Task 2: Implementing a Simple Firewall

Before starting the task, the default policy file `/etc/default/ufw` is modified, where the `DEFAULT_INPUT_POLICY="DROP"` is changed to `DEFAULT_INPUT_POLICY="ACCEPT"` as shown in Figure 1.

```
# /etc/default/ufw
#

# Set to yes to apply rules to support IPv6 (no means only IPv6 on loopback
# accepted). You will need to 'disable' and then 'enable' the firewall for
# the changes to take affect.
IPV6=yes

# Set the default input policy to ACCEPT, DROP, or REJECT. Please note that if
# you change this you will most likely want to adjust your rules.
DEFAULT_INPUT_POLICY="ACCEPT"
```

Figure 1: Modify /etc/default/ufw

#1 Prevent VM A (`10.0.2.4`) Telnet to VM B (`10.0.2.5`):

Figure 2 shows the completed attempt in proving that we can prevent a successful Telnet connection from VM A (`10.0.2.4`) to VM B (`10.0.2.5`).

```
[04/04/20]seed@VM:~/.../Lab7$ 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: seed
Password:
Last login: Sat Apr  4 09:07:11 EDT 2020 from 10.0.2.4 on pts/17
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

0 packages can be updated.
0 updates are security updates.

[04/04/20]seed@VM:~$ exit
logout
Connection closed by foreign host.
[04/04/20]seed@VM:~/.../Lab7$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/NetSec/Lab7 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/04/20]seed@VM:~/.../Lab7$ sudo insmod task2.ko
[04/04/20]seed@VM:~/.../Lab7$ lsmod | grep task2
task2                  16384  0
[04/04/20]seed@VM:~/.../Lab7$ telnet 10.0.2.5
Trying 10.0.2.5...
telnet: Unable to connect to remote host: Connection timed out
```

Figure 2: Prevent VM A (10.0.2.4) Telnet to VM B (10.0.2.5)

Before introducing anything, when we attempt to Telnet from VM A (`10.0.2.4`) to VM B (`10.0.2.5`) and we see that we can successfully Telnet in as shown in Figure 3.

```
[04/04/20]seed@VM:~/.../Lab7$ 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: seed
Password:
Last login: Sat Apr  4 09:07:11 EDT 2020 from 10.0.2.4 on pts/17
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

0 packages can be updated.
0 updates are security updates.
```

Figure 3: Successful Telnet from VM A (10.0.2.4) to VM B (10.0.2.5)

Using Loadable Kernel Module (LKM), a new module can be added to the kernel at runtime, so the packet filtering part of a firewall can be implemented. For the filtering module to block incoming/outgoing packets, the module must be inserted into the packet processing path. This is completed by using `Netfilter`.

To support the first rule required where we have to prevent a successful Telnet connection from VM A (`10.0.2.4`) to VM B (`10.0.2.5`), a program named `task2.c` is written and the specific firewall rule is shown in Figure 4. A `Makefile` is created as shown in Figure 5 and the command `make` is run, compiling the program into a loadable kernel module as shown in Figure 6.

```
// Rule 1: Preventing VM A (10.0.2.4) from doing telnet to VM B (10.0.2.5)
if (iph->protocol == 6 &&                 //TCP protocol
    tcph->dest == htons(23) &&            //dest_port
    iph->saddr == in_aton("10.0.2.4") &&  //src_ip
    iph->daddr == in_aton("10.0.2.5")     //dest_ip
    )
    {
        return NF_DROP;
    }
    else
    {
        return NF_ACCEPT;
    }
```

Figure 4: Prevent Telnet from A to B Firewall Rule

```
obj-m += task2.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
```

Figure 5: Makefile for LKM & Netfilter

```
[04/04/20]seed@VM:~/.../Lab7$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/NetSec/Lab7 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
```

Figure 6: Compiling Program into LKM

After the module is built, the module can be inserted by using `sudo insmod task2.ko` and we verify that the module has been inserted by using `lsmod | grep task2` as shown in Figure 7.

```
[04/04/20]seed@VM:~/.../Lab7$ sudo insmod task2.ko
[04/04/20]seed@VM:~/.../Lab7$ lsmod | grep task2
task2                  16384  0
```

Figure 7: Insert module

Upon completion of all the above steps, we see that we are no longer able to successfully establish a Telnet connection from VM A (`10.0.2.4`) to VM B (`10.0.2.5`) as shown in Figure 8, showing that our new rule is successful.

```
[04/04/20]seed@VM:~/.../Lab7$ telnet 10.0.2.5
Trying 10.0.2.5...
telnet: Unable to connect to remote host: Connection timed out
```

Figure 8: Unsuccessful Telnet from VM A (10.0.2.4) to VM B (10.0.2.5)

#2 Prevent VM A (`10.0.2.4`) from visiting a website (`www.syr.edu`):

Figure 9 shows the completed attempt in proving that we can prevent VM A (`10.0.2.4`) from visiting a website, `www.syr.edu` (`128.230.18.198`).

```
[04/04/20]seed@VM:~/.../Lab7$ ping www.syr.edu
PING syr.edu (128.230.18.198) 56(84) bytes of data.
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=1 ttl=52 time=256 ms
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=2 ttl=52 time=255 ms
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=3 ttl=52 time=258 ms
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=4 ttl=52 time=256 ms
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=5 ttl=52 time=255 ms
^C
--- syr.edu ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 255.652/256.436/258.051/1.029 ms
[04/04/20]seed@VM:~/.../Lab7$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/NetSec/Lab7 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/NetSec/Lab7/task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/NetSec/Lab7/task2.mod.o
  LD [M]  /home/seed/NetSec/Lab7/task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/04/20]seed@VM:~/.../Lab7$ sudo insmod task2.ko
[04/04/20]seed@VM:~/.../Lab7$ lsmod | grep task2
task2                  16384  0
[04/04/20]seed@VM:~/.../Lab7$ ping www.syr.edu
PING syr.edu (128.230.18.198) 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
--- syr.edu ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4095ms
```

Figure 9: Prevent VM A (10.0.2.4) from visiting a website (www.syr.edu)

Before introducing anything, when we attempt to ping `www.syr.edu` from VM A (`10.0.2.4`), we see that we the packets are successfully transmitted as shown in Figure 10.

```
[04/04/20]seed@VM:~/.../Lab7$ ping www.syr.edu
PING syr.edu (128.230.18.198) 56(84) bytes of data.
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=1 ttl=52 time=256 ms
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=2 ttl=52 time=255 ms
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=3 ttl=52 time=258 ms
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=4 ttl=52 time=256 ms
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=5 ttl=52 time=255 ms
^C
--- syr.edu ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 255.652/256.436/258.051/1.029 ms
```

Figure 10: Successful ping from VM A (10.0.2.4) to www.syr.edu

To support the second rule required where we have to prevent VM A (`10.0.2.4`) from visiting a website, `www.syr.edu` (`128.230.18.198`), a program named `task2.c` is written and the specific firewall rule is shown in Figure 11. The same `Makefile` as used in Figure 5 above is utilized and the command `make` is run, compiling the program into a loadable kernel module as shown in Figure 12.

```
// Rule 2: Preventing VM A (10.0.2.4) from visiting a website (www.syr.edu)
if (iph->saddr == in_aton("10.0.2.4") &&  //src_ip
    iph->daddr == in_aton("128.230.18.198")     //dest_ip
    )
    {
        return NF_DROP;
    }
    else
    {
        return NF_ACCEPT;
    }
```

Figure 11: Prevent access to www.syr.edu Firewall Rule

```
[04/04/20]seed@VM:~/.../Lab7$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/NetSec/Lab7 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/NetSec/Lab7/task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/NetSec/Lab7/task2.mod.o
  LD [M]  /home/seed/NetSec/Lab7/task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
```

Figure 12: Compiling Program into LKM

After the module is built, the module can be inserted by using `sudo insmod task2.ko` and we verify that the module has been inserted by using `lsmod | grep task2` as shown in Figure 13.

```
[04/04/20]seed@VM:~/.../Lab7$ sudo insmod task2.ko
[04/04/20]seed@VM:~/.../Lab7$ lsmod | grep task2
task2                  16384  0
```

Figure 13: Insert module

Upon completion of all the above steps, we see that VM A (`10.0.2.4`) is no longer able to visit a website, `www.syr.edu` (`128.230.18.198`), as shown from the unsuccessful ping as shown in Figure 14, showing that our new rule is successful.

```
[04/04/20]seed@VM:~/.../Lab7$ ping www.syr.edu
PING syr.edu (128.230.18.198) 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
--- syr.edu ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4095ms
```

Figure 14: Unsuccessful ping from VM A (10.0.2.4) to www.syr.edu

#3 Prevent VM A (10.0.2.4) SSH to VM B (10.0.2.5):

Figure 15 shows the completed attempt in proving that we can prevent VM A (10.0.2.4) from successfully establishing an SSH tunnel to VM B (10.0.2.5).

```
[04/04/20]seed@VM:~/.../Lab7$ ssh 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

0 packages can be updated.
0 updates are security updates.

Last login: Sat Apr  4 09:44:02 2020 from 10.0.2.4
[04/04/20]seed@VM:~$ exit
logout
Connection to 10.0.2.5 closed.
[04/04/20]seed@VM:~/.../Lab7$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/NetSec/Lab7 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/NetSec/Lab7/task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/NetSec/Lab7/task2.mod.o
  LD [M]  /home/seed/NetSec/Lab7/task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/04/20]seed@VM:~/.../Lab7$ sudo insmod task2.ko
[04/04/20]seed@VM:~/.../Lab7$ lsmod | grep task2
task2                  16384  0
[04/04/20]seed@VM:~/.../Lab7$ ssh 10.0.2.5
ssh: connect to host 10.0.2.5 port 22: Connection timed out
```

Figure 15: Prevent VM A (10.0.2.4) SSH to VM B (10.0.2.5)

Before introducing anything, when we attempt to establish an SSH tunnel to VM B (10.0.2.5) from VM A (10.0.2.4), we see that we can successfully establish the SSH tunnel as shown in Figure 16.

```
[04/04/20]seed@VM:~/.../Lab7$ ssh 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

0 packages can be updated.
0 updates are security updates.

Last login: Sat Apr  4 09:44:02 2020 from 10.0.2.4
```

Figure 16: Successful SSH Tunnel between VM A (10.0.2.4) and VM B (10.0.2.5)

To support the third rule required where we have to prevent VM A (`10.0.2.4`) from successfully establishing an SSH tunnel to VM B (`10.0.2.5`), a program named `task2.c` is written, and the specific firewall rule is shown in Figure 17. The same `Makefile` as used in Figure 5 above is utilized and the command `make` is run, compiling the program into a loadable kernel module as shown in Figure 18.

```
// Rule 3: Preventing VM A (10.0.2.4) from doing SSH to VM B (10.0.2.5)
if (tcph->dest == htons(22) &&            //dest_port
    iph->saddr == in_aton("10.0.2.4") &&  //src_ip
    iph->daddr == in_aton("10.0.2.5")     //dest_ip
    )
    {
        return NF_DROP;
    }
    else
    {
        return NF_ACCEPT;
    }
```

Figure 17: Prevent VM A (10.0.2.4) SSH to VM B (10.0.2.5) Firewall Rule

```
[04/04/20]seed@VM:~/.../Lab7$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/NetSec/Lab7 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/NetSec/Lab7/task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/NetSec/Lab7/task2.mod.o
  LD [M]  /home/seed/NetSec/Lab7/task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
```

Figure 18: Compiling Program into LKM

After the module is built, the module can be inserted by using `sudo insmod task2.ko` and we verify that the module has been inserted by using `lsmod | grep task2` as shown in Figure 19.

```
[04/04/20]seed@VM:~/.../Lab7$ sudo insmod task2.ko
[04/04/20]seed@VM:~/.../Lab7$ lsmod | grep task2
task2                  16384  0
```

Figure 19: Insert module

Upon completion of all the above steps, we see that VM A (`10.0.2.4`) is no longer able to successfully establish an SSH tunnel to VM B (`10.0.2.5`), as shown in Figure 20, showing that our new rule is successful.

```
[04/04/20]seed@VM:~/.../Lab7$ ssh 10.0.2.5
ssh: connect to host 10.0.2.5 port 22: Connection timed out
```

Figure 20: Unsuccessful SSH Tunnel between VM A (10.0.2.4) and VM B (10.0.2.5)

Task 3: Evading Egress Filtering

Machine		IP Address
A	Behind firewall (inside organization's network)	`10.0.2.4`
B	Outside of firewall	`10.0.2.5`
C	External Telnet server	`10.0.2.6`

Before implementing any firewall rules, we can see from Figure 21 that we are able to successfully Telnet to Machine B (`10.0.2.5`) and Machine C (`10.0.2.6`) from Machine A (`10.0.2.4`).

```
[04/04/20]seed@VM:~/.../Lab7$ 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: seed
Password:
Last login: Sat Apr  4 09:44:21 EDT 2020 from 10.0.2.4 on pts/17
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

0 packages can be updated.
0 updates are security updates.

[04/04/20]seed@VM:~$ exit
logout
Connection closed by foreign host.
[04/04/20]seed@VM:~/.../Lab7$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue Feb 18 21:00:42 EST 2020 from 10.0.2.5 on pts/17
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

0 packages can be updated.
0 updates are security updates.

[04/04/20]seed@VM:~$ exit
logout
Connection closed by foreign host.
```

Figure 21: Successful Telnet from Machine A (10.0.2.4) to Machine B (10.0.2.5) & Machine C (10.0.2.6)

In order to block all the outgoing traffic to external telnet servers such as Machine C (`10.0.2.6`), the following firewall rule is applied on the firewall run on Machine A (`10.0.2.4`) as shown in Figure 22. A `Makefile` is created as shown in Figure 23 and the command `make` is run, compiling the program into a loadable kernel module as shown in Figure 24.

```
// Preventing VM A (10.0.2.4) from doing telnet to others
if (iph->protocol == 6 &&                 //TCP protocol
    tcph->dest == htons(23) &&            //dest_port
    iph->saddr == in_aton("10.0.2.4")     //src_ip
    )
    {
        return NF_DROP;
    }
    else
    {
        return NF_ACCEPT;
    }
```

Figure 22: Prevent Machine A from establishing Telnet connections with others Firewall Rule

```
obj-m += task3.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
```

Figure 23: Makefile for LKM & Netfilter

```
[04/04/20]seed@VM:~/.../Lab7$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/NetSec/Lab7 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/04/20]seed@VM:~/.../Lab7$ sudo insmod task3.ko
[04/04/20]seed@VM:~/.../Lab7$ lsmod | grep task3
task3                  16384  0
```

Figure 24: Compiling Program into LKM

Upon completion of all the above steps, we see that VM A (`10.0.2.4`) is no longer able to successfully establish a Telnet connection to VM B (`10.0.2.5`) or VM C (`10.0.2.6`), as shown in Figure 25, showing that our new rule is successful.

```
[04/04/20]seed@VM:~/.../Lab7$ telnet 10.0.2.5
Trying 10.0.2.5...
telnet: Unable to connect to remote host: Connection timed out
[04/04/20]seed@VM:~/.../Lab7$ telnet 10.0.2.6
Trying 10.0.2.6...
telnet: Unable to connect to remote host: Connection timed out
```

Figure 25: Unsuccessful Telnet from Machine A (10.0.2.4) to Machine B (10.0.2.5) & Machine C (10.0.2.6)

Figure 26 shows the completed attempt in proving that we can block all the outgoing traffic to www.facebook.com from VM A (`10.0.2.4`).

```
[04/05/20]seed@VM:~/.../Lab7$ ping www.facebook.com
PING star-mini.c10r.facebook.com (157.240.13.35) 56(84) bytes of data.
64 bytes from edge-star-mini-shv-02-sin6.facebook.com (157.240.13.35): icmp_seq=1 ttl=57 time=4.45 ms
64 bytes from edge-star-mini-shv-02-sin6.facebook.com (157.240.13.35): icmp_seq=2 ttl=57 time=4.32 ms
64 bytes from edge-star-mini-shv-02-sin6.facebook.com (157.240.13.35): icmp_seq=3 ttl=57 time=10.6 ms
^C
--- star-mini.c10r.facebook.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 4.328/6.482/10.659/2.954 ms
[04/05/20]seed@VM:~/.../Lab7$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/NetSec/Lab7 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/05/20]seed@VM:~/.../Lab7$ sudo insmod task3.ko
[04/05/20]seed@VM:~/.../Lab7$ lsmod | grep task3
task3                  16384  0
[04/05/20]seed@VM:~/.../Lab7$ ping www.facebook.com
PING star-mini.c10r.facebook.com (157.240.13.35) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- star-mini.c10r.facebook.com ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2034ms
```

Figure 26: Prevent Machine A from establishing connections with www.facebook.com

Before introducing anything, when we attempt to ping www.facebook.com VM A (`10.0.2.4`), we see that the packet transmission is successful as shown in Figure 27.

```
[04/05/20]seed@VM:~/.../Lab7$ ping www.facebook.com
PING star-mini.c10r.facebook.com (157.240.13.35) 56(84) bytes of data.
64 bytes from edge-star-mini-shv-02-sin6.facebook.com (157.240.13.35): icmp_seq=1 ttl=57 time=4.45 ms
64 bytes from edge-star-mini-shv-02-sin6.facebook.com (157.240.13.35): icmp_seq=2 ttl=57 time=4.32 ms
64 bytes from edge-star-mini-shv-02-sin6.facebook.com (157.240.13.35): icmp_seq=3 ttl=57 time=10.6 ms
^C
--- star-mini.c10r.facebook.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 4.328/6.482/10.659/2.954 ms
```

Figure 27: Successful packet transmission from Machine A to www.facebook.com

In order to block all the outgoing traffic to to www.facebook.com from VM A (`10.0.2.4`), the following firewall rule is applied on the firewall run on Machine A (`10.0.2.4`) as shown in Figure 28. The same `Makefile` as shown in Figure 23 is used and the command `make` is run, compiling the program into a loadable kernel module as shown in Figure 29.

```
// Preventing VM A (10.0.2.4) from visiting a website (www.facebook.com)
if (iph->saddr == in_aton("10.0.2.4") &&      //src_ip
    iph->daddr == in_aton("157.240.13.35")    //dest_ip
    )
    {
        return NF_DROP;
    }
    else
    {
        return NF_ACCEPT;
    }
```

Figure 28: Prevent Machine A from establishing connections with www.facebook.com Firewall Rule

```
[04/05/20]seed@VM:~/.../Lab7$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/NetSec/Lab7 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/05/20]seed@VM:~/.../Lab7$ sudo insmod task3.ko
[04/05/20]seed@VM:~/.../Lab7$ lsmod | grep task3
task3                  16384  0
```

Figure 29: Compiling Program into LKM

Upon completion of all the above steps, we see that the packet transmission from VM A (`10.0.2.4`) to www.facebook.com is no longer successful as shown in Figure 30, showing that our new rule is successful.

```
[04/05/20]seed@VM:~/.../Lab7$ ping www.facebook.com
PING star-mini.c10r.facebook.com (157.240.13.35) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- star-mini.c10r.facebook.com ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2034ms
```

Figure 30: Unsuccessful packet transmission from Machine A to www.facebook.com

Task 3a: Telnet to Machine B through the firewall

An SSH tunnel is set up between Machine A's (`10.0.2.4`) port 8000 and Machine B's (`10.0.2.5`) port 22, the standard SSH port. When packets come out of Machine B's end, it will be forwarded to Machine C's (`10.0.2.6`) port 23, the standard Telnet port. This is done by running the command, `ssh -L 8000:10.0.2.6:23 10.0.2.5` and we can see the successful SSH tunnel being set up in Figure 31.

```
[04/04/20]seed@VM:~/.../Lab7$ hostname -I
10.0.2.4
[04/04/20]seed@VM:~/.../Lab7$ ssh -L 8000:10.0.2.6:23 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

0 packages can be updated.
0 updates are security updates.

Last login: Sat Apr  4 22:30:58 2020 from 10.0.2.4
[04/04/20]seed@VM:~$ hostname -I
10.0.2.5
```

Figure 31: Successful SSH tunnel between Machine A (10.0.2.4) and Machine B (10.0.2.5)

The successful setup can also be seen from the PCAP shown in Figure 32.

Source	Destination	Protoco	Info
10.0.2.4	10.0.2.5	TCP	34066 → 22 [SYN] Seq=1848582532 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=15…
10.0.2.5	10.0.2.4	TCP	22 → 34066 [SYN, ACK] Seq=1381245569 Ack=1848582533 Win=28960 Len=0 MSS=1460 …
10.0.2.4	10.0.2.5	TCP	34066 → 22 [ACK] Seq=1848582533 Ack=1381245570 Win=29312 Len=0 TSval=1519635 …
10.0.2.4	10.0.2.5	SSHv2	Client: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.2)
10.0.2.5	10.0.2.4	TCP	22 → 34066 [ACK] Seq=1381245570 Ack=1848582574 Win=29056 Len=0 TSval=2118271 …
10.0.2.5	10.0.2.4	SSHv2	Server: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.2)
10.0.2.4	10.0.2.5	TCP	34066 → 22 [ACK] Seq=1848582574 Ack=1381245611 Win=29312 Len=0 TSval=1519638 …
10.0.2.5	10.0.2.4	SSHv2	Server: Key Exchange Init
10.0.2.4	10.0.2.5	TCP	34066 → 22 [ACK] Seq=1848582574 Ack=1381246587 Win=31232 Len=0 TSval=1519638 …
10.0.2.4	10.0.2.5	SSHv2	Client: Key Exchange Init
10.0.2.5	10.0.2.4	TCP	22 → 34066 [ACK] Seq=1381246587 Ack=1848583910 Win=31872 Len=0 TSval=2118285 …
10.0.2.4	10.0.2.5	SSHv2	Client: Diffie-Hellman Key Exchange Init
10.0.2.5	10.0.2.4	TCP	22 → 34066 [ACK] Seq=1381246587 Ack=1848583958 Win=31872 Len=0 TSval=2118285 …
10.0.2.5	10.0.2.4	SSHv2	Server: Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet (len=84)
10.0.2.4	10.0.2.5	SSHv2	Client: New Keys
10.0.2.5	10.0.2.4	TCP	22 → 34066 [ACK] Seq=1381246951 Ack=1848583974 Win=31872 Len=0 TSval=2118302 …
10.0.2.4	10.0.2.5	SSHv2	Client: Encrypted packet (len=44)
10.0.2.5	10.0.2.4	TCP	22 → 34066 [ACK] Seq=1381246951 Ack=1848584018 Win=31872 Len=0 TSval=2118302 …
10.0.2.5	10.0.2.4	SSHv2	Server: Encrypted packet (len=44)
10.0.2.4	10.0.2.5	SSHv2	Client: Encrypted packet (len=60)
10.0.2.5	10.0.2.4	SSHv2	Server: Encrypted packet (len=52)

Figure 32: PCAP of successful SSH tunnel between Machine A (10.0.2.4) and Machine B (10.0.2.5)

On Machine A (`10.0.2.4`), we run the command telnet localhost 8000 and we can see that we can successfully Telnet from Machine A (`10.0.2.4`) to Machine C (`10.0.2.6`) as shown in Figure 33.

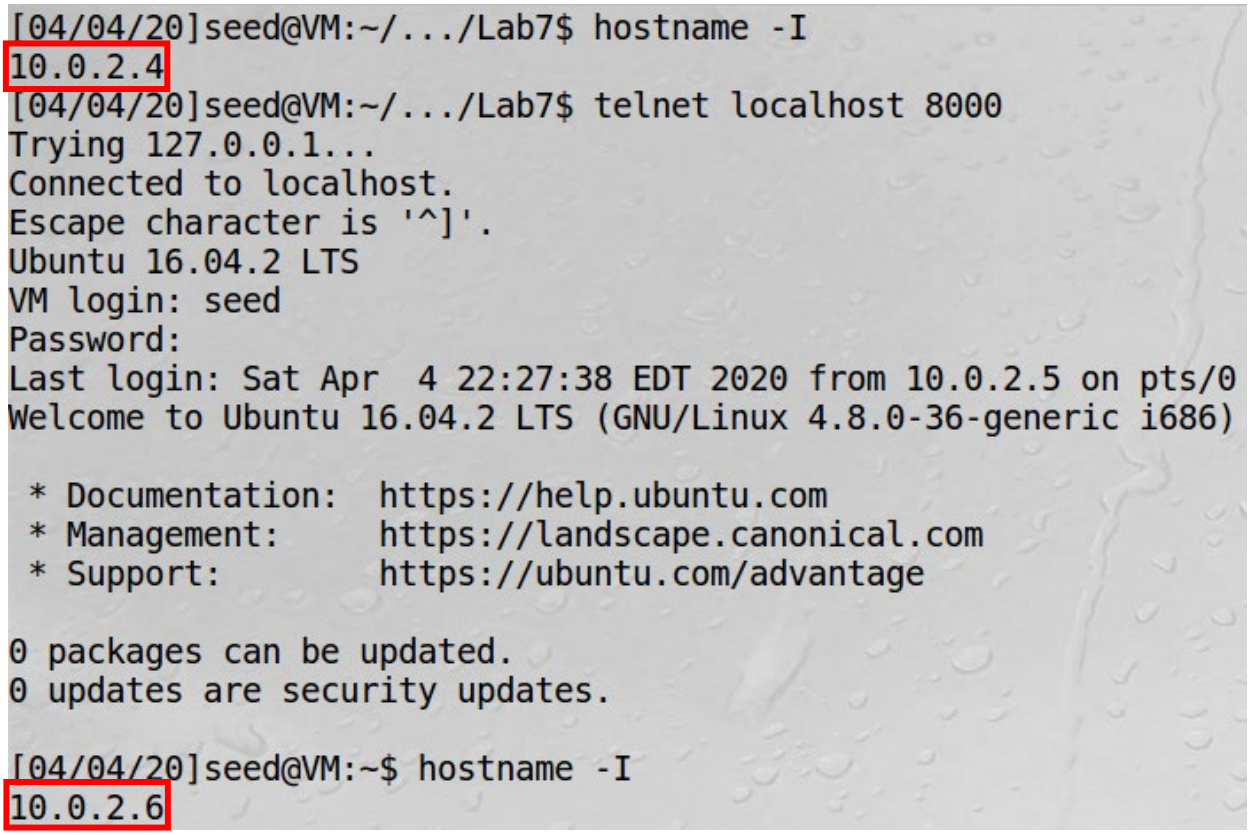


Figure 33: Successful Telnet connection from Machine A (10.0.2.4) to Machine C (10.0.2.6)

All the telnet traffic goes through port 8000 from Machine A (`10.0.2.4`) to Machine B (`10.0.2.5`) through port 22, the SSH port as shown in Figure 34.

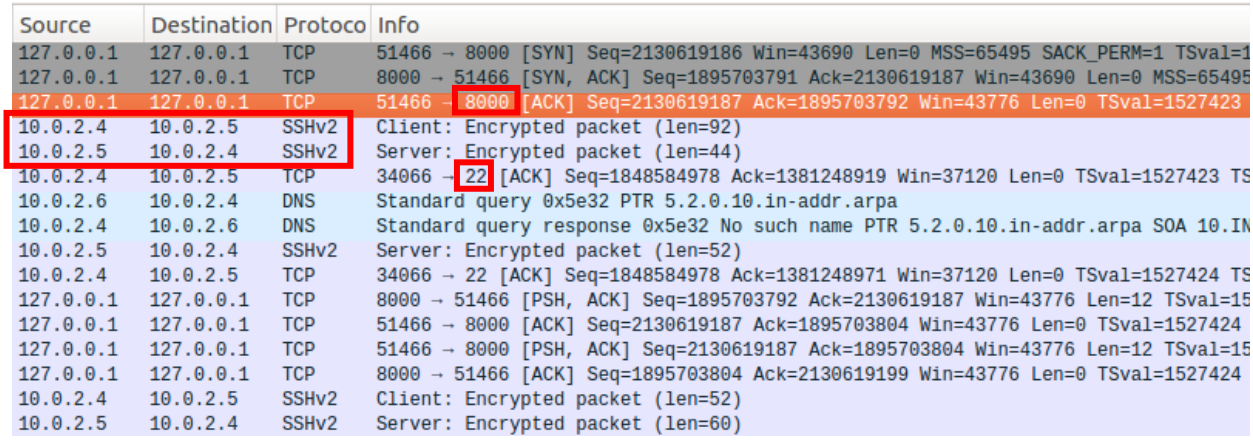


Figure 34: PCAP showing packet flow

Therefore, this shows that we are essentially able to Telnet to Machine C (`10.0.2.6`) from Machine A (`10.0.2.4`) by bypassing the firewall with an SSH tunnel setup between Machine A (`10.0.2.4`) and Machine B (`10.0.2.5`).

Task 3b: Connect to Facebook using SSH Tunnel

Before establishing the SSH tunnel, we can see that we are unable to successfully ping www.facebook.com using Machine A (`10.0.2.4`).

```
[04/05/20]seed@VM:~/.../Lab7$ hostname -I
10.0.2.4
[04/05/20]seed@VM:~/.../Lab7$ ping www.facebook.com
PING star-mini.c10r.facebook.com (157.240.13.35) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- star-mini.c10r.facebook.com ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2044ms
```

Figure 35: Unsuccessful ping to www.facebook.com from Machine A (10.0.2.4) before SSH

To achieve dynamic port forwarding, we specify the local port number instead of the final destination when we set up the SSH tunnel between Machine A's (`10.0.2.4`) port 9000 and Machine B's (`10.0.2.5`) port 22, the standard SSH port, using `ssh -D 9000 -C 10.0.2.5` as shown in Figure 36, which allows us to successfully ping www.facebook.com.

```
[04/05/20]seed@VM:~/.../Lab7$ hostname -I
10.0.2.4
[04/05/20]seed@VM:~/.../Lab7$ ssh -D 9000 -C 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

0 packages can be updated.
0 updates are security updates.

Last login: Sun Apr  5 00:22:25 2020 from 10.0.2.4
[04/05/20]seed@VM:~$ hostname -I
10.0.2.5
[04/05/20]seed@VM:~$ ping www.facebook.com
PING star-mini.c10r.facebook.com (157.240.13.35) 56(84) bytes of data.
64 bytes from edge-star-mini-shv-02-sin6.facebook.com (157.240.13.35): icmp_seq=1 ttl=57 time=43.6 ms
64 bytes from edge-star-mini-shv-02-sin6.facebook.com (157.240.13.35): icmp_seq=2 ttl=57 time=24.2 ms
64 bytes from edge-star-mini-shv-02-sin6.facebook.com (157.240.13.35): icmp_seq=3 ttl=57 time=4.65 ms
^C
--- star-mini.c10r.facebook.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 4.656/24.181/43.664/15.924 ms
```

Figure 36: Successful ping to www.facebook.com from Machine A (10.0.2.4) after SSH

In order to ensure that the traffic goes through our SSH tunnel, every time Firefox needs to connect to a web server, we get Machine A (`10.0.2.4`) to use `localhost:9000` as its proxy using the `SOCKS Proxy` to support dynamic port forwarding as shown in Figure 37.

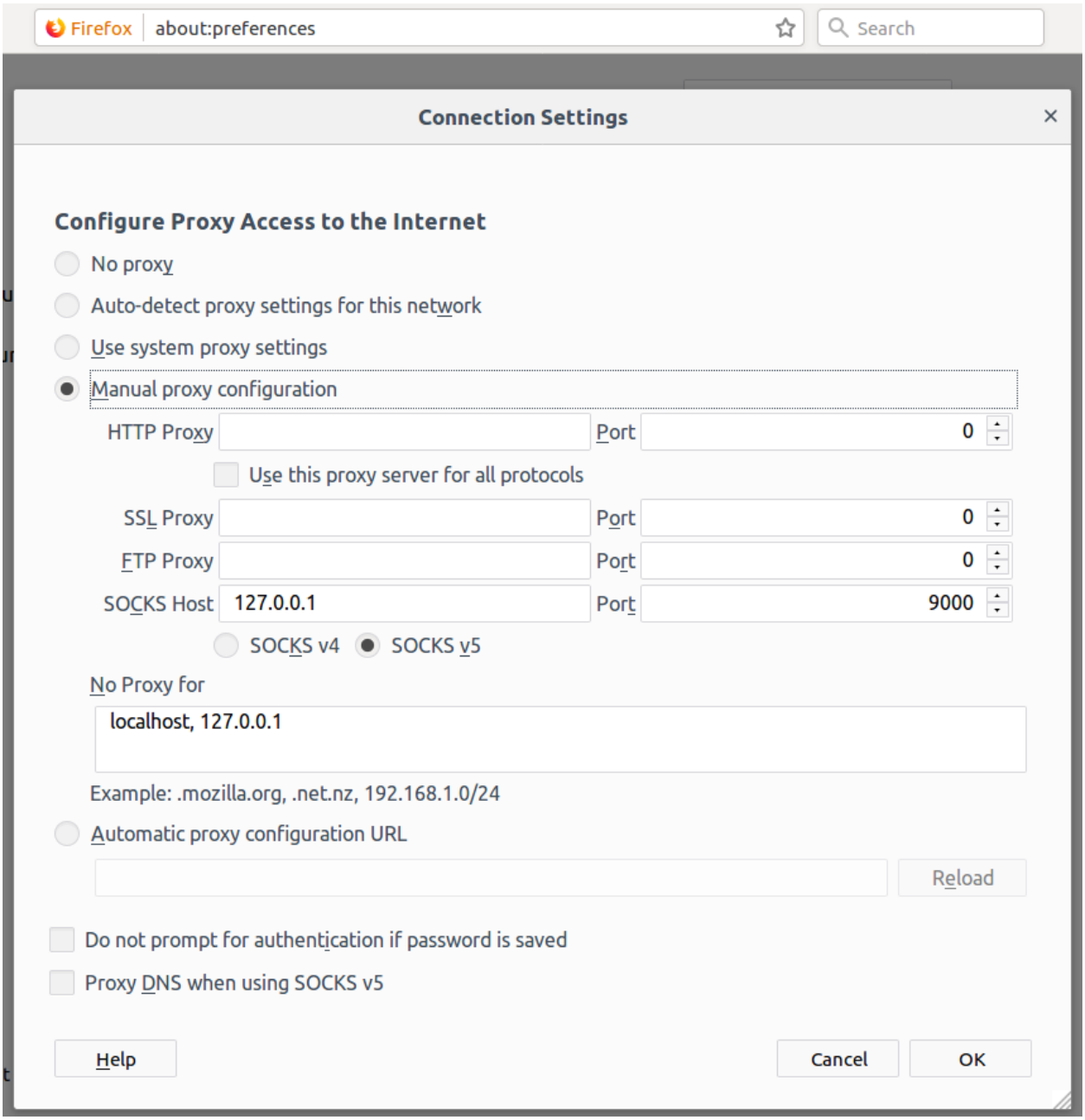


Figure 37: Configure Proxy Access of Firefox

1. Run Firefox and go visit the Facebook page. Can you see the Facebook page? Please describe your observation.

Yes, you can see the Facebook page as shown in Figure 38. The Facebook page appears to load as per normal.

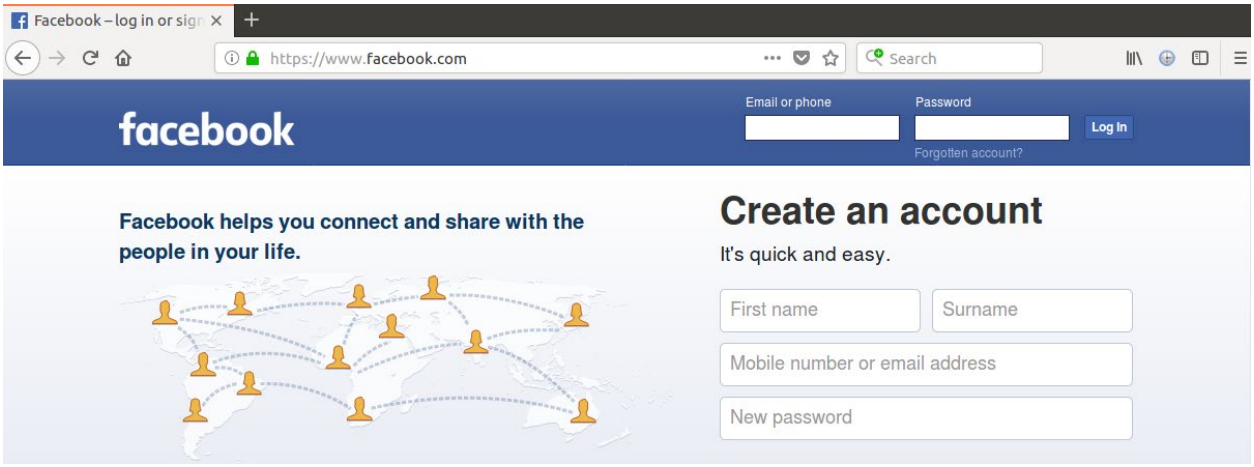


Figure 38: Facebook Page Successfully Loaded

2. After you get the Facebook page, break the SSH tunnel, clear the Firefox cache, and try the connection again. Please describe your observation.

No, you cannot see the Facebook page. Instead, there is an error page stating that "The proxy server is refusing connections" as shown in Figure 39, just as we have set previously in Figure 37.

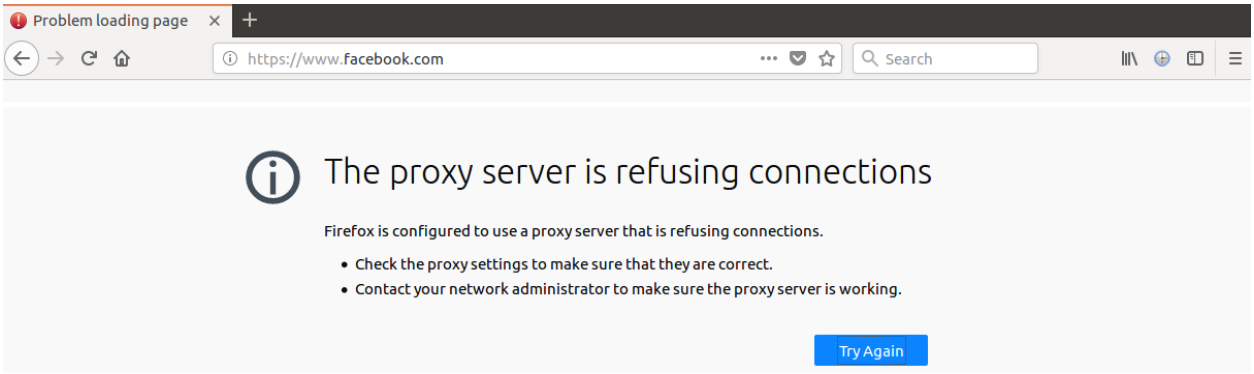


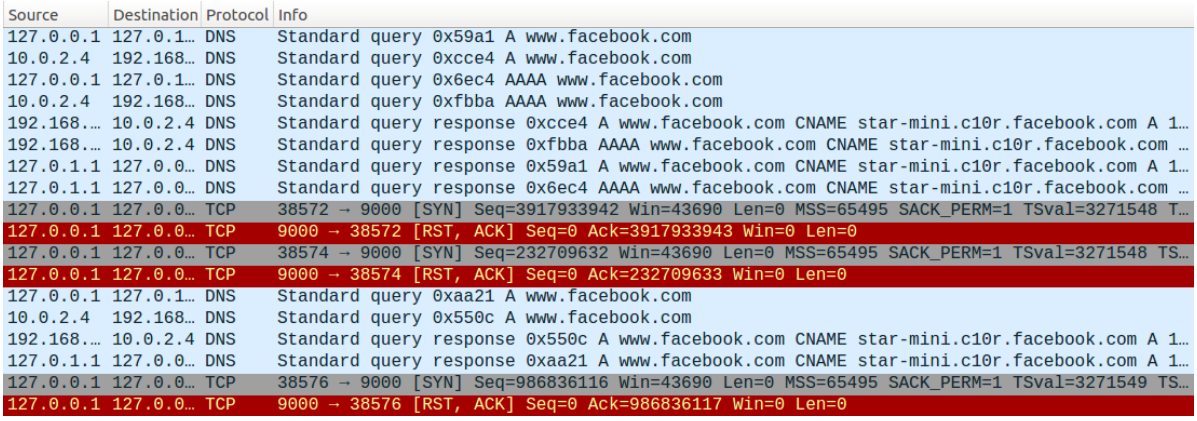
Figure 39: Facebook Page error.

3. Establish the SSH tunnel again and connect to Facebook. Please describe your observation.

Once again, you can see the Facebook page and it appears to load as per normal, just as in Figure 38.

4. Please explain what you have observed, especially on why the SSH tunnel can help bypass the egress filtering. You should use Wireshark to see what exactly is happening on the wire. Please describe your observations and explain them using the packets that you have captured.

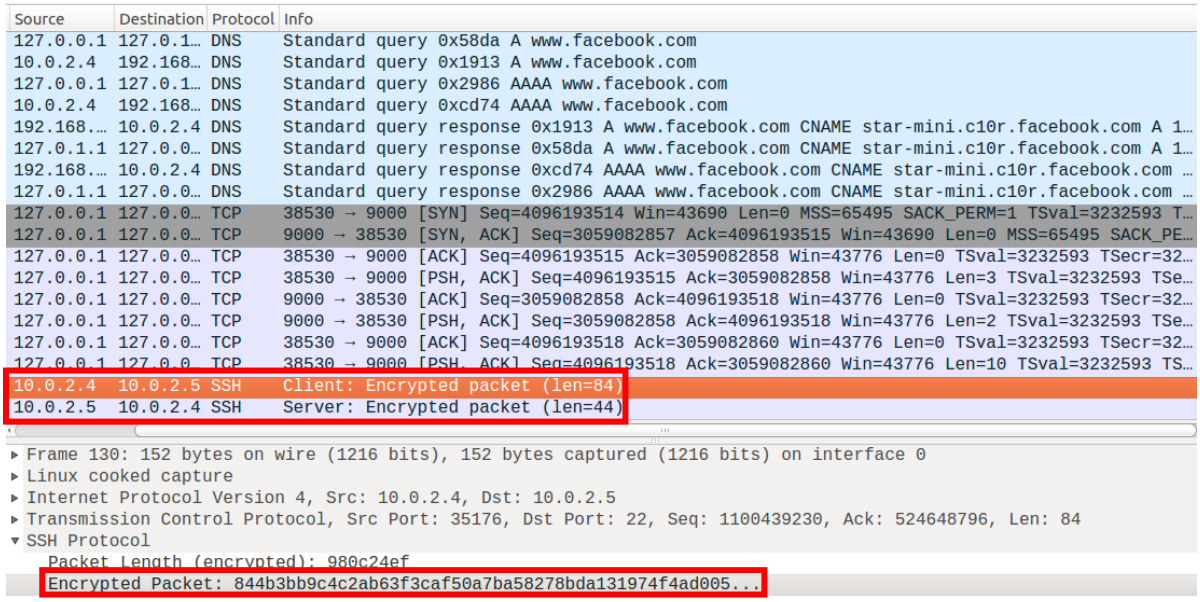
Figure 40 shows the packet capture of what occurs when we attempt to load www.facebook.com on Firefox on Machine A (`10.0.2.4`) without the SSH tunnel being set up and with the egress filtering. We see that TCP RST packets are sent continuously, which results in the page not being able to load as the three-way TCP handshake is never established.



Source	Destination	Protocol	Info
127.0.0.1	127.0.1…	DNS	Standard query 0x59a1 A www.facebook.com
10.0.2.4	192.168…	DNS	Standard query 0xcce4 A www.facebook.com
127.0.0.1	127.0.1…	DNS	Standard query 0x6ec4 AAAA www.facebook.com
10.0.2.4	192.168…	DNS	Standard query 0xfbba AAAA www.facebook.com
192.168.…	10.0.2.4	DNS	Standard query response 0xcce4 A www.facebook.com CNAME star-mini.c10r.facebook.com A 1…
192.168.…	10.0.2.4	DNS	Standard query response 0xfbba AAAA www.facebook.com CNAME star-mini.c10r.facebook.com …
127.0.1.1	127.0.0…	DNS	Standard query response 0x59a1 A www.facebook.com CNAME star-mini.c10r.facebook.com A 1…
127.0.1.1	127.0.0…	DNS	Standard query response 0x6ec4 AAAA www.facebook.com CNAME star-mini.c10r.facebook.com …
127.0.0.1	127.0.0…	TCP	38572 → 9000 [SYN] Seq=3917933942 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=3271548 T…
127.0.0.1	127.0.0…	TCP	9000 → 38572 [RST, ACK] Seq=0 Ack=3917933943 Win=0 Len=0
127.0.0.1	127.0.0…	TCP	38574 → 9000 [SYN] Seq=232709632 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=3271548 TS…
127.0.0.1	127.0.0…	TCP	9000 → 38574 [RST, ACK] Seq=0 Ack=232709633 Win=0 Len=0
127.0.0.1	127.0.1…	DNS	Standard query 0xaa21 A www.facebook.com
10.0.2.4	192.168…	DNS	Standard query 0x550c A www.facebook.com
192.168.…	10.0.2.4	DNS	Standard query response 0x550c A www.facebook.com CNAME star-mini.c10r.facebook.com A 1…
127.0.1.1	127.0.0…	DNS	Standard query response 0xaa21 A www.facebook.com CNAME star-mini.c10r.facebook.com A 1…
127.0.0.1	127.0.0…	TCP	38576 → 9000 [SYN] Seq=986836116 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=3271549 TS…
127.0.0.1	127.0.0…	TCP	9000 → 38576 [RST, ACK] Seq=0 Ack=986836117 Win=0 Len=0

Figure 40: PCAP without SSH Tunnel with Egress Filtering

However, when the SSH tunnel is setup, the egress filtering can be successfully bypassed. We can see the packets sent through SSH from Machine A (`10.0.2.4`) to Machine B (`10.0.2.5`) are encrypted as shown in Figure 41, and since there is no mention of www.facebook.com's IP address (`157.240.13.35`) and port 22 is not blocked, the packets are successfully transmitted, allowing the webpage to load successfully as the three-way TCP handshake can be successfully established.



Source	Destination	Protocol	Info
127.0.0.1	127.0.1…	DNS	Standard query 0x58da A www.facebook.com
10.0.2.4	192.168…	DNS	Standard query 0x1913 A www.facebook.com
127.0.0.1	127.0.1…	DNS	Standard query 0x2986 AAAA www.facebook.com
10.0.2.4	192.168…	DNS	Standard query 0xcd74 AAAA www.facebook.com
192.168.…	10.0.2.4	DNS	Standard query response 0x1913 A www.facebook.com CNAME star-mini.c10r.facebook.com A 1…
127.0.1.1	127.0.0…	DNS	Standard query response 0x58da A www.facebook.com CNAME star-mini.c10r.facebook.com A 1…
192.168.…	10.0.2.4	DNS	Standard query response 0xcd74 AAAA www.facebook.com CNAME star-mini.c10r.facebook.com …
127.0.1.1	127.0.0…	DNS	Standard query response 0x2986 AAAA www.facebook.com CNAME star-mini.c10r.facebook.com …
127.0.0.1	127.0.0…	TCP	38530 → 9000 [SYN] Seq=4096193514 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=3232593 T…
127.0.0.1	127.0.0…	TCP	9000 → 38530 [SYN, ACK] Seq=3059082857 Ack=4096193515 Win=43690 Len=0 MSS=65495 SACK_PE…
127.0.0.1	127.0.0…	TCP	38530 → 9000 [ACK] Seq=4096193515 Ack=3059082858 Win=43776 Len=0 TSval=3232593 TSecr=32…
127.0.0.1	127.0.0…	TCP	38530 → 9000 [PSH, ACK] Seq=4096193515 Ack=3059082858 Win=43776 Len=3 TSval=3232593 TSe…
127.0.0.1	127.0.0…	TCP	9000 → 38530 [ACK] Seq=3059082858 Ack=4096193518 Win=43776 Len=0 TSval=3232593 TSecr=32…
127.0.0.1	127.0.0…	TCP	9000 → 38530 [PSH, ACK] Seq=3059082858 Ack=4096193518 Win=43776 Len=2 TSval=3232593 TSe…
127.0.0.1	127.0.0…	TCP	38530 → 9000 [ACK] Seq=4096193518 Ack=3059082860 Win=43776 Len=0 TSval=3232593 TSecr=32…
127.0.0.1	127.0.0…	TCP	38530 → 9000 [PSH, ACK] Seq=4096193518 Ack=3059082860 Win=43776 Len=10 TSval=3232593 TS…
10.0.2.4	10.0.2.5	SSH	Client: Encrypted packet (len=84)
10.0.2.5	10.0.2.4	SSH	Server: Encrypted packet (len=44)

```
▸ Frame 130: 152 bytes on wire (1216 bits), 152 bytes captured (1216 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.5
▸ Transmission Control Protocol, Src Port: 35176, Dst Port: 22, Seq: 1100439230, Ack: 524648796, Len: 84
▾ SSH Protocol
    Packet Length (encrypted): 980c24ef
    Encrypted Packet: 844b3bb9c4c2ab63f3caf50a7ba58278bda131974f4ad005...
```

Figure 41: PCAP with SSH Tunnel with Egress Filtering