

Machine A: 10.0.2.7

Machine B: 10.0.2.8

Task1: Using Firewall

- Prevent A from doing telnet to Machine B.

Code & result:

```
[03/04/21]seed@VM:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target    prot opt source          destination
Chain FORWARD (policy ACCEPT)
target    prot opt source          destination
Chain OUTPUT (policy ACCEPT)
target    prot opt source          destination
[03/04/21]seed@VM:~$ man iptables
[03/04/21]seed@VM:~$ sudo iptables -A OUTPUT -d 10.0.2.8 -p tcp
--dport 23 -j DROP
[03/04/21]seed@VM:~$ telnet 10.0.2.8
Trying 10.0.2.8...
telnet: Unable to connect to remote host: Connection timed out
[03/04/21]seed@VM:~$
```

As you can see, I use the command ‘sudo iptables -A OUTPUT -d 10.0.2.8 -p tcp –dport 23 -j DROP’ on machine A to drop all the packets which send from machine A to 10.0.2.8 which is machine B with destination port of 23. Which means I drop all the telnet packet which send from A to B. After I added this new iptables rule, the telnet command will not work.

- Prevent B from doing telnet to machine A

Code&result:

```
[03/04/21]seed@VM:~$ sudo iptables -A INPUT -s 10.0.2.8 -p tcp
--dport 23 -j DROP
[03/04/21]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
telnet: Unable to connect to remote host: Connection timed out
[03/04/21]seed@VM:~/Desktop$
```

As you can see, I add a new iptables rule to drop all the telnet packets send from machine B to machine A by using the command “sudo iptables -A INPUT -s 10.0.2.8 -p tcp –dport 23 -j DROP”. After added this rule, the telnet command on Machine B will not work.

- Prevent A form visiting an external website.

By using the command “sudo iptables -A OUTPUT -p tcp -d www.youtube.com – dport 80 -j DROP”, I dropped all the http request packets which sending from machine A to request the YouTube websit. Since the YouTube website has multiple Ip address, so the iptables will block them all.

Result:

```
[root@seed ~]# /bin/bash 70x24
Chain OUTPUT (policy ACCEPT)
target    prot opt source          destination
DROP      tcp   --  anywhere       172.217.194.91    tcp dpt:
http
DROP      tcp   --  anywhere       sa-in-f190.1e100.net  tcp dpt:
:httP
DROP      tcp   --  anywhere       172.253.118.136    tcp dpt:
http
DROP      tcp   --  anywhere       sa-in-f136.1e100.net  tcp dpt:
:http
DROP      tcp   --  anywhere       172.217.194.190    tcp dpt:
http
DROP      tcp   --  anywhere       172.217.194.136    tcp dpt:
http
DROP      tcp   --  anywhere       74.125.24.190     tcp dpt:
http
DROP      tcp   --  anywhere       172.217.194.93     tcp dpt:
http
DROP      tcp   --  anywhere       74.125.24.93      tcp dpt:
http
DROP      tcp   --  anywhere       sa-in-f91.1e100.net  tcp dpt:
[03/04/21] seed@VM:~/Desktop$
```

The connection has timed out

The server at www.youtube.com is taking too long to respond.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

[Try Again](#) [Timed Out](#)

As you can see, after added the iptables rule, machine A cannot visit the youtube website.

Task2: Implementing a Simple Firewall

- In order to implement a new firewall rule, a Loadable Kernel Module should be coded which can directly implemented into the kernel process. The LKM file should be coded in C language. I coded 5 different file in order to implement these five different rules.

- o Rule 1:

Code:

```
C KMod.c > void setUpFilter(void)
1  #include <linux/module.h>
2  #include <linux/kernel.h>
3  #include <linux/netfilter.h>
4  #include <linux/init.h>
5  #include <linux/netfilter_ipv4.h>
6  #include <linux/ip.h>
7  #include <linux/tcp.h>
8
9  static struct nf_hook_ops firewallHook;
10
11 unsigned int telnetFilter(void *priv, struct sk_buff *skb, const struct nf_hook_state *state)
12 {
13     struct iphdr *iph;
14     struct tcphdr *tcp;
15     iph = ip_hdr(skb);
16     tcp = (void *)iph + iph->ihl * 4;
17
18     // hardcode filter logic
19     if (iph->protocol == IPPROTO_TCP && tcp->dest == htons(23)) Set the condition to filter all the telnet packets
20     {
21         printk(KERN_INFO "Dropping telnet packet from %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]);
22         return NF_DROP; Drop the packets
23     }
24     else
25     {
26         return NF_ACCEPT;
27     }
28 }
29
30 int setUpFilter(void)
31 {
32     printk(KERN_INFO "Registering a Telnet filter. \n");
33     firewallHook.function = telnetFilter;
34     user_hooks_register(&firewallHook);
35     firewallHook.hooknum = NF_INET_LOCAL_IN; Using the NetFilter Local Input hook
36     firewallHook.priority = PR_INENET;
37     firewallHook.priority = NF_IP_PRI_FIRST;
38
39     // register the hook
40     nf_register_hook(&firewallHook);
41     return 0;
42 }
43 void removeFilter(void)
44 {
45     printk(KERN_INFO "Telnet filter has been removed. \n");
46     nf_unregister_hook(&firewallHook);
47 }
48
49 module_init(setUpFilter);
50 module_exit(removeFilter);
51
52 MODULE_LICENSE("GPL");
```

After finished coding, I create a Makefile to order to compile this file to a LKM.

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "/bin/bash" and the command being run is "GNU nano 2.5.3". The file being edited is "Makefile". The content of the Makefile is:

```
obj-m += kMod.o

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

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

The terminal window also displays a list of keyboard shortcuts at the bottom:

- [Read 7 lines]
- ^G Get Help
- ^O Write Out
- ^W Where Is
- ^K Cut Text
- ^J Justify
- ^X Exit
- ^R Read File
- ^V Replace
- ^U Uncut Tex
- ^T To Spell

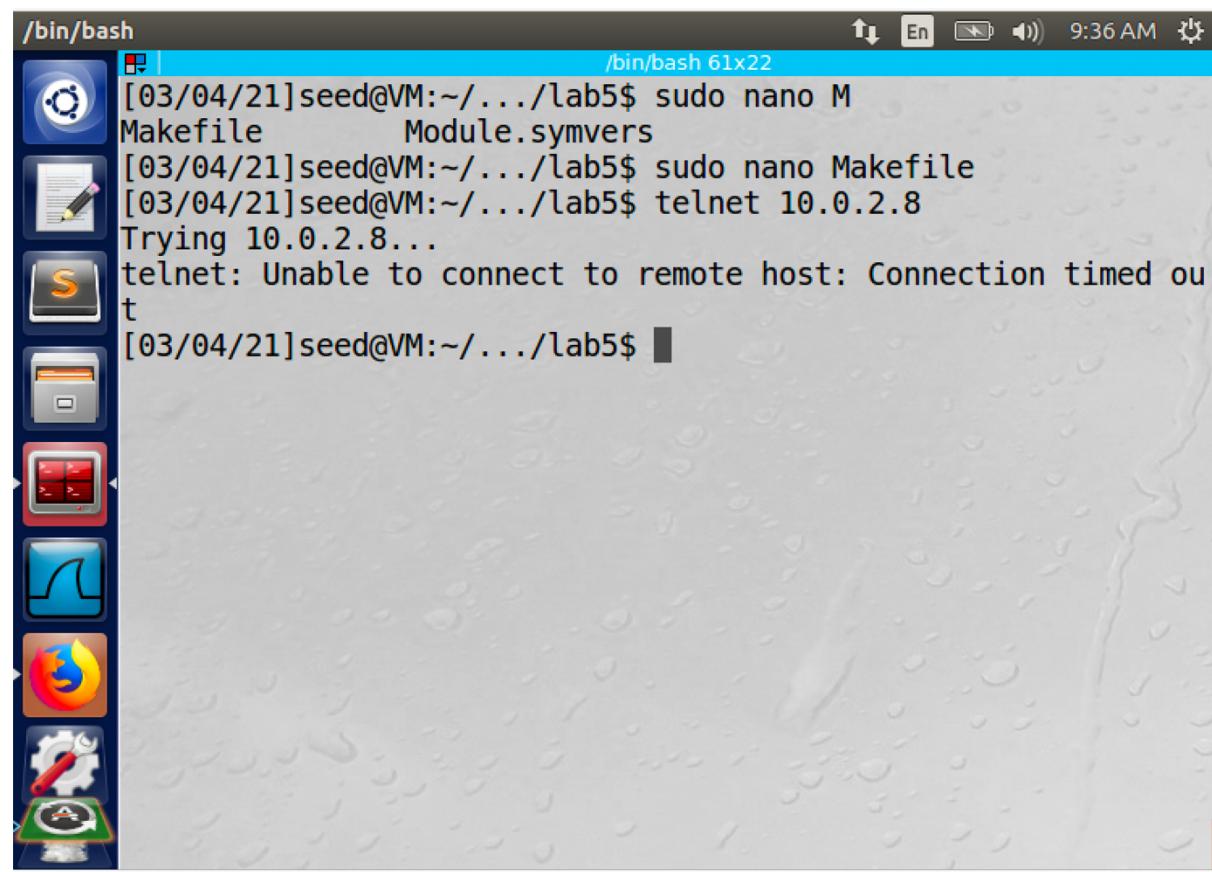
Then I use the command 'make' to create the kMod.ko file which is the LKM File.

```
[03/04/21]seed@VM:~/.../lab5$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/Network-Security/lab5 modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/Network-Security/lab5/kMod.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/Network-Security/lab5/kMod.mod.o
  LD [M]  /home/seed/Network-Security/lab5/kMod.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[03/04/21]seed@VM:~/.../lab5$ ls
kMod.c      kMod.mod.o  Makefile
kMod.ko     kMod.o      modules.order
kMod.mod.c  Lab05.pdf   Module.symvers
[03/04/21]seed@VM:~/.../lab5$ sudo insmod kMod.ko
```

After “make” command I use the command “ sudo insmod kMod.ko” to insert this custom module to the kernel.

```
[03/04/21]seed@VM:~/.../lab5$ lsmod
Module           Size  Used by
kMod            16384  0
xt_tcpudp      16384  17
iptable_filter 16384  1
xt_owner       16384  0
ip_tables      20480  1 iptable_filter
```

After loaded the module, the telnet command will not work.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "/bin/bash" and its content shows the user has run several commands: "sudo nano M" (to edit a file), "sudo nano Makefile" (to edit the Makefile), and "telnet 10.0.2.8" (attempting to connect to a remote host). The connection attempt failed, as indicated by the message "telnet: Unable to connect to remote host: Connection timed out". The desktop interface includes a dock on the left with icons for the terminal, file manager, browser, and system tools, and a taskbar at the bottom with various application icons.

```
/bin/bash
[03/04/21]seed@VM:~/.../lab5$ sudo nano M
Makefile           Module.symvers
[03/04/21]seed@VM:~/.../lab5$ sudo nano Makefile
[03/04/21]seed@VM:~/.../lab5$ telnet 10.0.2.8
Trying 10.0.2.8...
telnet: Unable to connect to remote host: Connection timed out
[03/04/21]seed@VM:~/.../lab5$
```

Then I use the command “dmesg | tail” to check the log.

```
/bin/bash 61x22
06.0-1/input0
[14993.622930] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Du
plex, Flow Control: RX
[16600.906455] Dropping telnet packet from 10.0.2.8
[16601.911557] Dropping telnet packet from 10.0.2.8
[16603.944033] Dropping telnet packet from 10.0.2.8
[16608.022644] Dropping telnet packet from 10.0.2.8
[16616.214904] Dropping telnet packet from 10.0.2.8
[16632.369230] Dropping telnet packet from 10.0.2.8
[16664.855791] Dropping telnet packet from 10.0.2.8
[03/05/21]seed@VM:~$ dmesg | tail
[16759.133196] Bluetooth: L2CAP socket layer initialized
[16759.133201] Bluetooth: SCO socket layer initialized
[16759.137325] Netfilter messages via NETLINK v0.30.
[16781.405226] Dropping telnet packet from 10.0.2.7
[16782.422560] Dropping telnet packet from 10.0.2.7
[16784.438627] Dropping telnet packet from 10.0.2.7
[16788.503003] Dropping telnet packet from 10.0.2.7
[16796.694674] Dropping telnet packet from 10.0.2.7
[16812.823250] Dropping telnet packet from 10.0.2.7
[16845.078761] Dropping telnet packet from 10.0.2.7
[03/05/21]seed@VM:~$
```

- o Rule 2:

The rule number 2 is to filter the HTTP request sending out from A to 157.240.13.35 which is the ip address for facebook website.

Code:

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

static struct nf_hook_ops firewallHook;

unsigned int httpFilter(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;

    // http code filter logic
    if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(80) && ((unsigned char *)iph->daddr)[0] == 157 && ((unsigned char *)iph->daddr)[1] == 240 && ((unsigned char *)iph->daddr)[2] == 13 && ((unsigned char *)iph->daddr)[3] == 35)
        printk(KERN_INFO "Dropping http packet to %d.%d.%d.%d", ((unsigned char *)iph->daddr)[0], ((unsigned char *)iph->daddr)[1], ((unsigned char *)iph->daddr)[2], ((unsigned char *)iph->daddr)[3]);
        return NF_DROP; // Drop the Packet
    }
    else
    {
        return NF_ACCEPT;
    }
}

int setUpFilter(void)
{
    printk(KERN_INFO "Registering a http filter. \n");
    firewallHook.hook = httpFilter;
    // use the netfilter hook
    //firewallHook.hooknum = NF_INET_LOCAL_OUT;
    firewallHook.priority = PF_INET;
    firewallHook.priority = NF_IP_PRI_FIRST;

    // register the hook
    nf_register_hook(&firewallHook);
    return 0;
}
void removeFilter(void)
{
    printk(KERN_INFO "Http filter has been removed. \n");
    nf_unregister_hook(&firewallHook);
}

module_init(setUpFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

Then I repeat the same procedure for rule 1 to load this module to the kernel.

VMA [Running] 5:03 AM

/bin/bash /bin/bash 61x22

```
LD [M] /home/seed/Network-Security/lab5/kMod_http_filter.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[03/05/21]seed@VM:~/.../lab5$ ls
kMod_http_filter.c      Lab05.pdf
kMod_http_filter.ko     Makefile
kMod_http_filter.mod.c  modules.order
kMod_http_filter.mod.o  Module.symvers
kMod_http_filter.o

[03/05/21]seed@VM:~/.../lab5$ sudo insmod kMod_http_filter.ko
[03/05/21]seed@VM:~/.../lab5$ lsmod
Module                Size  Used by
kMod_http_filter        16384  0
nfnetlink_queue         20480  0
nfnetlink_log           20480  0
nfnetlink               16384  2 nfnetlink_log,nfnetlink_queue
bluetooth              491520  0
cfg80211               512000  0
kMod                   16384  0
xt_tcpudp              16384  15
iptable_filter          16384  1
```

After the module is loaded, the facebook website is not reachable.

VMA [Running] 5:10 AM

Problem loading page - Mozilla Firefox

Insecure Connection Problem loading page +

File Edit View History Bookmarks Tools Help

157.240.13.35

Most Visited SEED Labs Sites for Labs Bookmark this page (Ctrl+D)

The connection has timed out

The server at 157.240.13.35 is taking too long to respond.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Timed Out Try Again

The log message is:

```
[03/05/21]seed@VM:~/.../lab5$ dmesg | tail
[19058.262491] Dropping http packet to 157.240.13.35
[19058.519229] Dropping http packet to 157.240.13.35
[19062.297916] Dropping http packet to 157.240.13.35
[19062.562466] Dropping http packet to 157.240.13.35
[19070.486481] Dropping http packet to 157.240.13.35
[19070.742771] Dropping http packet to 157.240.13.35
[19086.614770] Dropping http packet to 157.240.13.35
[19086.870598] Dropping http packet to 157.240.13.35
[19120.406560] Dropping http packet to 157.240.13.35
[19120.406563] Dropping http packet to 157.240.13.35
[03/05/21]seed@VM:~/.../lab5$
```

o Rule3:

I create rule 3 to prohibit the ping request for machine A

Code:

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

static struct nf_hook_ops firewallHook;

unsigned int icmpFilter(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;

    // hardcoded filter logic
    if (iph->protocol == IPPROTO_ICMP) Set the condition to filter the ICMP packets
    {
        printk(KERN_INFO "Dropping ICMP packet from %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 ICMP filter.\n");
    firewallHook.function = icmpFilter;
    // use the netfilter hook
    firewallHook.hooknum = NF_INET_LOCAL_IN; Use the local in hook
    firewallHook.priority = NF_IP_PRI_FIRST;

    // register the hook
    nf_register_hook(&firewallHook);
    return 0;
}

void removeFilter(void)
{
    printk(KERN_INFO "ICMP filter has been removed.\n");
    nf_unregister_hook(&firewallHook);
}

module_init(setupFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

After the same procedure to load this module, all the ping request to A will not be successful.

Result:

The screenshot shows a Linux desktop environment with several windows open:

- A terminal window titled "/bin/bash" (VMA [Running]) showing the output of "ls" and "lsmod" commands. It lists kernel modules kMod_icmp_filter and kMod_http_filter.
- A terminal window titled "/bin/bash" (VMA [Running]) showing the output of "dmesg | tail" command, which logs ICMP packets being dropped from 10.0.2.7.
- A terminal window titled "/bin/bash" (VMB [Running]) showing the output of "ping 10.0.2.7" command, which is stopped with ^C.
- A Firefox Web Browser window titled "Firefox Web Browser".

- Rule4:

I create rule 4 to only allow then ssh request from 10.0.2.8.

Code:

```

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

static struct nf_hook_ops firewallHook;

unsigned int sshFilter(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;

    // hardcode filter logic
    // only allow 10.0.2.8 to ssh to machine A
    Set the condition
    if (tcph->dest == htons(22) && ((unsigned char *)&iph->saddr)[0] == 10 && ((unsigned char *)&iph->saddr)[1] == 0 && ((unsigned char *)&iph->saddr)[2] == 2 && ((unsigned char *)&iph->saddr)[3] == 8)
    {
        printk(KERN_INFO "Accepting packet from %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]);
        return NF_ACCEPT;
    }
    else
    {
        printk(KERN_INFO "Drop packet from %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]);
        return NF_DROP;
    }
}

int setUpFilter(void)
{
    printk(KERN_INFO "Registering a ssh filter. \n");
    firewallHook.hook = sshFilter;
    // use the netfilter hook
    firewallHook.hooknum = NF_INET_PRE_ROUTING;
    firewallHook.pf = PF_INET;
    firewallHook.priority = NF_IP_PRI_FIRST;

    // register the hook
    nf_register_hook(&firewallHook);
    return 0;
}

void removeFilter(void)
{
    printk(KERN_INFO "ssh filter has been removed. \n");
    nf_unregister_hook(&firewallHook);
}

module_init(setUpFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");

```

Result:

```
ic'
[05/21]seed@VM:~/.../lab5$ sudo rmmod kMod_ssh filter
[05/21]seed@VM:~/.../lab5$ sudo insmod kMod_ssh_filter.ko
[05/21]seed@VM:~/.../lab5$ dmesg | tail
[86.0007361] Accepting packet from 10.0.2.8
[86.0007580] Accepting packet from 10.0.2.8
[86.0009511] Accepting packet from 10.0.2.8
[86.107535] Accepting packet from 10.0.2.8
[90.0018719] Drop packet from 127.0.1.1
[90.0018723] Drop packet from 127.0.1.1
[90.0019182] Drop packet from 10.0.2.7
[95.024803] Drop packet from 127.0.1.1
[95.025437] Drop packet from 10.0.2.7
[95.025440] Drop packet from 10.0.2.7
[05/21]seed@VM:~/.../lab5$ █
[03/05/21]seed@VM:~$ ssh 10.0.2.7
^C
[03/05/21]seed@VM:~$ ssh 10.0.2.7
^C
[03/05/21]seed@VM:~$ ssh 10.0.2.7
seed@10.0.2.7's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-1013-aws x86_64)

 * 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: Fri Mar 5 04:27:50 2021 from 10.0.2.1
\[03/05/21\]seed@VM:~\$ █
```

- o Rule 5:

I use the rule 5 to edit all the source ip address to 1.2.3.4.

Code:

```

1 #include <linux/module.h>
2 #include <linux/kernel.h>
3 #include <linux/netfilter.h>
4 #include <linux/init.h>
5 #include <linux/netfilter_ipv4.h>
6 #include <linux/ip.h>
7 #include <linux/tcp.h>
8
9 static struct nf_hook_ops firewallHook;
10
11 unsigned int editipFilter(void *priv, struct sk_buff *skb, const struct nf_hook_state *state)
12 {
13     struct iphdr *iph;
14     struct tcphdr *tcph;
15     iph = ip_hdr(skb);
16     tcph = (void *)iph + iph->ihl * 4;
17
18     // hardcoded filter logic
19     // edit all the src IP address to 1.2.3.4
20     if (iph->protocol == IPPROTO_ICMP)
21     {
22         printk(KERN_INFO "Editing ICMP packet from %d.%d.%d.%d\n",
23               ((unsigned char *)iph->saddr)[0] = 1;
24               ((unsigned char *)iph->saddr)[1] = 2;
25               ((unsigned char *)iph->saddr)[2] = 3;
26               ((unsigned char *)iph->saddr)[3] = 4;
27         skb->ip_summed = 1;
28         return NF_ACCEPT;
29     }
30     else
31     {
32         return NF_DROP;
33     }
34 }
35
36 int setUpFilter(void)
37 {
38     printk(KERN_INFO "Registering a editip filter. \n");
39     firewallHook.hook = editipFilter;
40     // use the netfilter hook
41     firewallHook.hooknum = NF_INET_POST_ROUTING;
42     firewallHook.pf = PF_INET;
43     firewallHook.priority = NF_IP_PRI_FIRST;
44
45     // register the hook
46     nf_register_hook(&firewallHook);
47     return 0;
48 }
49 void removeFilter(void)
50 {
51     printk(KERN_INFO "Editip filter has been removed. \n");
52     nf_unregister_hook(&firewallHook);
53 }
54
55 module_init(setUpFilter);
56 module_exit(removeFilter);
57
58 MODULE_LICENSE("GPL");
59

```

Result:

No.	Time	Source	Destination
1	2021-03-05 08:30:18.0785235...	10.0.2.9	10.0.2.7
2	2021-03-05 08:30:18.0785533...	1.2.3.4	10.0.2.9
3	2021-03-05 08:30:18.0786000...	::1	::1
4	2021-03-05 08:30:19.0823146...	10.0.2.9	10.0.2.7
5	2021-03-05 08:30:19.0823416...	1.2.3.4	10.0.2.9
6	2021-03-05 08:30:20.1063740...	10.0.2.9	10.0.2.7
7	2021-03-05 08:30:20.1064566...	1.2.3.4	10.0.2.9
8	2021-03-05 08:30:21.1291705...	10.0.2.9	10.0.2.7
9	2021-03-05 08:30:21.1292246...	1.2.3.4	10.0.2.9
10	2021-03-05 08:30:22.1537730...	10.0.2.9	10.0.2.7
11	2021-03-05 08:30:22.1538282...	1.2.3.4	10.0.2.9

[23145.986939] Registering a editip filter.
[23188.195272] Editing ICMP packet from 10.0.2.7
[23189.199061] Editing ICMP packet from 10.0.2.7
[23190.223155] Editing ICMP packet from 10.0.2.7
[23191.245925] Editing ICMP packet from 10.0.2.7
[23192.270528] Editing ICMP packet from 10.0.2.7
[23193.292964] Editing ICMP packet from 10.0.2.7
[03/05/21]seed@VM:~/.../lab5\$

Task 3: Evading Egress Filtering

- 3a:

- o Step1:

I add a new iptables rule to block all the telnet request to 10.0.2.8 which is machine B by using the command “sudo iptables -A OUTPUT -p tcp -d 10.0.2.8 –dport 23 -j DROP”. After this new rule is added the telnet command will not work.

- o Step2:

In order to contact machine B, I use the command “ssh -L 8000:10.0.2.8:23 seed@10.0.2.8” to create a ssh tunnel which liked 8000 port to the 23 port of machine B. After the tunnel is created, I am able to telnet the machine B by telnet the localhost port 8000:

- o Result:

```

Last login: Fri Mar  5 07:56:43 2021 from 10.0.2.8
[03/07/21]seed@VM:~$ telnet localhost 8000
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
[03/07/21]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet HWaddr 08:00:27:66:b0:79
             inet addr:10.0.2.8 Bcast:10.0.2.255 Mask:255.255.
                           255.0
                           inet6 addr: fe80::a80e:6b3f:e26b:f414/64 Scope:Link
                           UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                           RX packets:857 errors:0 dropped:0 overruns:0 frame:0
                           TX packets:1030 errors:0 dropped:0 overruns:0 carrier:0
                           collisions:0 txqueuelen:1000
                           RX bytes:214762 (214.7 KB)  TX bytes:96128 (96.1 KB)

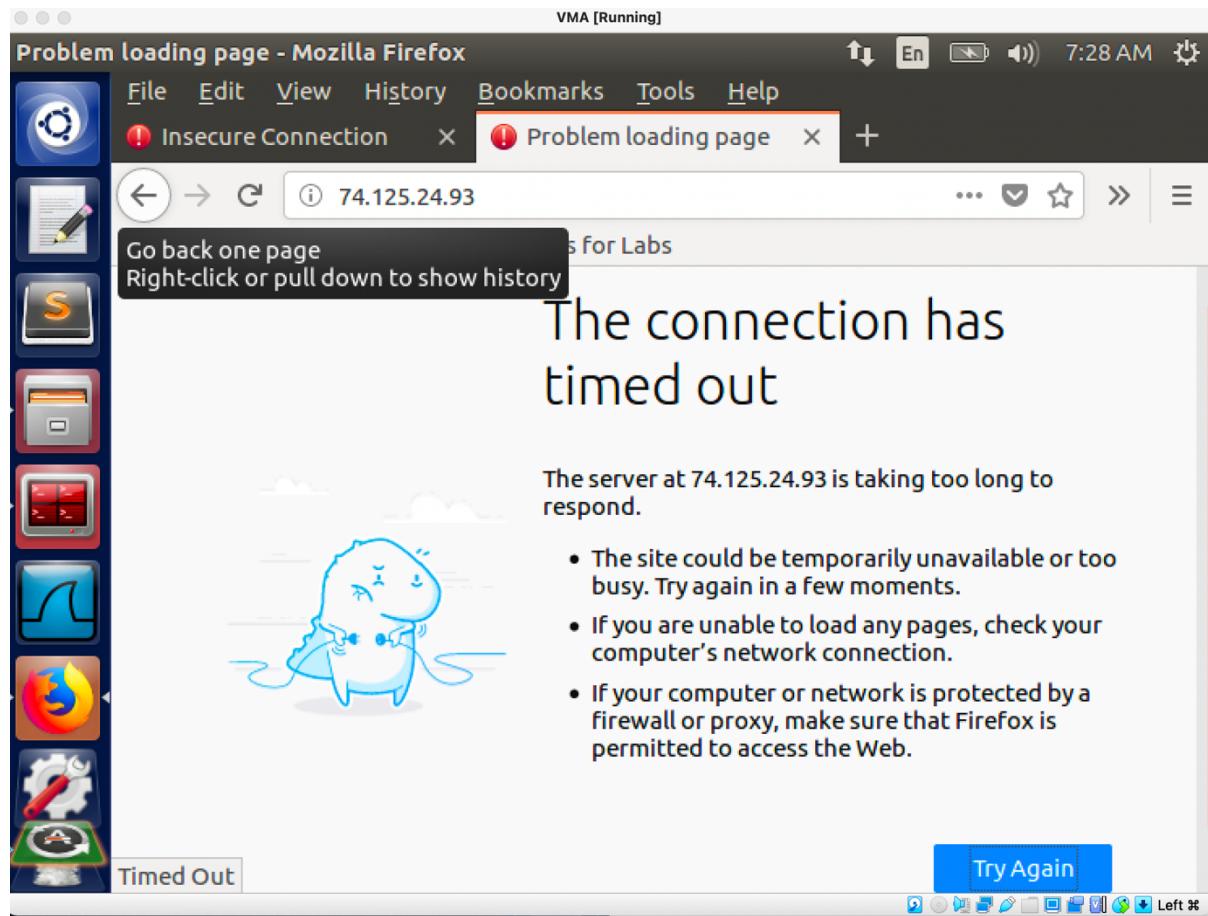
lo         Link encap:Local Loopback
             inet addr:127.0.0.1 Mask:255.0.0.0
             inet6 addr: ::1/128 Scope:Host

```

- 3b:

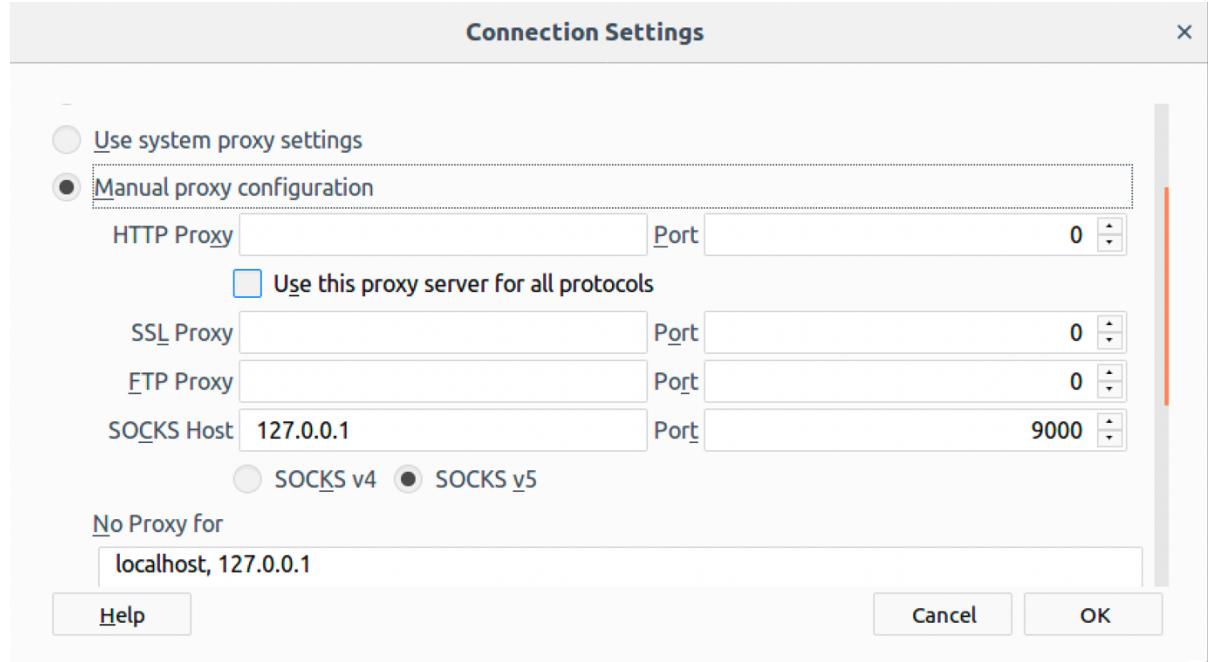
- o Step1:

I use the command “sudo iptables -A OUTPUT -p tcp -d 74.125.24.93 –dport 80 -j DROP” to add a new iptables rule to block the machine A to visit 74.125.24.93 which is one of the ip address of google. After the rule is added, machine A is not able to visit that website.



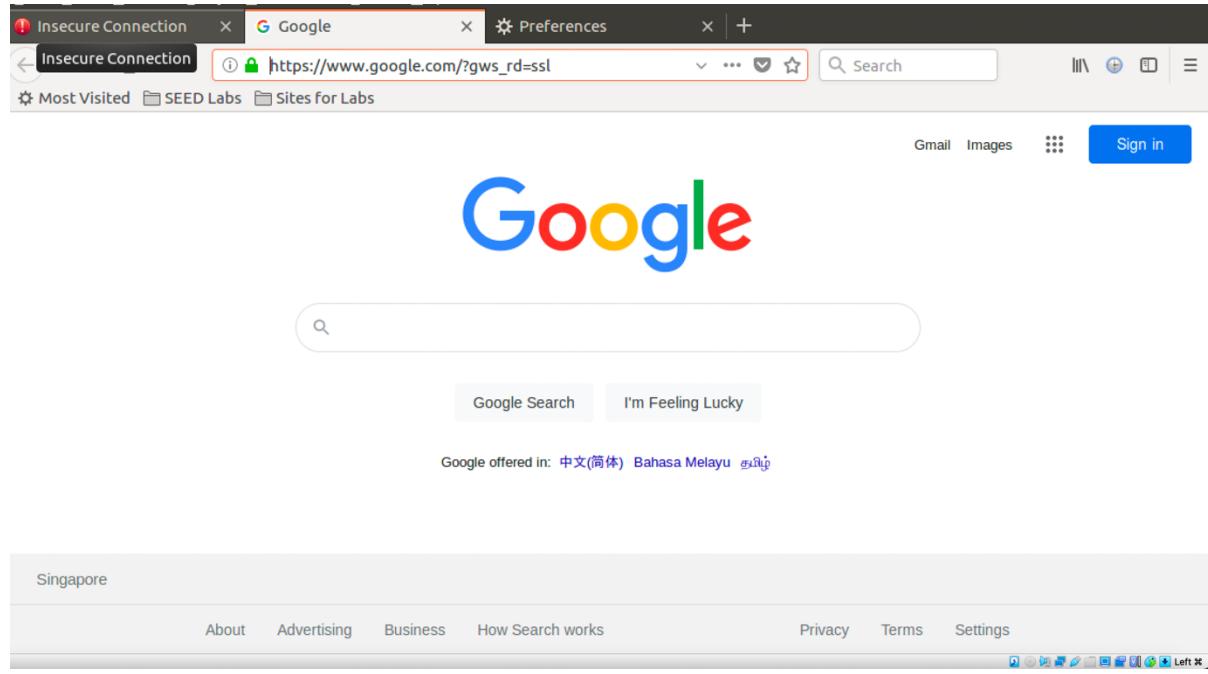
o Step2:

I set up the Firefox proxy to help the machine A to reach the website.



After I set up the proxy, machine A are able to reach the google website.

Result:



- Task4: Evading Ingress Filtering

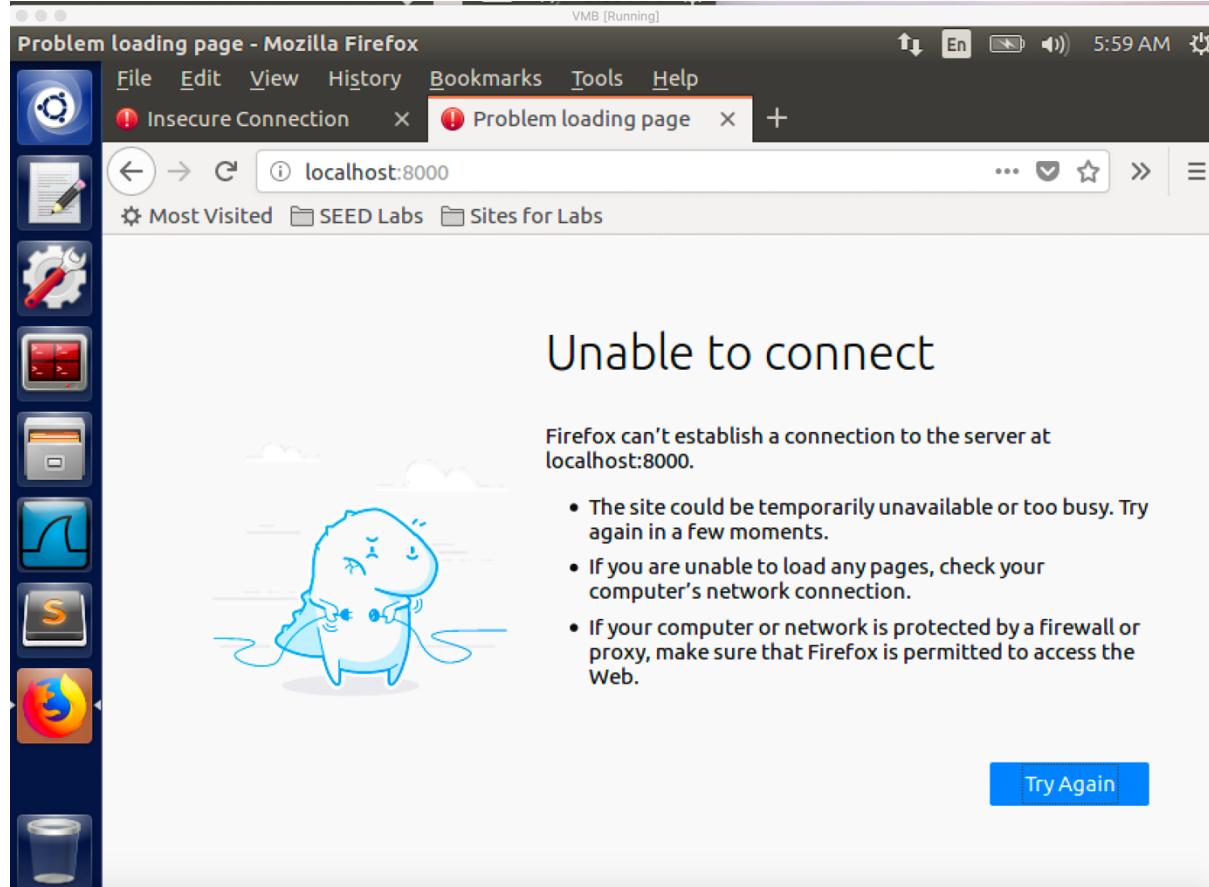
- o Step1:

Initially, I added two iptables rules to block machine B from accessing machine A's port 80 and 22.

```
[03/07/21]seed@VM:~$ sudo iptables -A INPUT -p tcp --dport 80 -j DROP
[03/07/21]seed@VM:~$ sudo iptables -A INPUT -p tcp --dport 22 -j DROP
[03/07/21]seed@VM:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                               destination
DROP      tcp  --  anywhere
          tcp dpt:http
DROP      tcp  --  anywhere
          tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source                               destination
```

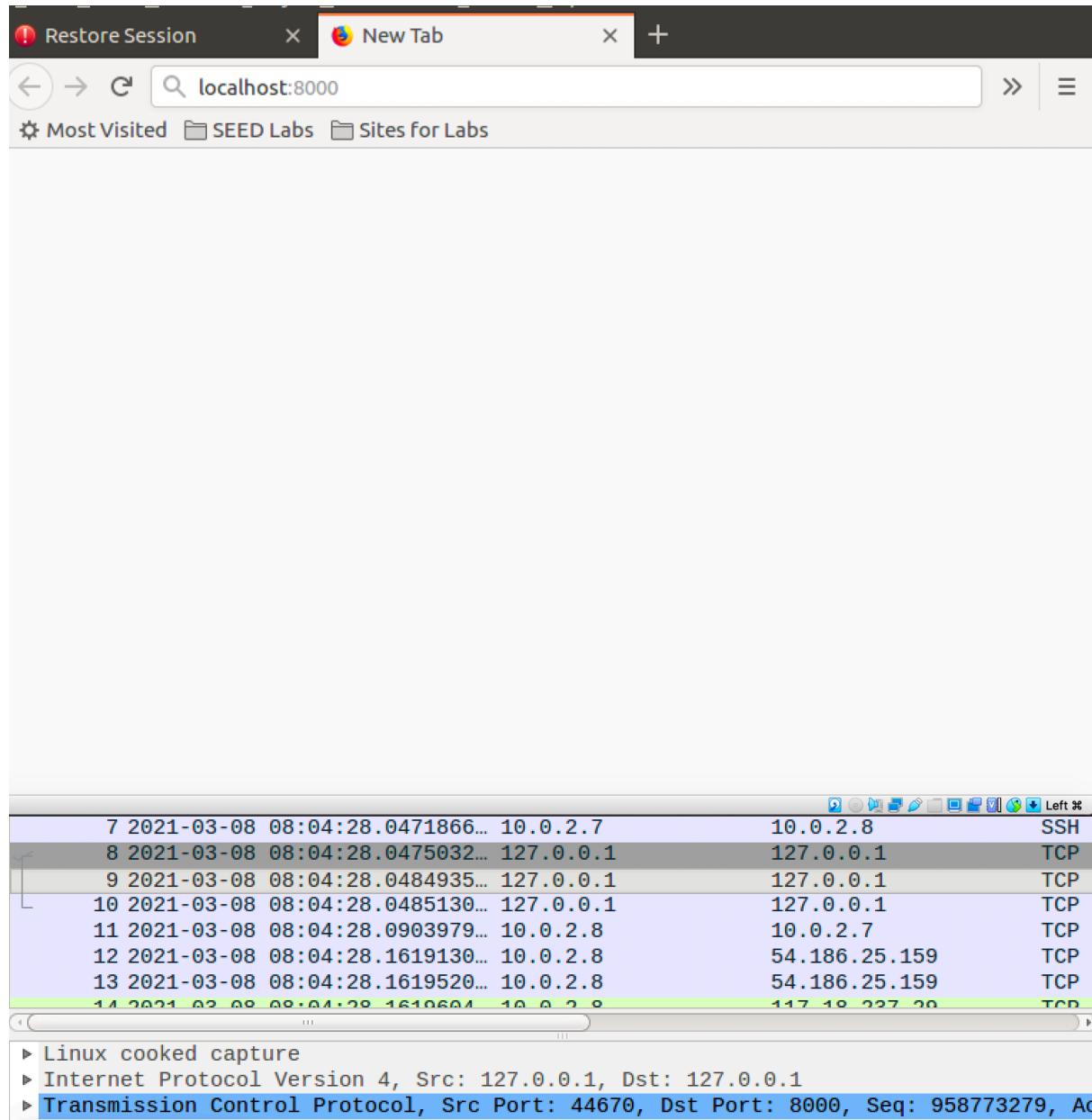
Under this condition, machine B are not able to access the secret server at machine A.



- Step2:

I created the reverse tunnel at machine A in order to let machine B to access the secret sever at machine A by using the command “ ssh -R 8000:localhost:80 seed@10.0.2.8 ”. I connected the local port 8000 of machine B to port 80 of machine A. Under this condition machine B is able to access the secret server.

Result:



As you can see, the destination of the TCP packet is sending to port 8000 and machine B is able to access the secret server.