LAB 2: Linux Firewall Exploration Lab

```
[02/15/2019 15:24]Shenava(10.0.2.6)@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:5f:2e:af
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::2142:7c95:5d2d:aba6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1070 (1.0 KB)  TX bytes:7040 (7.0 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:67 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:21409 (21.4 KB)  TX bytes:21409 (21.4 KB)
```

Machine A

```
[02/15/2019 15:26]Shenava(10.0.2.5)@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:1d:3c:a2
          inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::1b16:e46:4143:36cf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:139 errors:0 dropped:0 overruns:0 frame:0
          TX packets:133 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:63061 (63.0 KB)  TX bytes:14112 (14.1 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:107 errors:0 dropped:0 overruns:0 frame:0
          TX packets:107 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:24233 (24.2 KB)  TX bytes:24233 (24.2 KB)
```

Machine B

TASK 1: Using Firewall

Prevent A from doing telnet to Machine B

We change the default input policy to accept from drop in the /etc/default/ufw config file.

```
[02/17/2019 22:10]Shenava(10.0.2.6)@VM:~$ sudo gedit /etc/default/ufw
[sudo] password for seed:
```

```
# /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="DROP"
DEFAULT_INPUT_POLICY="ACCEPT"

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

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

# Set the default application policy to ACCEPT, DROP, REJECT or SKIP. Please
# note that setting this to ACCEPT may be a security risk. See 'man ufw' for
# details
DEFAULT_APPLICATION_POLICY="SKIP"

# By default, ufw only touches its own chains. Set this to 'yes' to have ufw
# manage the built-in chains too. Warning: setting this to 'yes' will break
# non-ufw managed firewall rules
MANAGE_BUILTINS=no

#
# IPT backend
```

We will first telnet to a host 10.0.2.5 and the telnet connection can be established. But when we run the ufw packet filter and try it again, the telnet doesn't work between 10.0.2.6 and 10.0.2.5. The ufw rule is shown in the screenshot below.

```
[02/17/2019 22:14]Shenava(10.0.2.6)@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[02/17/2019 22:16]Shenava(10.0.2.6)@VM:~$ sudo ufw status
Status: active
[02/17/2019 22:16]Shenava(10.0.2.6)@VM:~$
```

```
[02/17/2019 22:16]Shenava(10.0.2.6)@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: Connection closed by foreign host.
[02/17/2019 22:17]Shenava(10.0.2.6)@VM:~$
```

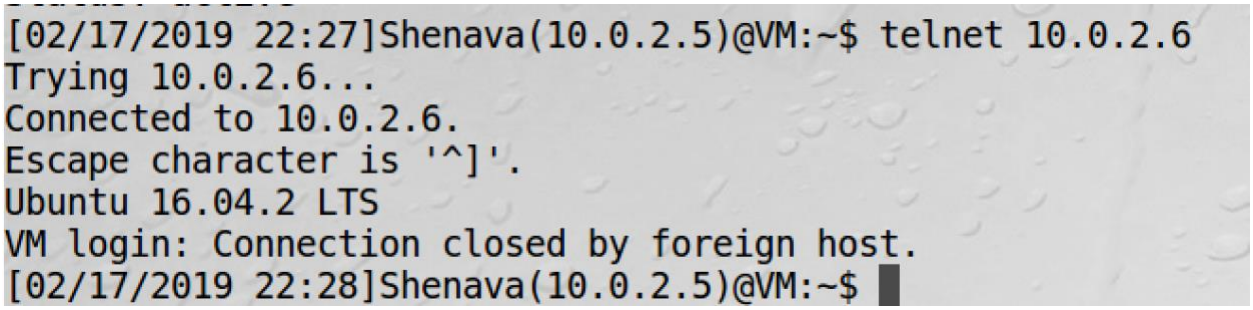
```
[02/17/2019 22:17]Shenava(10.0.2.6)@VM:~$ sudo ufw deny out from 10.0.2.6
to 10.0.2.5 port 23
Rule added
[02/17/2019 22:21]Shenava(10.0.2.6)@VM:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
10.0.2.5 23                DENY OUT    10.0.2.6

[02/17/2019 22:21]Shenava(10.0.2.6)@VM:~$
```

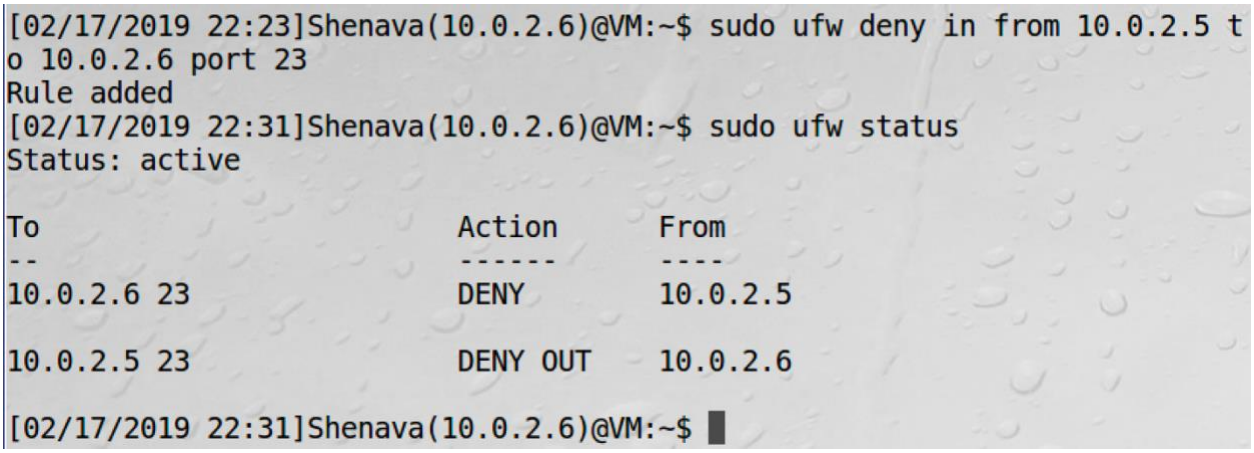
```
[02/17/2019 22:21]Shenava(10.0.2.6)@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
```

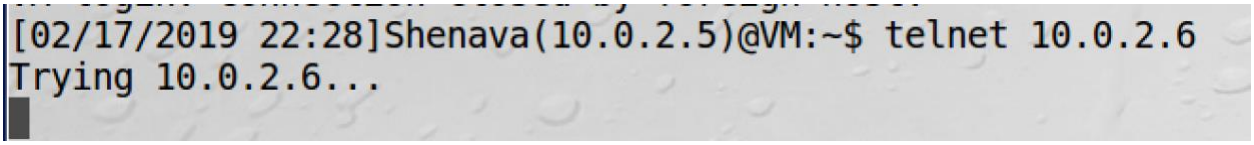
UFW is a packet level firewall that looks at each packet and decides whether to accept it or drop it. It is stateless. It looks into IP address and, port number etc. Here, we block port 23 because telnet works on that port.

Prevent B from doing telnet to Machine A

Here a machine B with 10.0.2.5 is trying to telnet to machine with 10.0.2.6. Before implementing the rule, telnet is successful. After implementing the rule, the packet filter firewall blocks all the telnet traffic from 10.0.2.5 to 10.0.2.6. The ufw rule is shown in the screenshot below.







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

We ping Syracuse.edu to gets its IP address and then write the ufw rule. After implementing the rule, packets are not sent from 10.0.2.6 to the web page Syracuse.edu.

UFW is a packet level firewall that looks at each packet and decides whether to drop it or accept it. It is stateless. It looks into the IP address and, pot number etc.

```
[02/17/2019 22:31]Shenava(10.0.2.6)@VM:~$ ping syracuse.edu -c2
PING syracuse.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=5
0 time=42.0 ms
64 bytes from syr-prod-web.syracuse.edu (128.230.18.198): icmp_seq=2 ttl=5
0 time=49.3 ms

--- syracuse.edu ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 42.009/45.696/49.383/3.687 ms
[02/17/2019 22:37]Shenava(10.0.2.6)@VM:~$
```

```
[02/17/2019 22:37]Shenava(10.0.2.6)@VM:~$ sudo ufw deny out from 10.0.2.6
to 128.230.18.198
Rule added
[02/17/2019 22:39]Shenava(10.0.2.6)@VM:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
10.0.2.6 23                DENY        10.0.2.5

10.0.2.5 23                DENY OUT    10.0.2.6
128.230.18.198             DENY OUT    10.0.2.6

[02/17/2019 22:39]Shenava(10.0.2.6)@VM:~$
```

```
[02/17/2019 22:39]Shenava(10.0.2.6)@VM:~$ ping syracuse.edu -c2
PING syracuse.edu (128.230.18.198) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted

--- syracuse.edu ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1005ms

[02/17/2019 22:40]Shenava(10.0.2.6)@VM:~$
```

```
[02/17/2019 22:40]Shenava(10.0.2.6)@VM:~$ ping 128.230.18.198 -c2
PING 128.230.18.198 (128.230.18.198) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted

--- 128.230.18.198 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1011ms

[02/17/2019 22:41]Shenava(10.0.2.6)@VM:~$
```

TASK 2: Implementing a Simple Firewall

Prevent A from doing telnet to B

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/inet.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>

static struct nf_hook_ops telnetFilterHook;

unsigned int telnetFilter(void *priv, struct sk_buff *skb, const
struct nf_hook_state *state)
{
        struct iphdr *iph;
        struct tcphdr *tcph;
        iph = ip_hdr(skb);
        tcph = (void *)iph+iph->ihl*4;
        if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23)
&& iph->saddr == in_aton("10.0.2.6") && iph->daddr == 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;
        }
}

int setUpFilter(void)
{
        printk(KERN_INFO "Registering a Telnet filter.\n");
        telnetFilterHook.hook = telnetFilter;
        telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.pf = PF_INET;
        telnetFilterHook.priority = NF_IP_PRI_FIRST;
        nf_register_net_hook(&init_net,&telnetFilterHook);
        return 0;
}

void removeFilter(void)
{
         printk(KERN_INFO "Telnet filter is being removed.\n");
         nf_unregister_net_hook(&init_net,&telnetFilterHook);
}

module_init(setUpFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

The code for Loadable Kernel Module (LKM) using Netfilter. Here the netfilter hook is NF_INET_POST_ROUTING. Here we are preventing machine A with IP 10.0.2.6 from doing a telnet to 10.0.2.8 address. We block port 23 since telnet works on that port.

Makefile code:

```
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
```

We perform make to compile the LKM, then we check if telnet to 10.0.2.5 address works. It will work since kernel module isn't inserted yet. We then insert the kernel module and check again. Now the telnet will not work. We then check the kernel log to see the proof that packets are dropped.

```
[02/17/2019 23:15]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit task2.c
[02/17/2019 23:20]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit Makefile
[02/17/2019 23:21]Shenava(10.0.2.6)@VM:~/.../lab3$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/Desktop/labs/lab3
 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/Desktop/labs/lab3/task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/Desktop/labs/lab3/task2.mod.o
  LD [M]  /home/seed/Desktop/labs/lab3/task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
```

```
[02/17/2019 23:24]Shenava(10.0.2.6)@VM:~/.../lab3$ 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: Connection closed by foreign host.
[02/17/2019 23:24]Shenava(10.0.2.6)@VM:~/.../lab3$ sudo insmod task2.ko
[02/17/2019 23:25]Shenava(10.0.2.6)@VM:~/.../lab3$ telnet 10.0.2.5
Trying 10.0.2.5...
^C
[02/17/2019 23:26]Shenava(10.0.2.6)@VM:~/.../lab3$ dmesg | tail -10
[ 3702.297463] Bluetooth: L2CAP socket layer initialized
[ 3702.297468] Bluetooth: SCO socket layer initialized
[ 3702.324316] Netfilter messages via NETLINK v0.30.
[ 3977.566460] ip_tables: (C) 2000-2006 Netfilter Core Team
[ 3977.587279] nf_conntrack version 0.5.0 (16384 buckets, 65536 max)
[ 3977.624328] ip6_tables: (C) 2000-2006 Netfilter Core Team
[ 8154.637113] Registering a Telnet filter.
[ 8168.257582] Dropping telnet packet to 10.0.2.5
[ 8169.265497] Dropping telnet packet to 10.0.2.5
[ 8171.280943] Dropping telnet packet to 10.0.2.5
```

```
[02/17/2019 23:28]Shenava(10.0.2.6)@VM:~/.../lab3$ modinfo task2.ko
filename:       /home/seed/Desktop/labs/lab3/task2.ko
license:        GPL
srcversion:     1BAA2C94670B3FE14DBB01F
depends:
vermagic:       4.8.0-36-generic SMP mod_unload modversions 686
[02/17/2019 23:29]Shenava(10.0.2.6)@VM:~/.../lab3$
```

Prevent B from doing telnet to Machine A

The code for Loadable Kernel Module using Netfilter. Here the netfilter hook is NF_INET_PRE_ROUTING. Here we are preventing machine B with 10.0.2.5 address from doing a telnet to 10.0.2.6. We block port 23 since telnet works on that port.

```c
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/inet.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>

static struct nf_hook_ops telnetFilterHook;

unsigned int telnetFilter(void *priv, struct sk_buff *skb, const struct nf_hook_state
*state)
{
        struct iphdr *iph;
        struct tcphdr *tcph;
        iph = ip_hdr(skb);
        tcph = (void *)iph+iph->ihl*4;
        if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23) && iph->saddr ==
in_aton("10.0.2.5") && iph->daddr == in_aton("10.0.2.6"))
        {
                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;
        }
}

int setUpFilter(void)
{
        printk(KERN_INFO "Registering a Telnet filter.\n");
        telnetFilterHook.hook = telnetFilter;
        telnetFilterHook.hooknum = NF_INET_PRE_ROUTING;
        telnetFilterHook.pf = PF_INET;
        telnetFilterHook.priority = NF_IP_PRI_FIRST;
        nf_register_net_hook(&init_net,&telnetFilterHook);
        return 0;
}

void removeFilter(void)
{
        printk(KERN_INFO "Telnet filter is being removed.\n");
        nf_unregister_net_hook(&init_net,&telnetFilterHook);
}

module_init(setUpFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

The Makefile code:

```
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
```

We perform make to compile the LKM, then we check if telnet to 10.0.2.6 works from 10.0.2.5. It will work since kernel module isn't inserted yet. We then insert the kernel module and check again. Now the telnet will not work. We then check the kernel log to see the proof that packets are dropped.

```
[02/18/2019 00:00]Shenava(10.0.2.5)@VM:~$ 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: ^CConnection closed by foreign host.
[02/18/2019 00:06]Shenava(10.0.2.5)@VM:~$ telnet 10.0.2.6
Trying 10.0.2.6...
```

```
[02/18/2019 00:00]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit task2.c
[02/18/2019 00:06]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit Makefile
[02/18/2019 00:06]Shenava(10.0.2.6)@VM:~/.../lab3$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/Desktop/labs/lab3
 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/Desktop/labs/lab3/task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/Desktop/labs/lab3/task2.mod.o
  LD [M]  /home/seed/Desktop/labs/lab3/task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[02/18/2019 00:06]Shenava(10.0.2.6)@VM:~/.../lab3$ sudo rmmod task2.ko
[02/18/2019 00:06]Shenava(10.0.2.6)@VM:~/.../lab3$ sudo insmod task2.ko
[02/18/2019 00:07]Shenava(10.0.2.6)@VM:~/.../lab3$ dmesg | tail -10
[ 9824.592046] Registering a Telnet filter.
[ 9986.913209] Telnet filter is being removed.
[10223.591758] Registering a Telnet filter.
[10629.000935] Telnet filter is being removed.
[10633.143356] Registering a Telnet filter.
[10637.664459] Dropping telnet packet to 10.0.2.6
[10638.671838] Dropping telnet packet to 10.0.2.6
[10640.688073] Dropping telnet packet to 10.0.2.6
[10644.812986] Dropping telnet packet to 10.0.2.6
[10653.002005] Dropping telnet packet to 10.0.2.6
[02/18/2019 00:07]Shenava(10.0.2.6)@VM:~/.../lab3$ modinfo task2.ko
filename:       /home/seed/Desktop/labs/lab3/task2.ko
license:        GPL
srcversion:     088CD9CF70302433278488C
depends:
vermagic:       4.8.0-36-generic SMP mod_unload modversions 686
[02/18/2019 00:07]Shenava(10.0.2.6)@VM:~/.../lab3$
```

Here we are implementing a mini firewall and that requires us to load our code into the kernel using the hooks provided by netfilter. We place our code at one of the hooks and do the packet filtering.

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

We ping google.com to find the IP address.

```
[02/18/2019 00:15]Shenava(10.0.2.6)@VM:~/.../lab3$ ping google.com -c2
PING google.com (172.217.10.46) 56(84) bytes of data.
64 bytes from lga34s13-in-f14.1e100.net (172.217.10.46): icmp_seq=1 ttl=54 time=
42.6 ms
64 bytes from lga34s13-in-f14.1e100.net (172.217.10.46): icmp_seq=2 ttl=54 time=
44.3 ms

--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 42.651/43.513/44.375/0.862 ms
[02/18/2019 00:16]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit task2.c
```

The code for Loadable Kernel Module using Netfilter. Here the netfilter hook is NF_INET_POST_ROUTING. Here we are preventing machine A with IP 10.0.2.6 from doing a http request to google.com.

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/inet.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>

static struct nf_hook_ops telnetFilterHook;

unsigned int telnetFilter(void *priv, struct sk_buff *skb, const struct nf_hook_state *state)
{
        struct iphdr *iph;
        struct tcphdr *tcph;
        iph = ip_hdr(skb);
        tcph = (void *)iph+iph->ihl*4;
        if(iph->saddr == in_aton("10.0.2.6") && iph->daddr == in_aton("172.217.10.46"))
        {
                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;
        }
}

int setUpFilter(void)
{
        printk(KERN_INFO "Registering a Telnet filter.\n");
        telnetFilterHook.hook = telnetFilter;
        telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.pf = PF_INET;
        telnetFilterHook.priority = NF_IP_PRI_FIRST;
        nf_register_net_hook(&init_net,&telnetFilterHook);
        return 0;
}
void removeFilter(void)
{
        printk(KERN_INFO "Telnet filter is being removed.\n");
        nf_unregister_net_hook(&init_net,&telnetFilterHook);
}

module_init(setUpFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

The Makefile code:

```
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
```

We perform make to compile the LKM. We then insert the kernel module and check again. Now the request to google.com will not work. We then check the kernel log to see the proof that packets are dropped.

```
[02/18/2019 00:16]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit task2.c
[02/18/2019 00:21]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit Makefile
[02/18/2019 00:22]Shenava(10.0.2.6)@VM:~/.../lab3$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/Desktop/labs/lab3 modul
es
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/Desktop/labs/lab3/task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/Desktop/labs/lab3/task2.mod.o
  LD [M]  /home/seed/Desktop/labs/lab3/task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[02/18/2019 00:22]Shenava(10.0.2.6)@VM:~/.../lab3$ sudo rmmod task2.ko
[sudo] password for seed:
[02/18/2019 00:23]Shenava(10.0.2.6)@VM:~/.../lab3$ sudo insmod task2.ko
[02/18/2019 00:23]Shenava(10.0.2.6)@VM:~/.../lab3$ ping google.com -c2
PING google.com (172.217.10.14) 56(84) bytes of data.

--- google.com ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1020ms
```

Here we are implementing a mini firewall and that requires us to load our code into the kernel using the hooks provided by netfilter. We place our code at one of the hooks and do the packet filtering.

Prevent A from receiving an ICMP request from machine B

The code for Loadable Kernel Module using Netfilter. Here the netfilter hook is NF_INET_PRE_ROUTING. Here we are preventing machine A with IP 10.0.2.6 from receiving an ICMP request from 10.0.2.5.

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/inet.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>

static struct nf_hook_ops telnetFilterHook;

unsigned int telnetFilter(void *priv, struct sk_buff *skb, const struct nf_hook_state
*state)
{
        struct iphdr *iph;
        struct tcphdr *tcph;
        iph = ip_hdr(skb);
        tcph = (void *)iph+iph->ihl*4;
        if(iph->protocol == IPPROTO_ICMP && iph->saddr == in_aton("10.0.2.5") && iph-
>daddr == in_aton("10.0.2.6"))
        {
                printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d.%d.%d.%d.%d\n",
                ((unsigned char *)&iph->saddr)[0],
                ((unsigned char *)&iph->saddr)[1],
                ((unsigned char *)&iph->saddr)[2],
                ((unsigned char *)&iph->saddr)[3],
                ((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;
        }
}

int setUpFilter(void)
{
        printk(KERN_INFO "Registering a Telnet filter.\n");
        telnetFilterHook.hook = telnetFilter;
        telnetFilterHook.hooknum = NF_INET_PRE_ROUTING;
        telnetFilterHook.pf = PF_INET;
        telnetFilterHook.priority = NF_IP_PRI_FIRST;
        nf_register_net_hook(&init_net,&telnetFilterHook);
        return 0;
}

void removeFilter(void)
{
        printk(KERN_INFO "Telnet filter is being removed.\n");
        nf_unregister_net_hook(&init_net,&telnetFilterHook);
}

module_init(setUpFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

The Makefile code:

```
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
```

We perform make to compile the LKM, then we check if ping to 10.0.2.6 works from 10.0.2.5. It will work since kernel module isn't inserted yet. We then insert the kernel module and check again. Now the ping will not work. We then check the kernel log to see the proof that packets are dropped.

```
[02/18/2019 01:03]Shenava(10.0.2.5)@VM:~$ ping 10.0.2.6 -c2
PING 10.0.2.6 (10.0.2.6) 56(84) bytes of data.
64 bytes from 10.0.2.6: icmp_seq=1 ttl=64 time=0.431 ms
64 bytes from 10.0.2.6: icmp_seq=2 ttl=64 time=0.698 ms

--- 10.0.2.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.431/0.564/0.698/0.135 ms
[02/18/2019 01:17]Shenava(10.0.2.5)@VM:~$ ping 10.0.2.6
PING 10.0.2.6 (10.0.2.6) 56(84) bytes of data.
```

```
[02/18/2019 01:09]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit task2.c
[02/18/2019 01:19]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit Makefile
[02/18/2019 01:19]Shenava(10.0.2.6)@VM:~/.../lab3$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/Desktop/labs/lab3 mo
dules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/Desktop/labs/lab3/task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/Desktop/labs/lab3/task2.mod.o
  LD [M]  /home/seed/Desktop/labs/lab3/task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[02/18/2019 01:19]Shenava(10.0.2.6)@VM:~/.../lab3$ sudo insmod task2.ko
[02/18/2019 01:19]Shenava(10.0.2.6)@VM:~/.../lab3$ dmesg | tail -10
[ 1872.905972] Dropping telnet packet to 10.0.2.5.10.0.2.6
[ 1873.930403] Dropping telnet packet to 10.0.2.5.10.0.2.6
[ 1874.953137] Dropping telnet packet to 10.0.2.5.10.0.2.6
[ 1875.976306] Dropping telnet packet to 10.0.2.5.10.0.2.6
[ 1876.999777] Dropping telnet packet to 10.0.2.5.10.0.2.6
[ 1878.023177] Dropping telnet packet to 10.0.2.5.10.0.2.6
[ 1879.047210] Dropping telnet packet to 10.0.2.5.10.0.2.6
[ 1880.070566] Dropping telnet packet to 10.0.2.5.10.0.2.6
[ 1881.093338] Dropping telnet packet to 10.0.2.5.10.0.2.6
[ 1882.118397] Dropping telnet packet to 10.0.2.5.10.0.2.6
[02/18/2019 01:20]Shenava(10.0.2.6)@VM:~/.../lab3$
```

Prevent A from sending an ICMP request to any machine.

The code for Loadable Kernel Module using Netfilter. Here the netfilter hook is NF_INET_POST_ROUTING. Here we are preventing machine A with IP 10.0.2.6 from sending a ICMP request out.

```c
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/inet.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>

static struct nf_hook_ops telnetFilterHook;

unsigned int telnetFilter(void *priv, struct sk_buff *skb, const struct nf_hook_state
*state)
{
        struct iphdr *iph;
        struct tcphdr *tcph;
        iph = ip_hdr(skb);
        tcph = (void *)iph+iph->ihl*4;
        if(iph->protocol == IPPROTO_ICMP && iph->saddr == in_aton("10.0.2.6"))
        {
                printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",
                ((unsigned char *)&iph->saddr)[0],
                ((unsigned char *)&iph->saddr)[1],
                ((unsigned char *)&iph->saddr)[2],
                ((unsigned char *)&iph->saddr)[3]
                ((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;
        }
}

int setUpFilter(void)
{
        printk(KERN_INFO "Registering a Telnet filter.\n");
        telnetFilterHook.hook = telnetFilter;
        telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.pf = PF_INET;
        telnetFilterHook.priority = NF_IP_PRI_FIRST;
        nf_register_net_hook(&init_net,&telnetFilterHook);
        return 0;
}

void removeFilter(void)
{
        printk(KERN_INFO "Telnet filter is being removed.\n");
        nf_unregister_net_hook(&init_net,&telnetFilterHook);
}

module_init(setUpFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

The Makefile code:

```
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
```

We perform make to compile the LKM, then we check if ping to 10.0.2.5 works from 10.0.2.6. It will work since kernel module isn't inserted yet. We then insert the kernel module and check again. Now the ping will not work. We then check the kernel log to see the proof that packets are dropped.

```
[02/18/2019 01:27]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit task2.c
[02/18/2019 01:27]Shenava(10.0.2.6)@VM:~/.../lab3$ gedit Makefile
[02/18/2019 01:28]Shenava(10.0.2.6)@VM:~/.../lab3$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/Desktop/labs/lab3 mo
dules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/Desktop/labs/lab3/task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/Desktop/labs/lab3/task2.mod.o
  LD [M]  /home/seed/Desktop/labs/lab3/task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[02/18/2019 01:29]Shenava(10.0.2.6)@VM:~/.../lab3$ ping 10.0.2.5 -c2
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=0.524 ms
64 bytes from 10.0.2.5: icmp_seq=2 ttl=64 time=0.524 ms

--- 10.0.2.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1012ms
rtt min/avg/max/mdev = 0.524/0.524/0.524/0.000 ms
```

```
[02/18/2019 01:30]Shenava(10.0.2.6)@VM:~/.../lab3$ sudo rmmod task2.ko
[02/18/2019 01:30]Shenava(10.0.2.6)@VM:~/.../lab3$ sudo insmod task2.ko
[02/18/2019 01:30]Shenava(10.0.2.6)@VM:~/.../lab3$ 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
^C
--- 10.0.2.5 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2046ms

[02/18/2019 01:30]Shenava(10.0.2.6)@VM:~/.../lab3$ sudo rmmod task2.ko
[02/18/2019 01:31]Shenava(10.0.2.6)@VM:~/.../lab3$ dmesg | tail -10
[ 2496.211355] Dropping telnet packet to 10.0.2.6.10.0.2.5
[ 2497.234120] Dropping telnet packet to 10.0.2.6.10.0.2.5
[ 2498.258061] Dropping telnet packet to 10.0.2.6.10.0.2.5
[ 2499.281882] Dropping telnet packet to 10.0.2.6.10.0.2.5
[ 2500.304274] Dropping telnet packet to 10.0.2.6.10.0.2.5
[ 2501.327784] Dropping telnet packet to 10.0.2.6.10.0.2.5
[ 2502.351164] Dropping telnet packet to 10.0.2.6.10.0.2.5
[ 2503.375133] Dropping telnet packet to 10.0.2.6.10.0.2.5
[ 2504.398724] Dropping telnet packet to 10.0.2.6.10.0.2.5
[ 2504.870061] Telnet filter is being removed.
[02/18/2019 01:31]Shenava(10.0.2.6)@VM:~/.../lab3$
```

TASK 3: Evading Egress Filtering

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

```
[02/18/2019 18:11]Shenava(10.0.2.6)@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: Connection closed by foreign host.
[02/18/2019 18:11]Shenava(10.0.2.6)@VM:~$
```

We telnet to 10.0.2.5 from Machine A.

Now we are setting ufw rule to prevent 10.0.2.6 from doing a telnet to any machine out.

```
[02/18/2019 18:11]Shenava(10.0.2.6)@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[02/18/2019 18:11]Shenava(10.0.2.6)@VM:~$ sudo ufw status
Status: active
[02/18/2019 18:11]Shenava(10.0.2.6)@VM:~$ sudo ufw deny out from 10.0.2.6 to
any port 23
Rule added
[02/18/2019 18:12]Shenava(10.0.2.6)@VM:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
23                         DENY OUT    10.0.2.6

[02/18/2019 18:12]Shenava(10.0.2.6)@VM:~$
```

```
[02/18/2019 18:12]Shenava(10.0.2.6)@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
```

After implementing the ufw rule we try to ping 10.0.2.5 from Machine A and we see that ping not working.

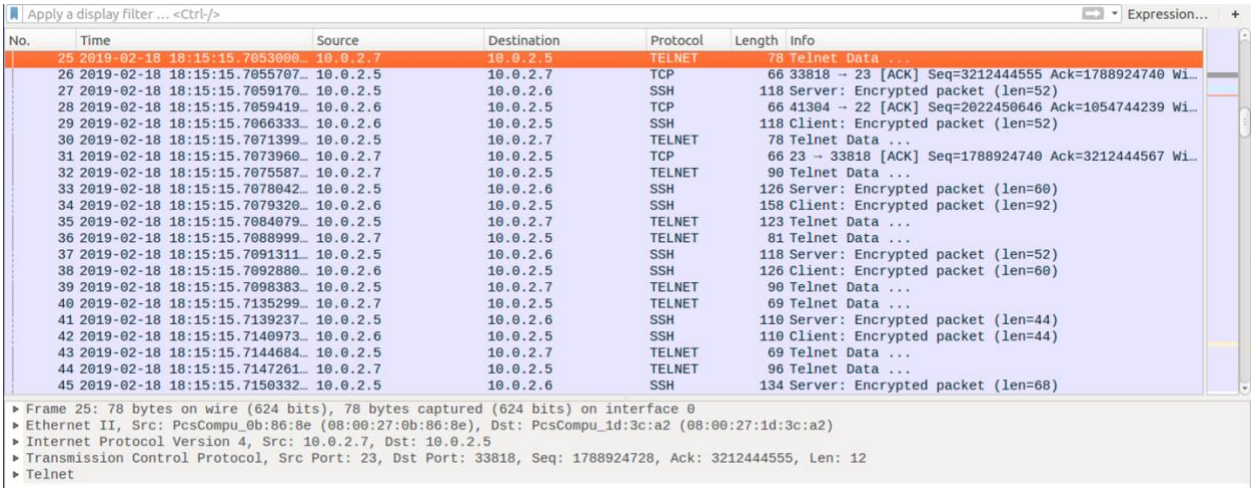
We are implementing a SSH connection to 10.0.2.7 from 10.0.2.6.

```
[02/18/2019 18:13]Shenava(10.0.2.6)@VM:~$ ssh -L 8000:10.0.2.7:23 seed@10.0.2
.5
The authenticity of host '10.0.2.5 (10.0.2.5)' can't be established.
ECDSA key fingerprint is SHA256:p1zAio6c1bI+8HDp5xa+eKRi561aFDaPE1/xq1eYzCI.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added '10.0.2.5' (ECDSA) to the list of known hosts.
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

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

Last login: Sun Feb 10 20:04:12 2019 from 10.0.2.7
[02/18/2019 18:15]Shenava(10.0.2.5)@VM:~$
```

We create a SSH tunnel connection as shown above and on another terminal we perform telnet to localhost 8000

```
[02/18/2019 18:15]Shenava(10.0.2.6)@VM:~$ telnet localhost 8000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sun Feb 10 21:40:10 EST 2019 from 10.0.2.5 on pts/4
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

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

[02/18/19]Shenava(10.0.2.7)@VM:~$
```

After implementing the SSH connection, we perform a telnet to localhost 8000. We can see that telnet is successful to 10.0.2.5 though we have ufw enabled that blocks direct telnet to any machine.



From the Wireshark capture, we see that SSH connection is established between 10.0.2.6 and 10.0.2.7 and a telnet connection from 10.0.2.7 and 10.0.2.5. So, telnet packets from 10.0.2.6 to 10.0.2.5 goes using the SSH tunnel evading the ufw firewall.

UFW blocks all telnet connection going out from machine 10.0.2.6. Hence, to evade the firewall, we create a SSH tunnel between 10.0.2.6 and 10.0.2.7 and pass the telnet packets to 10.0.2.5 from 10.0.2.6 using this tunnel. Packet filters usually don't look at content inside the packets, they only look at packet level data like IP address or port numbers.

Task 3.b: Connecting to Facebook using SSH Tunnel.

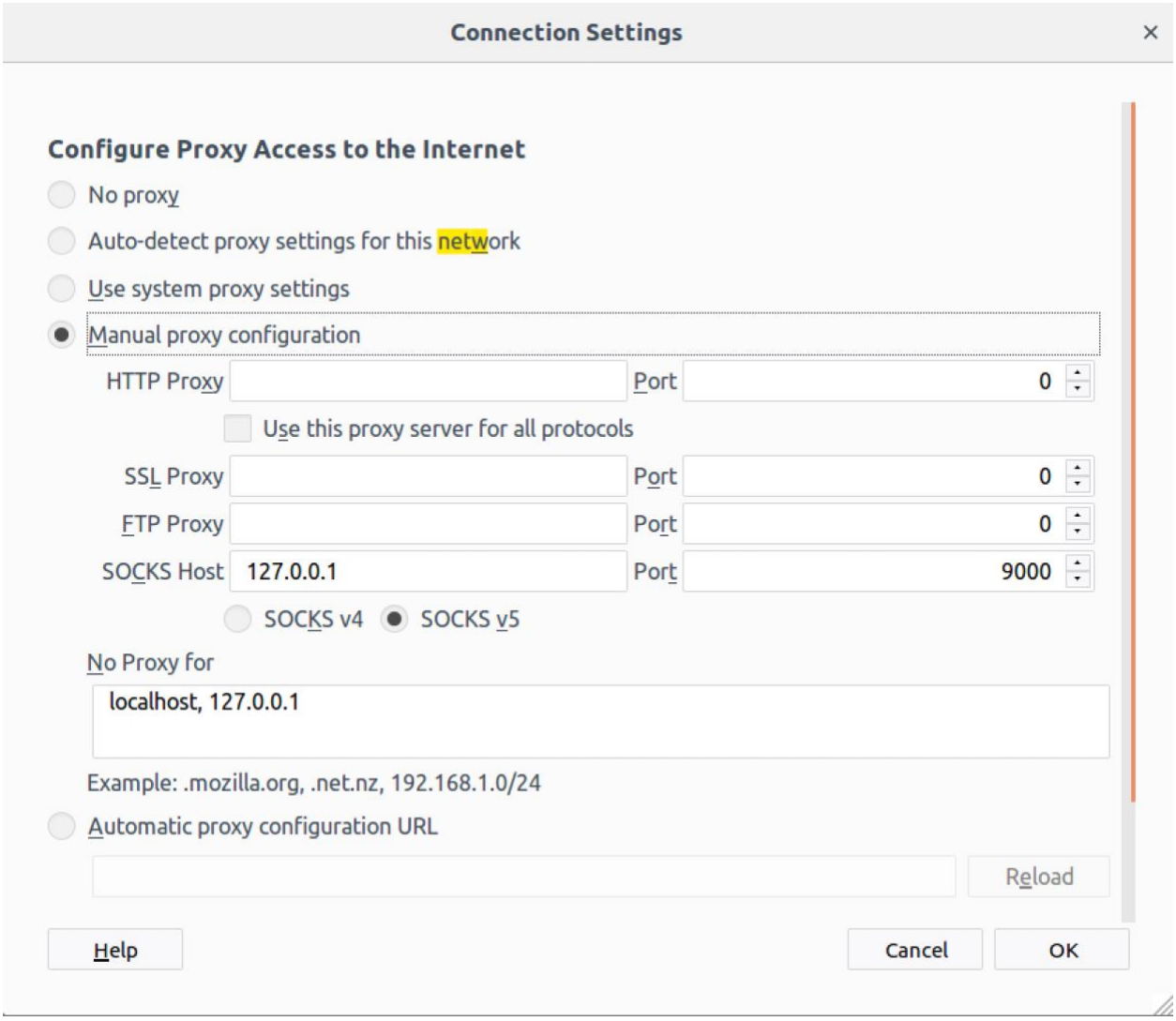
```
[02/18/2019 18:46]Shenava(10.0.2.6)@VM:~$ ping facebook.com -c1
PING facebook.com (157.240.2.35) 56(84) bytes of data.
64 bytes from edge-star-mini-shv-01-ort2.facebook.com (157.240.2.35): icmp_seq=1
 ttl=53 time=34.8 ms

--- facebook.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 34.832/34.832/34.832/0.000 ms
[02/18/2019 18:46]Shenava(10.0.2.6)@VM:~$ sudo ufw deny out to 157.240.2.35
Rule added
[02/18/2019 18:47]Shenava(10.0.2.6)@VM:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
23                         DENY OUT    10.0.2.6
157.240.2.35               DENY OUT    Anywhere

[02/18/2019 18:47]Shenava(10.0.2.6)@VM:~$
```

We ping Facebook.com page to obtain its IP address. Then we setup a ufw rule.

```
/bin/bash 80x24
[02/18/2019 18:52]Shenava(10.0.2.6)@VM:~$ sudo ufw status
[sudo] password for seed:
Status: active

To                         Action      From
--                         ------      ----
23                         DENY OUT    10.0.2.6
157.240.2.35               DENY OUT    Anywhere

[02/18/2019 18:52]Shenava(10.0.2.6)@VM:~$ ssh -D 9000 -C seed@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

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

Last login: Mon Feb 18 18:36:15 2019 from 10.0.2.6
[02/18/2019 18:52]Shenava(10.0.2.5)@VM:~$
```

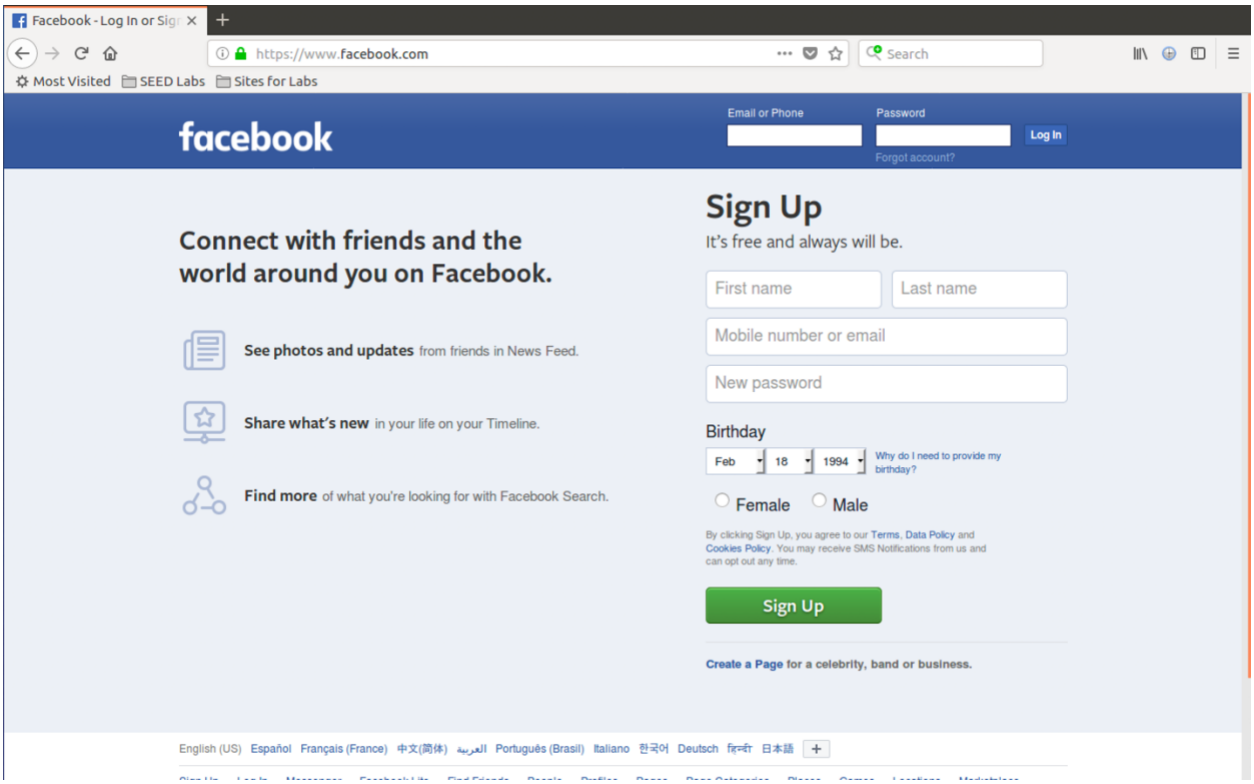
We establish a SSH connection between localhost:9000 and Machine B

We then go to the network setting and change the connection settings to manual and do the below procedure as show on screenshot in which we do dynamic port forwarding.



After the setup is done, please do the followings:

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



When we set the SSH tunnel, the page is loaded.

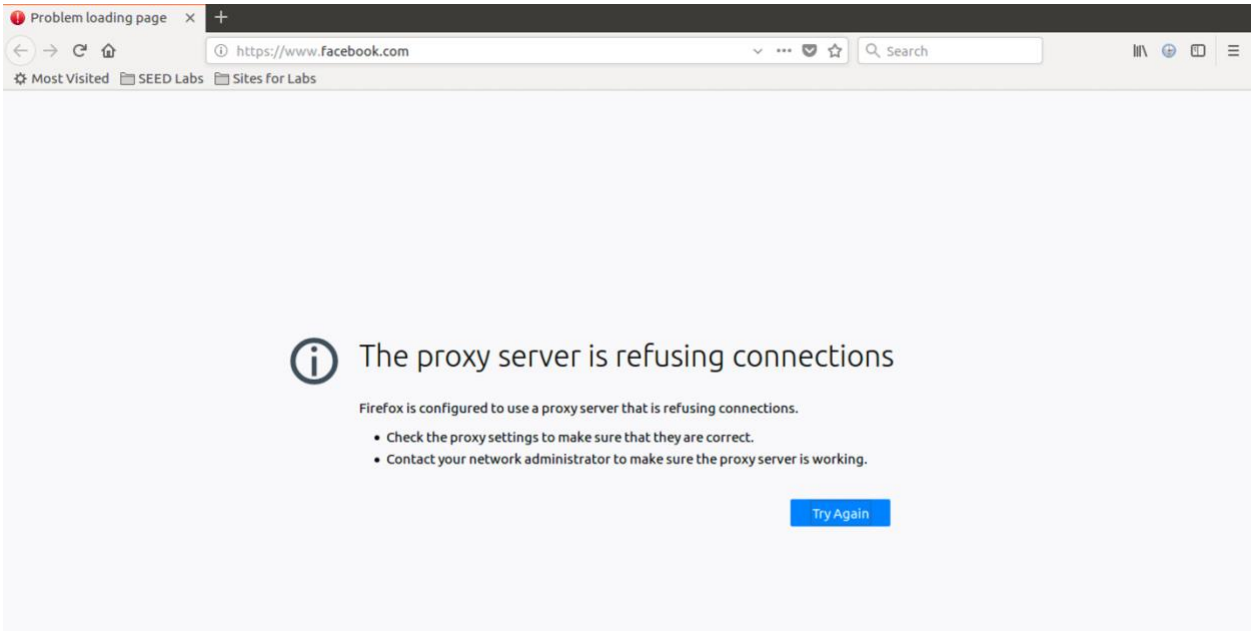
UFW blocks all http connection going out from machine 10.0.2.6 to Facebook.com. So, to evade the firewall, we create a SSH tunnel between 10.0.2.6 and the web server and pass the http packets to 10.0.2.5 from 10.0.2.6 using this tunnel. Packet filters usually don't look at content inside the packets, they only look at packet level data like IP address or port numbers. Hence, we can access Facebook.com

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

We break the SSH connection by entering exit as shown below.

```
[02/18/2019 18:52]Shenava(10.0.2.5)@VM:~$ exit
logout
Connection to 10.0.2.5 closed.
```

Then we will reload the Facebook page and we notice that we are not able to access the page and page does not load.



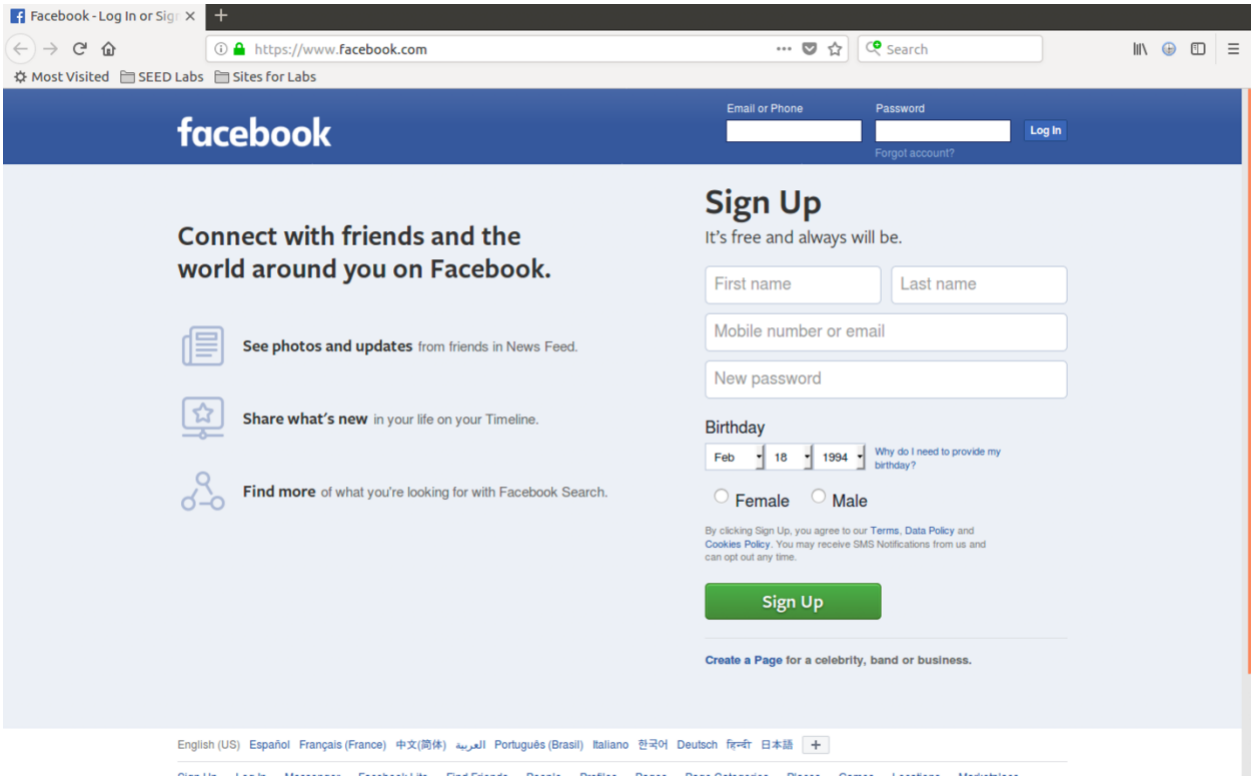
When the SSH tunnel is broken, there is no path for the http packets to go out since the packet filter blocks all the requests to Facebook.com.

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

When the connection is re-established, there is a path for the HTTP packets to go out through the SSH tunnel, so the page is loaded.

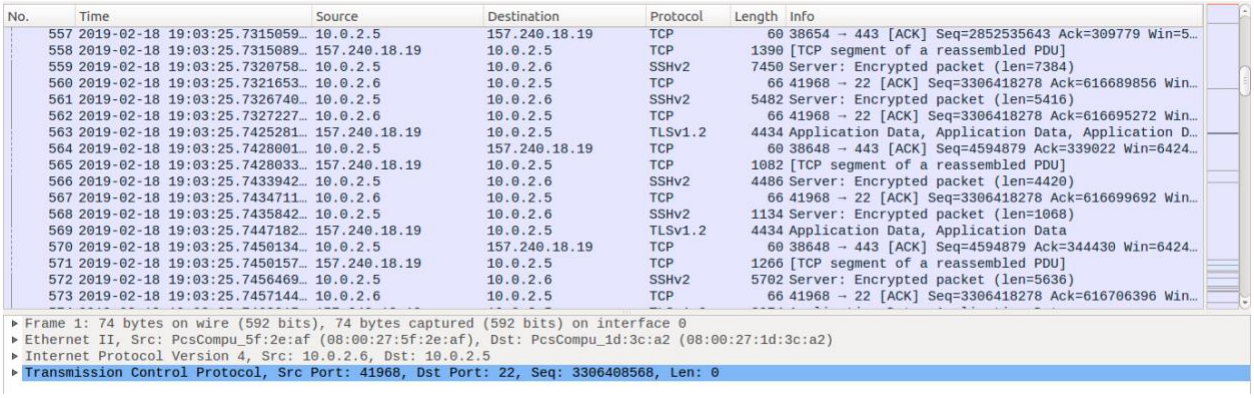
```
[02/18/2019 18:55]Shenava(10.0.2.6)@VM:~$ ssh -D 9000 -C seed@10.0.2.5
seed@10.0.2.5's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

Last login: Mon Feb 18 18:52:25 2019 from 10.0.2.6
[02/18/2019 18:58]Shenava(10.0.2.5)@VM:~$
```

We clear the cache of the web browser and reload the page and see the page is successfully loading.



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.

As seen in the Wireshark capture, packets from 10.0.2.6 goes to web server using the SSH tunnel and then the packets go to 157.240.18.19 which is Facebook.com as there is no packet filter there. The response also comes back in the reverse direction following the same path.



UFW blocks all http connection going out from machine 10.0.2.6 to Facebook.com. So, to evade the egress firewall, we create a SSH tunnel between 10.0.2.6 and web server and pass the http packets to 10.0.2.5 from 10.0.2.6 using this tunnel. Packet filters usually don't look at content inside the packets, they only look at packet level data like ip address or port numbers.

TASK 3: Evading Ingress Filtering

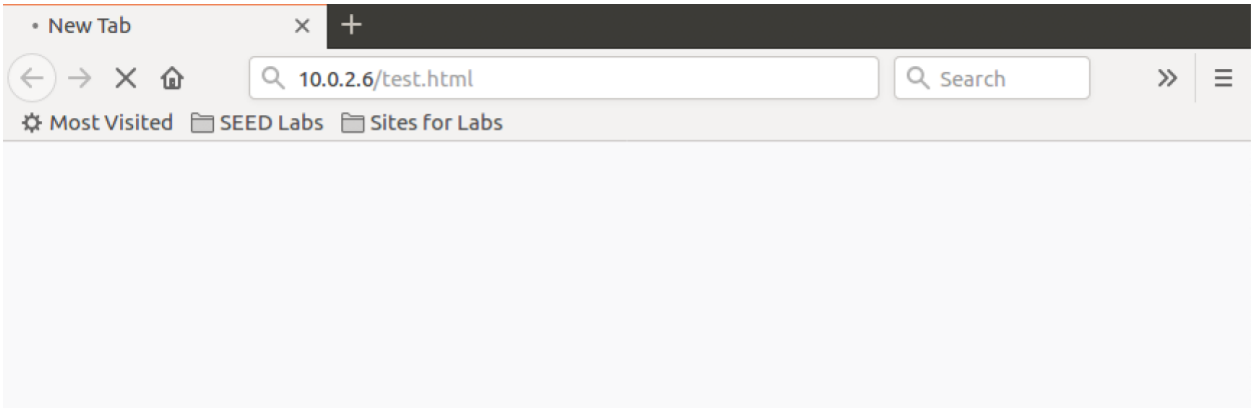
First, I created an html page (internal website) on the company machine which is Machine A.

```
<html>
<body>
Internal website: Company
</body>
</html>
```

Then we look for the created internal website in home machine which is Machine B which successfully opens the website.

We set up ufw rules to deny in from any source to Machine A from Port 80 and Port 23 (i.e, here we are trying to block Machine B from accessing its port 80 and 22) as shown below.

```
[02/18/2019 19:31]Shenava(10.0.2.6)@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[02/18/2019 19:31]Shenava(10.0.2.6)@VM:~$ sudo ufw status
Status: active
[02/18/2019 19:31]Shenava(10.0.2.6)@VM:~$ sudo ufw deny in from any to 10.
0.2.6 port 80
Rule added
[02/18/2019 19:31]Shenava(10.0.2.6)@VM:~$ sudo ufw deny in from any to 10.
0.2.6 port 22
Rule added
[02/18/2019 19:31]Shenava(10.0.2.6)@VM:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
10.0.2.6 80                DENY        Anywhere
10.0.2.6 22                DENY        Anywhere

[02/18/2019 19:31]Shenava(10.0.2.6)@VM:~$
```



When we again try to access the internal website from the home machine it is not able to open due to the ufw rule setup.

```
[02/18/2019 19:51]Shenava(10.0.2.6)@VM:~$ ssh -R 9000:10.0.2.6:80 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

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

Last login: Mon Feb 18 19:51:18 2019 from 10.0.2.6
[02/18/2019 19:52]Shenava(10.0.2.5)@VM:~$
```

Then we set up a reverse SSH tunnel on Machine A such that once we get home, we can still access the protected web server on A from home (Machine B).

When we run the webpage through localhost:9000 as done in task 3 we achieve the page from home again.

Below is a screenshot of the Wireshark packet capture to see how it is done and we see port 22 is used.