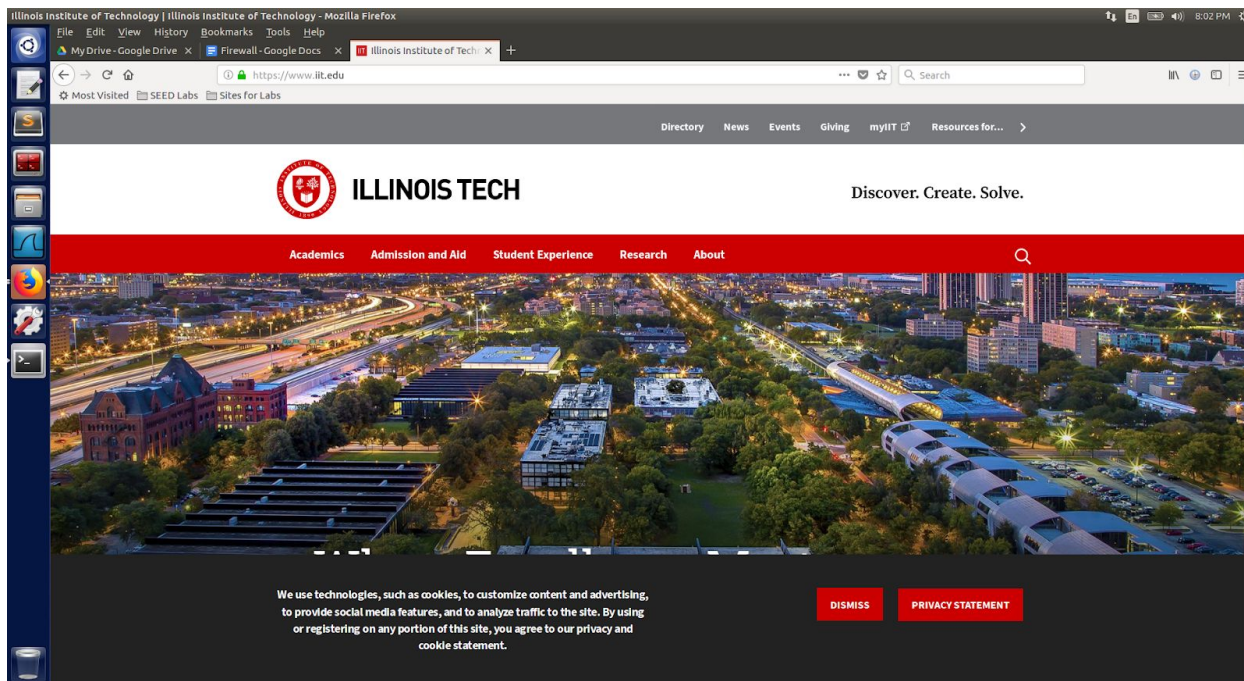
Once the proxy changes are made we now establish a tunnel. Below is how it is done



Once the ssh tunnel is established, I now try to access the www.iit.edu from the browser. This is how it looks when the packets are captured from the wireshark.



Since the Packet Filters usually don't look at the content inside the packets, they only look at packet level data like IP address and port information, we can now see the www.iit.edu page getting loaded on the browser as shows below. This makes this task execute successfully.



Task 4: Evading Ingress Filtering

The objective of this task is to be able to access the web server on Machine A with IP 10.0.2.4 from outside. In order to make this successful, I tweaked the firewall rules using the ufw commands that are shown below.

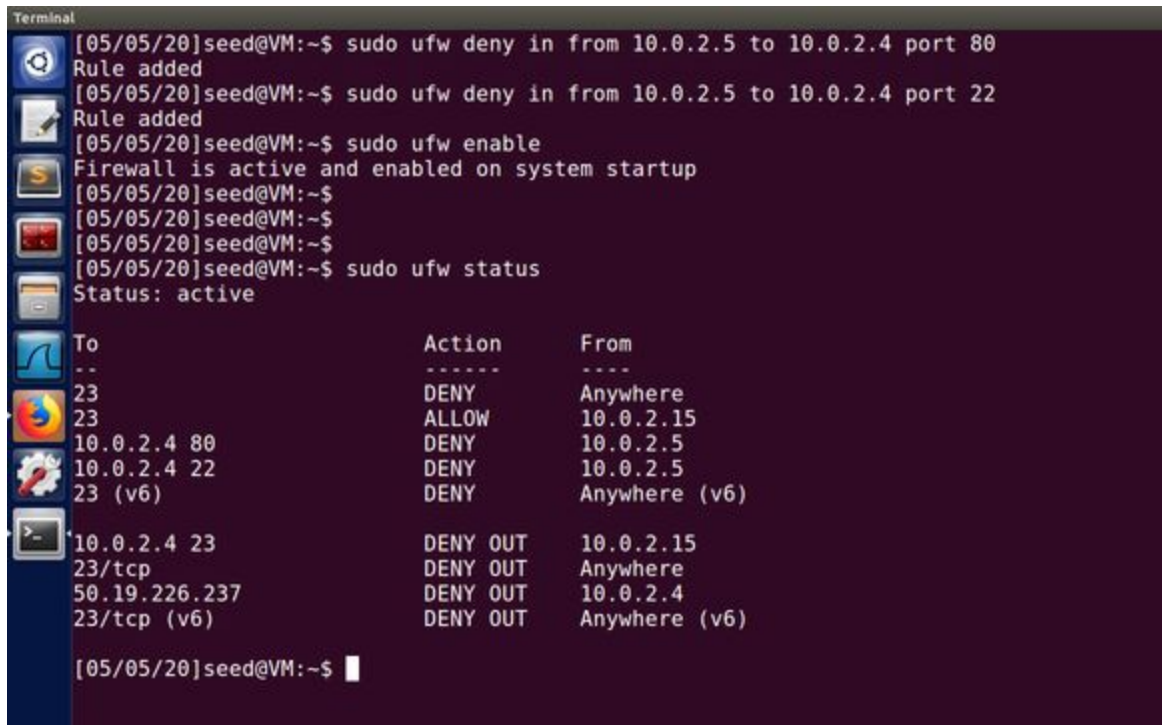
sudo ufw deny in from 10.0.2.5 to 10.0.2.4 port 80

sudo ufw deny in from 10.0.2.5 to 10.0.2.4 port 22

And then enable these firewall rules using

sudo ufw enable

Here is how it looks once we've updated the firewall rules using ufw.

A terminal window titled 'Terminal' showing a series of commands and their outputs. The commands are: 'sudo ufw deny in from 10.0.2.5 to 10.0.2.4 port 80', 'sudo ufw deny in from 10.0.2.5 to 10.0.2.4 port 22', 'sudo ufw enable', and 'sudo ufw status'. The outputs show 'Rule added' for the first two, 'Firewall is active and enabled on system startup' for the third, and a detailed status for the fourth. The status output includes a table of rules.

```
[05/05/20]seed@VM:~$ sudo ufw deny in from 10.0.2.5 to 10.0.2.4 port 80
Rule added
[05/05/20]seed@VM:~$ sudo ufw deny in from 10.0.2.5 to 10.0.2.4 port 22
Rule added
[05/05/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[05/05/20]seed@VM:~$
[05/05/20]seed@VM:~$
[05/05/20]seed@VM:~$
[05/05/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
23 DENY Anywhere
23 ALLOW 10.0.2.15
10.0.2.4 80 DENY 10.0.2.5
10.0.2.4 22 DENY 10.0.2.5
23 (v6) DENY Anywhere (v6)

10.0.2.4 23 DENY OUT 10.0.2.15
23/tcp DENY OUT Anywhere
50.19.226.237 DENY OUT 10.0.2.4
23/tcp (v6) DENY OUT Anywhere (v6)

[05/05/20]seed@VM:~$
```

Now that we have all the rules updated, I now ssh into Machine B with an IP 10.0.2.5. And on Machine A I block Machine B from accessing its port 80 and 22 as done in the above step.


```

Terminal
[05/05/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
---
23 DENY Anywhere
23 ALLOW 10.0.2.15
10.0.2.4 80 DENY 10.0.2.5
10.0.2.4 22 DENY 10.0.2.5
23 (v6) DENY Anywhere (v6)

10.0.2.4 23 DENY OUT 10.0.2.15
23/tcp DENY OUT Anywhere
50.19.226.237 DENY OUT 10.0.2.4
23/tcp (v6) DENY OUT Anywhere (v6)

[05/05/20]seed@VM:~$ ssh -R 8000:localhost:80 10.0.2.5
seed@10.0.2.5's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

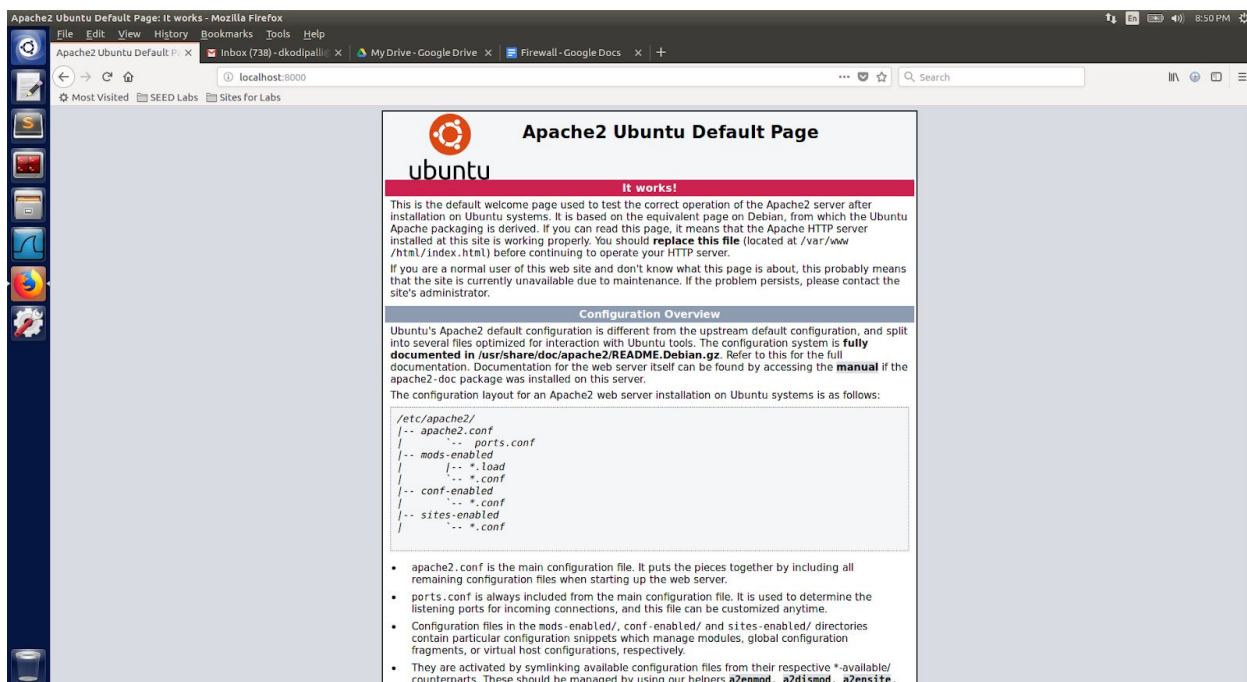
 * Documentation:  https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

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

Last login: Tue May 5 19:59:23 2020 from 10.0.2.4
[05/05/20]seed@VM:~$ firefox localhost:8000
Error: no DISPLAY environment variable specified
[05/05/20]seed@VM:~$ firefox --new-window localhost:8000
Error: no DISPLAY environment variable specified
[05/05/20]seed@VM:~$ export DISPLAY=localhost:8000
[05/05/20]seed@VM:~$ firefox &
[1] 2644
[05/05/20]seed@VM:~$ Failed to connect to Mir: Failed to connect to server socket: No such file or directory
Unable to init server: Broadway display type not supported: localhost:8000
Error: cannot open display: localhost:8000
[05/05/20]seed@VM:~$ 

```

In the above screenshot we can see we have now established an ssh connection to Machine B and when we now run localhost:8000 on the browser we get the following output.



From the output above we can see we've successfully set up a reverse SSH tunnel on Machine A as we were still able to access the protected web server on A from home.