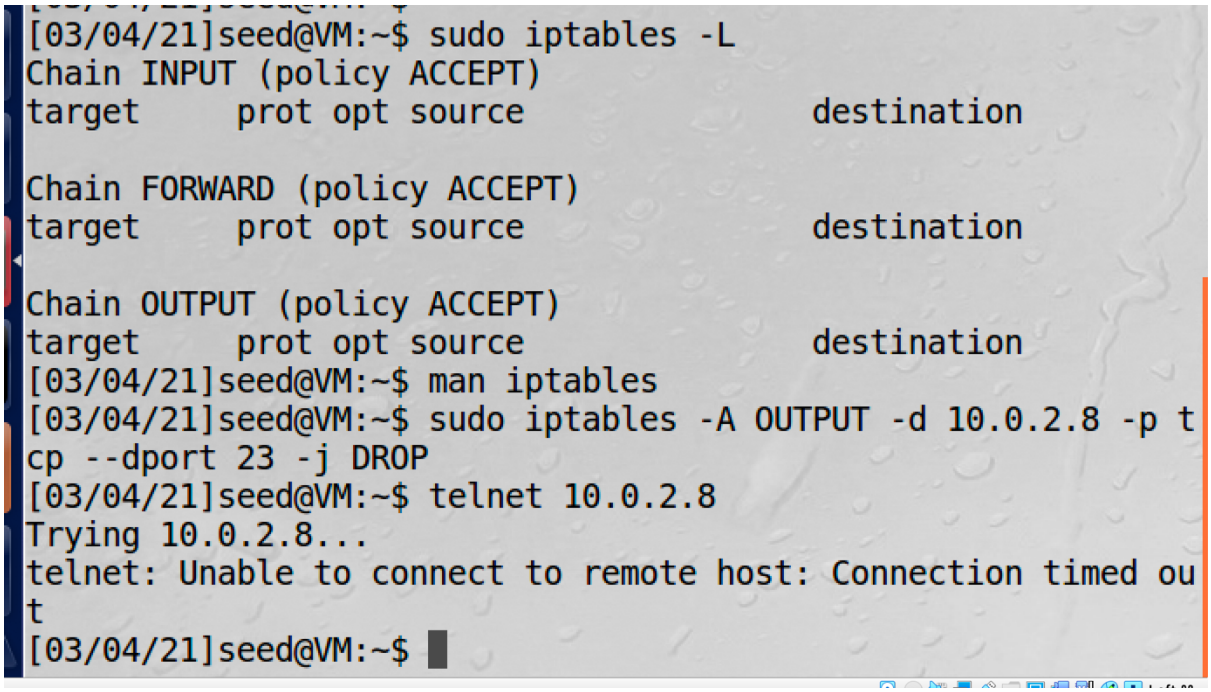
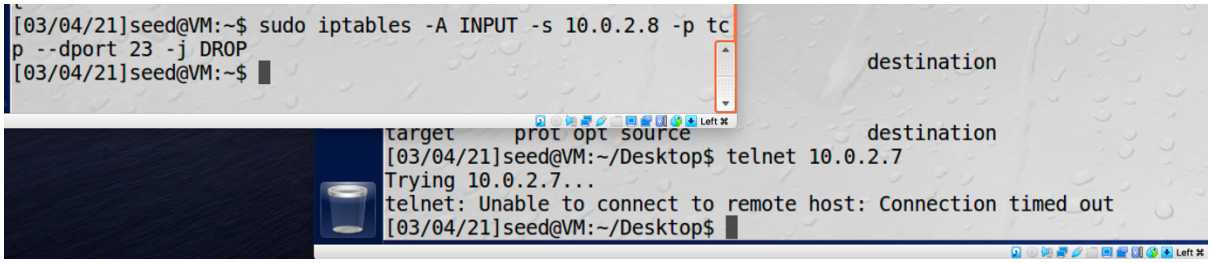
Machine A: 10.0.2.7
Machine B: 10.0.2.8

Task1: Using Firewall

- Prevent A from doing telnet to Machine B.
  Code & result:

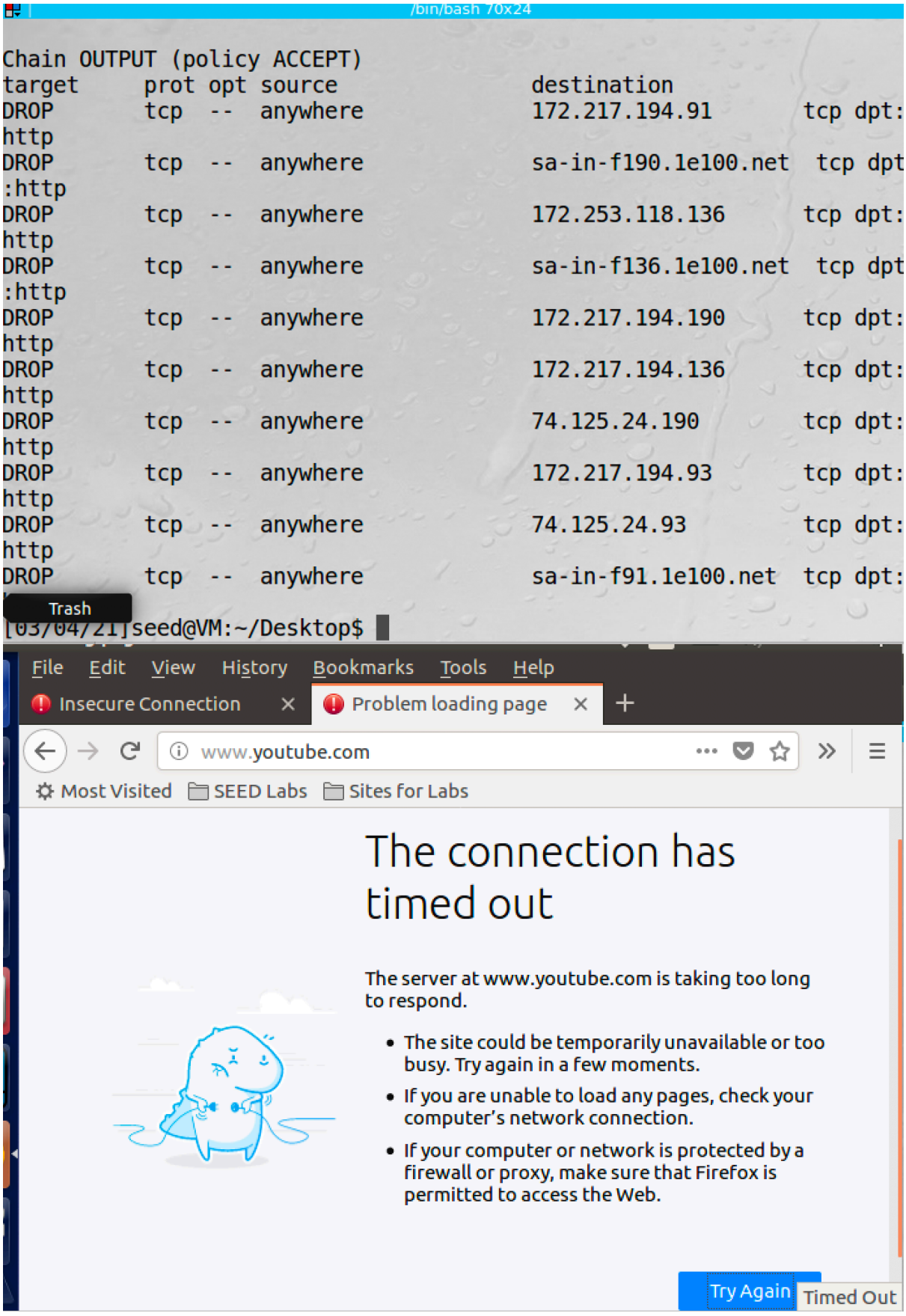


  As you can see, I use the command 'sudo iptables -A OUTPUT -d 10.0.2.8 -p tcp –dport 23 -j DROP' on machine A to drop all the packets which send form machine A to 10.0.2.8 which is machine B with destination prot of 23. Which means I drop all the telnet packet which send from A to B. After I added this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



  As you can see, I add a new iptables rule to drop all the telnet packets send form 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 form machine A to request the YouTube websit. Since the YouTube website has multiple Ip address, so the iptables will block them all.

Result:

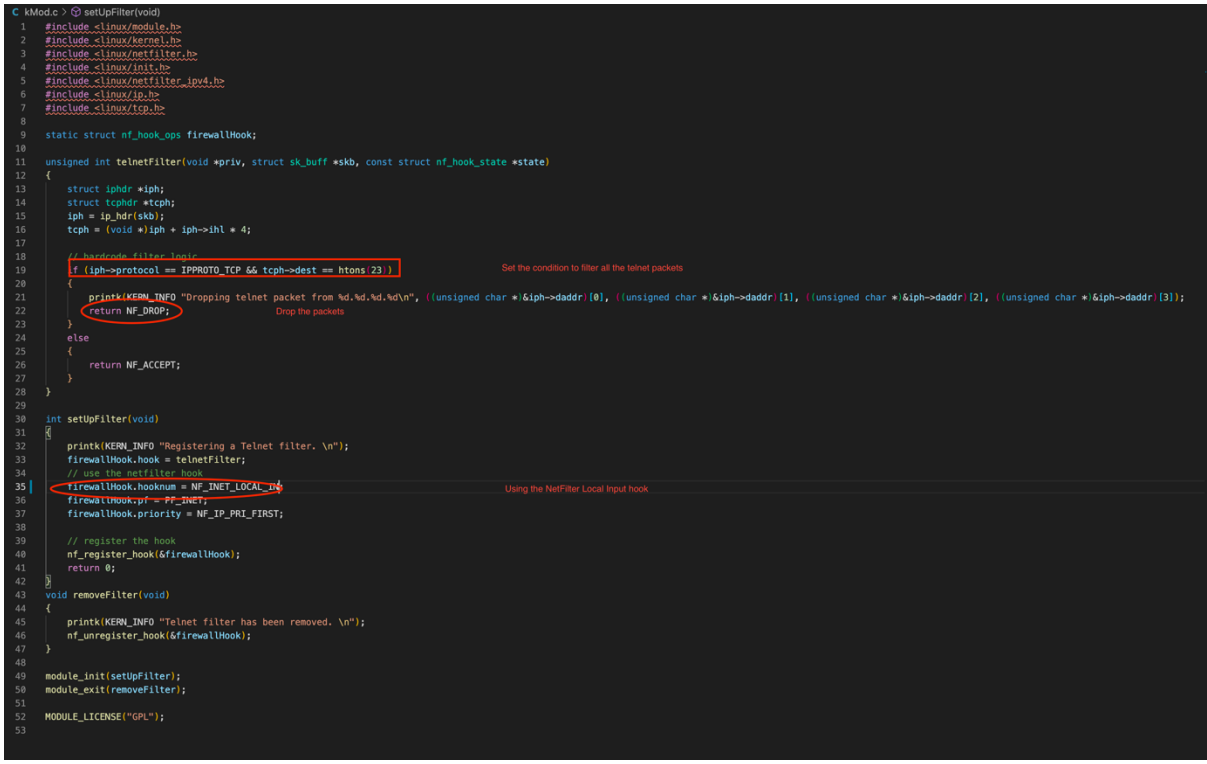

/bin/bash 70x24
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
DROP tcp -- anywhere 172.217.194.91 tcp dpt:http
DROP tcp -- anywhere sa-in-f190.1e100.net tcp dpt:http
DROP tcp -- anywhere 172.253.118.136 tcp dpt:http
DROP tcp -- anywhere sa-in-f136.1e100.net tcp dpt:http
DROP tcp -- 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:
Trash
[03/04/21]seed@VM:~/Desktop$
File Edit View History Bookmarks Tools Help
Insecure Connection
Problem loading page
www.youtube.com
Most Visited
SEED Labs
Sites for Labs
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.

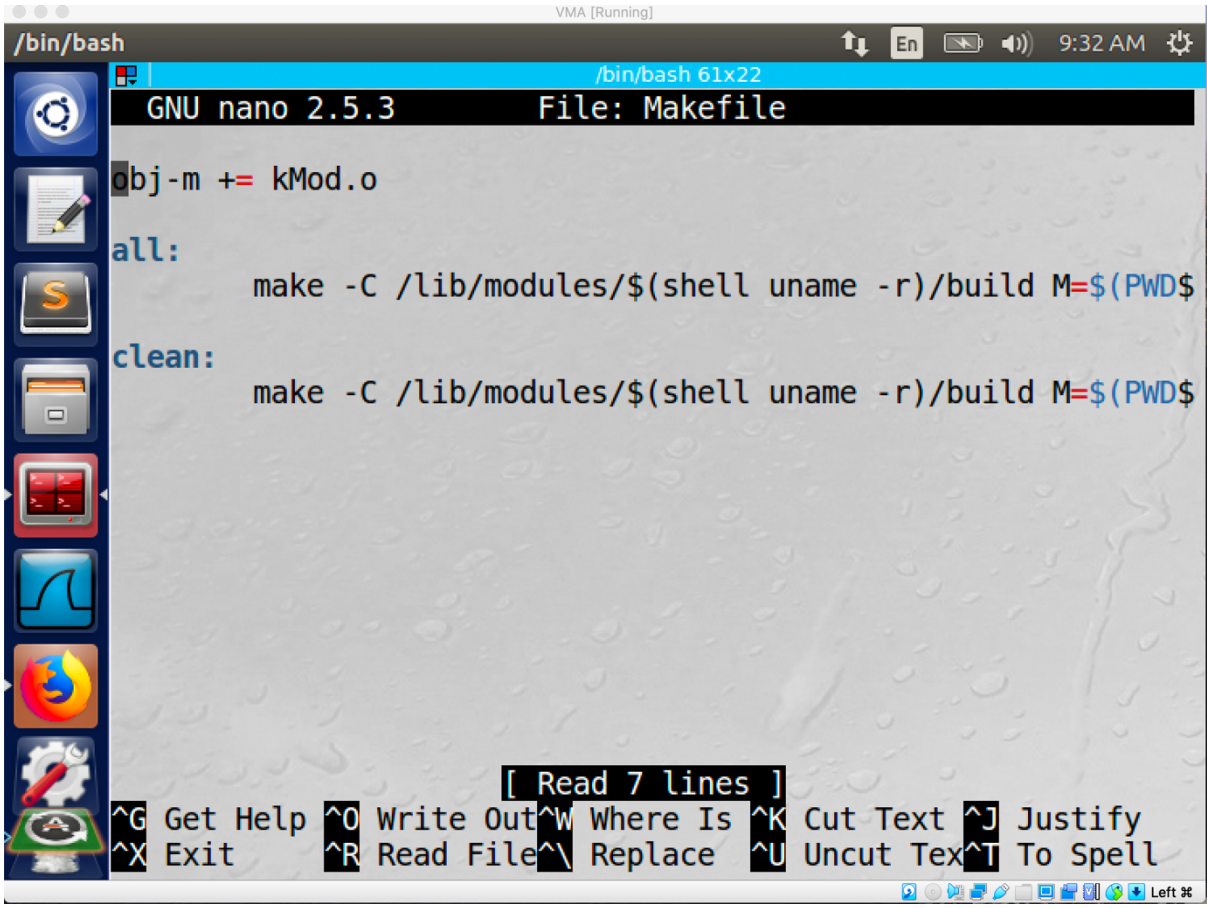
Task2: Implementing a Simple Firewall

- 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.
    - Rule 1:

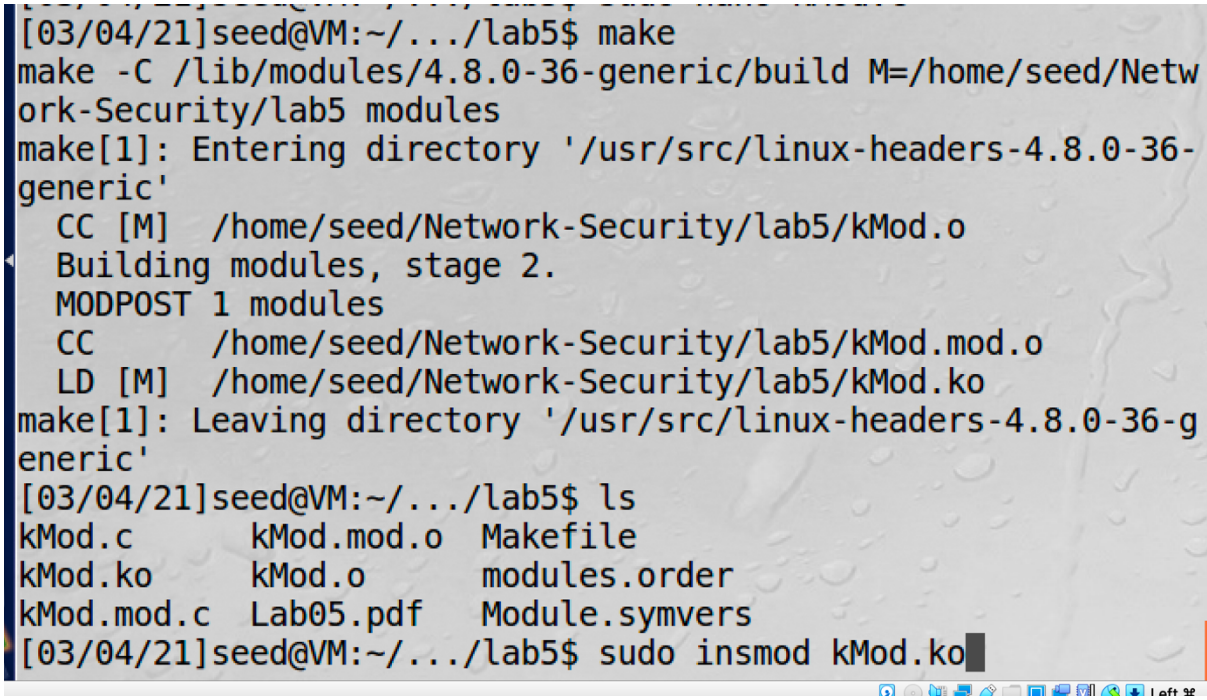
      Code:



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



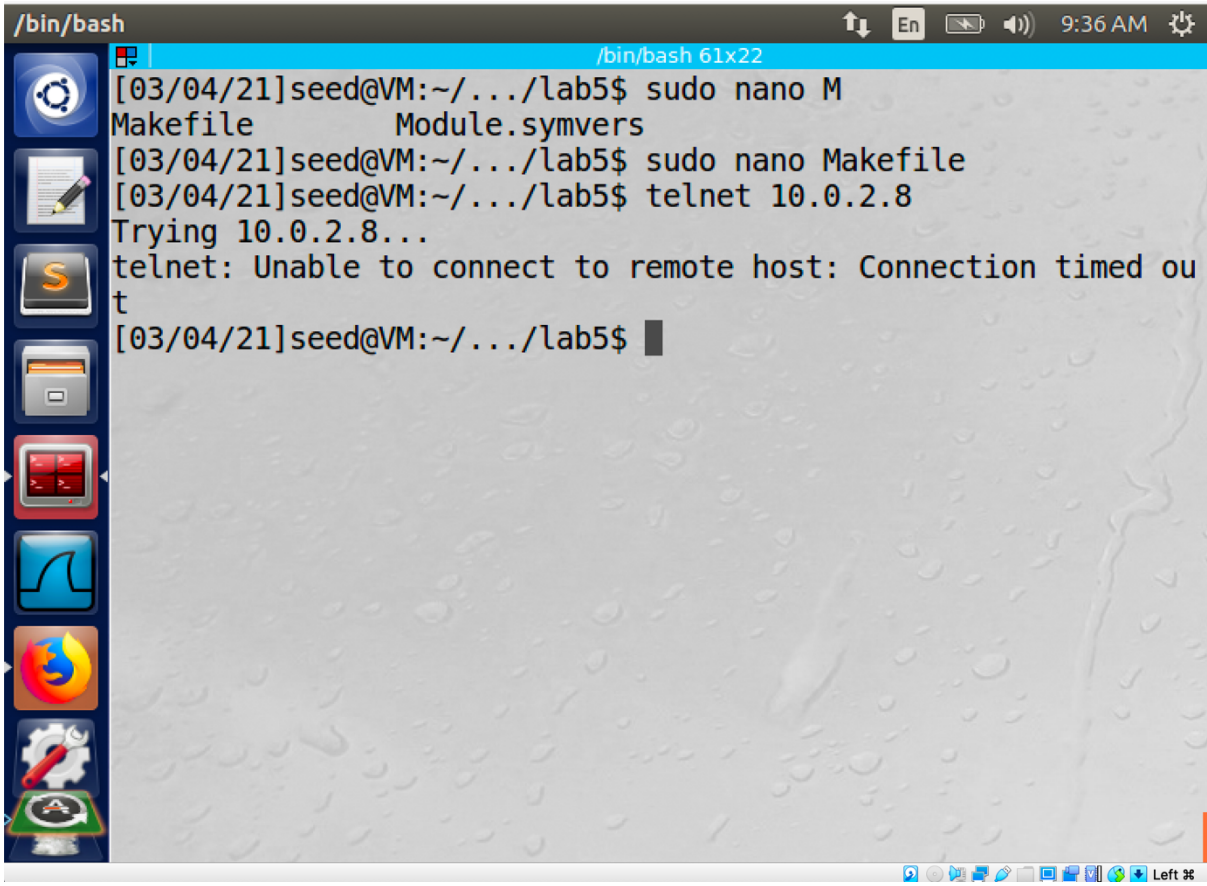
Then I use the command 'make' to create the kMod.ko file which is the LKM File.



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.



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:~$
```

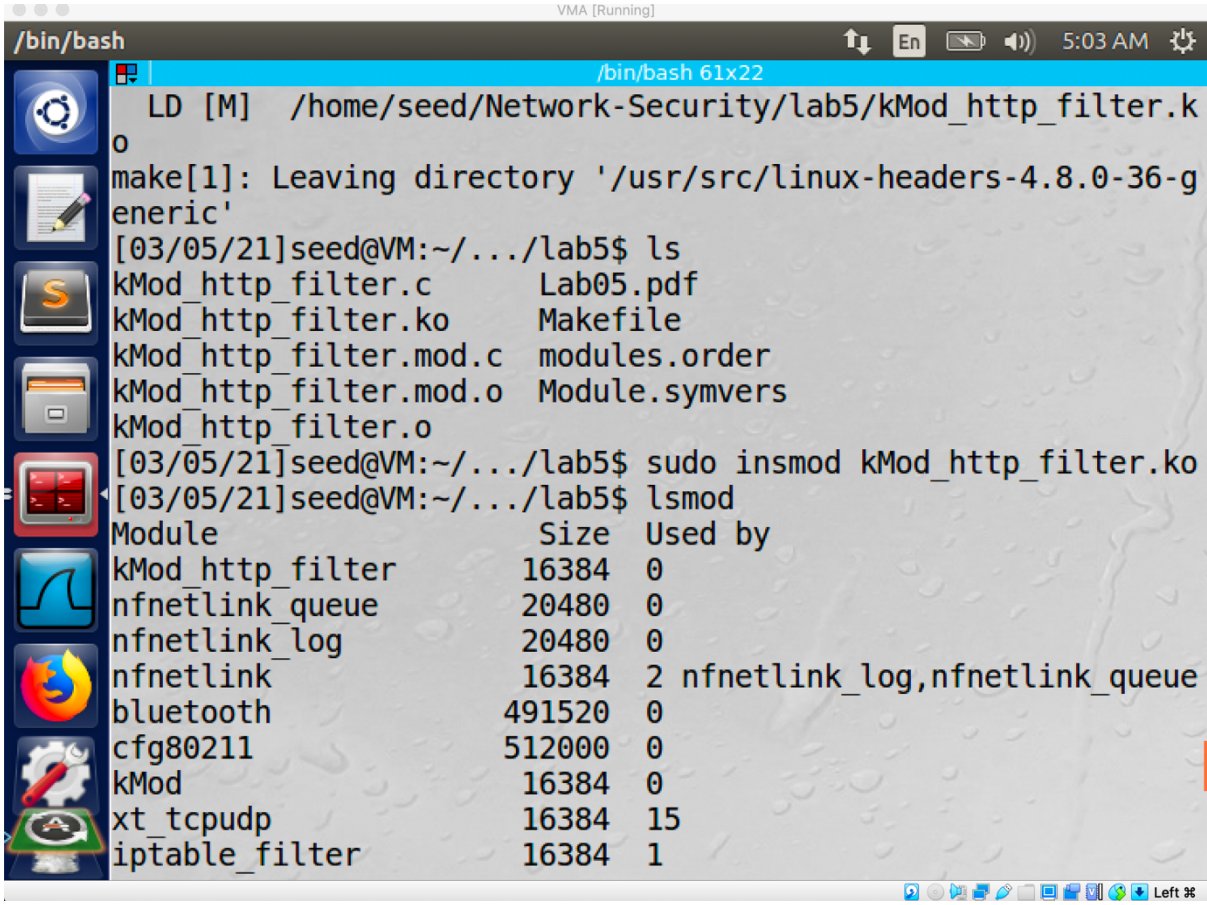
- Rule 2:

  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.

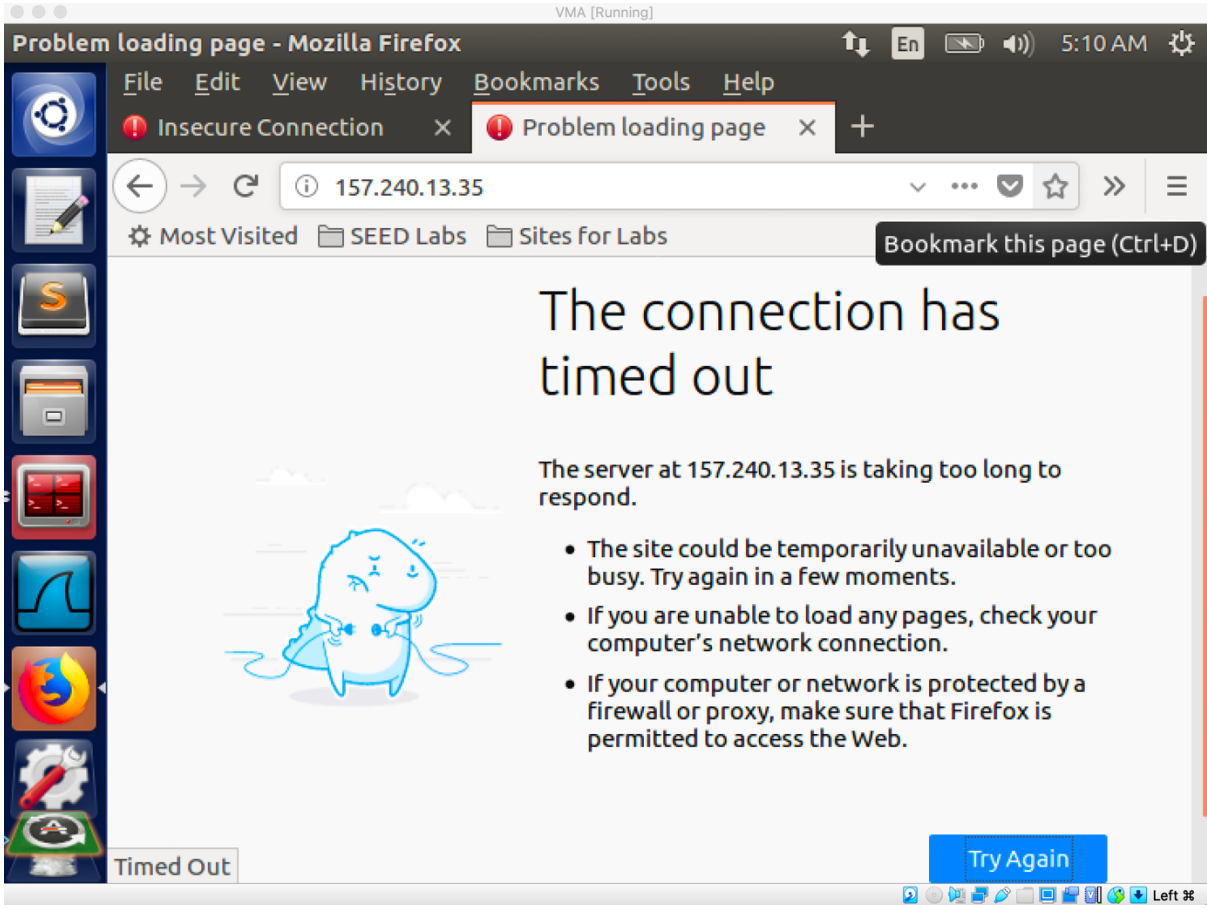
  Code:

```c
#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 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;                                  Setting up the condition to filter the http request packet send to 157.240.13.35

    // hardcode 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\n", ((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;                           Using the Netfilter Local_OUT hook
    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 "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.



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



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$
```

- Rule3:

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

  Code:

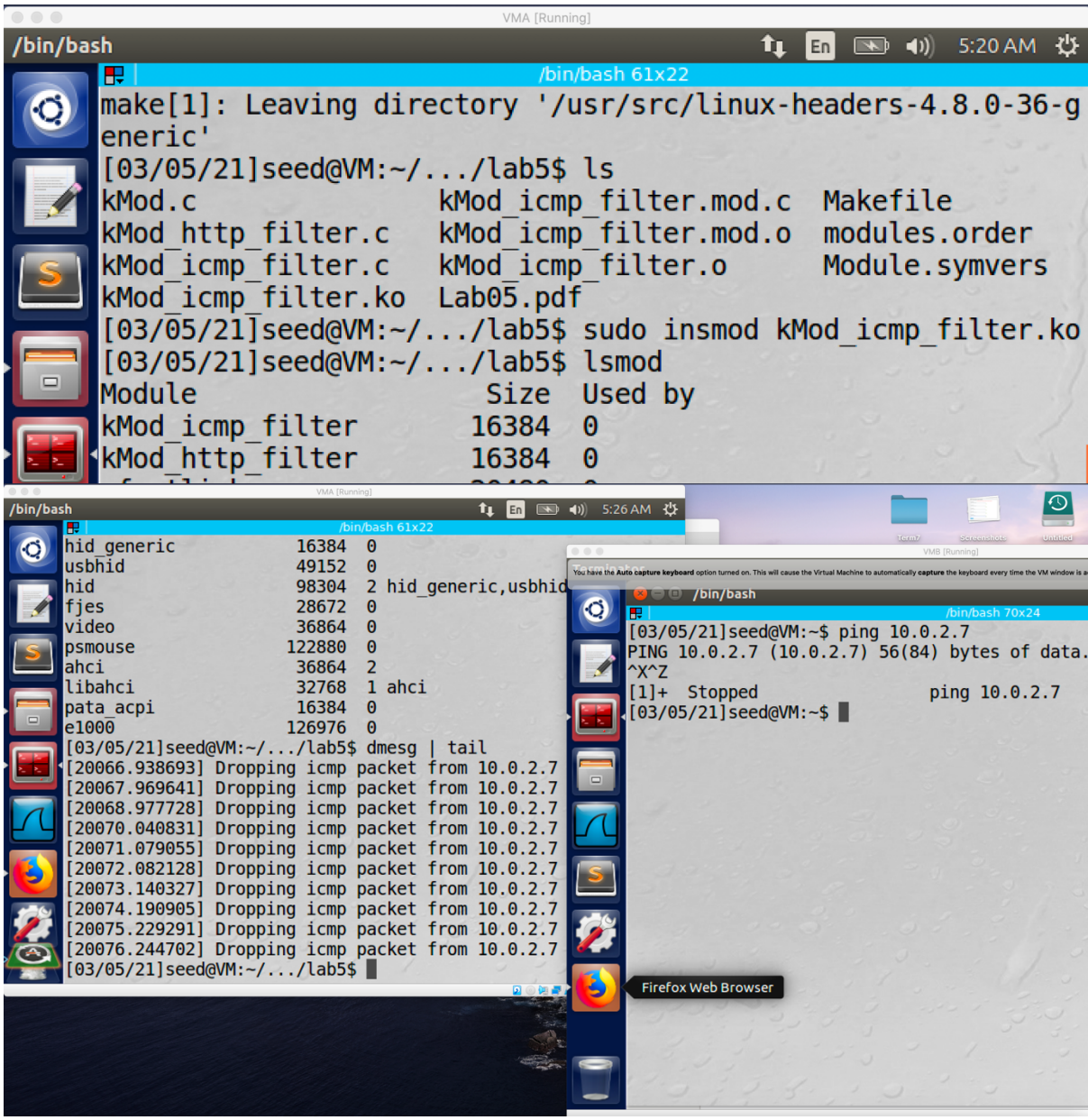
```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/netfilter.h>
#include <linux/init.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>

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;

    // hardcode filter logic
    if (iph->protocol == IPPROTO_ICMP)
    {
        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.hook = icmpFilter;
    // use the netfilter hook
    firewallHook.hooknum = NF_INET_LOCAL_IN;
    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 "Icmp filter has been removed. \n");
    nf_unregister_hook(&firewallHook);
}

module_init(setUpFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

  Set the condition to filter the ICMP packets

  Use the local in hood

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

  Result:



- Rule4:
  I create rule 4 to only allow then ssh request from 10.0.2.8.
  Code:

```c
#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                                    Set the condition
    // only allow 10.0.2.8 to ssh to machine A
    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;                 Using the Pre Routing hook
    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
86.007361] Accepting packet from 10.0.2.8
86.007580] Accepting packet from 10.0.2.8
86.009511] Accepting packet from 10.0.2.8
86.107535] Accepting packet from 10.0.2.8
90.018719] Drop packet from 127.0.1.1
90.018723] Drop packet from 127.0.1.1
90.019182] 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-

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.
 * 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.
[03/05/21]seed@VM:~$
```

- Rule 5:

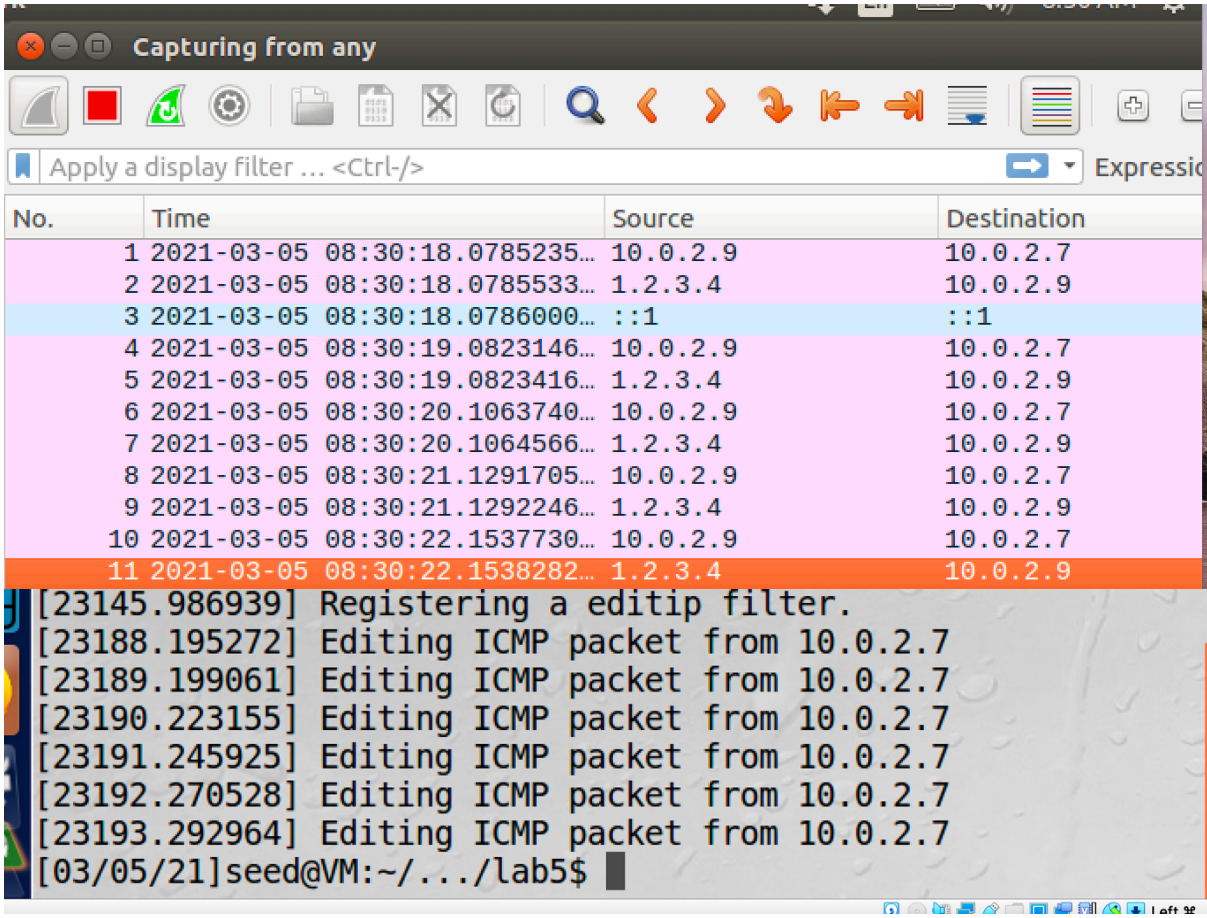
  I use the rule 5 to edit all the source ip address to 1.2.3.4.

  Code:

```c
#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 editipFilter(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
    // edit all the src ip address to 1.2.3.4
    if (iph->protocol == IPPROTO_ICMP)
    {
        printk(KERN_INFO "Editing ICMP 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]);
        ((unsigned char *)&iph->saddr)[0] = 1;
        ((unsigned char *)&iph->saddr)[1] = 2;
        ((unsigned char *)&iph->saddr)[2] = 3;
        ((unsigned char *)&iph->saddr)[3] = 4;
        skb->ip_summed = 1;
        return NF_ACCEPT;
    }
    else
    {
        return NF_DROP;
    }
}

int setUpFilter(void)
{
    printk(KERN_INFO "Registering a editip filter. \n");
    firewallHook.hook = editipFilter;
    // use the netfilter hook
    firewallHook.hooknum = NF_INET_POST_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 "Editip filter has been removed. \n");
    nf_unregister_hook(&firewallHook);
}

module_init(setUpFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

Edit the src ip address to 1.2.3.4

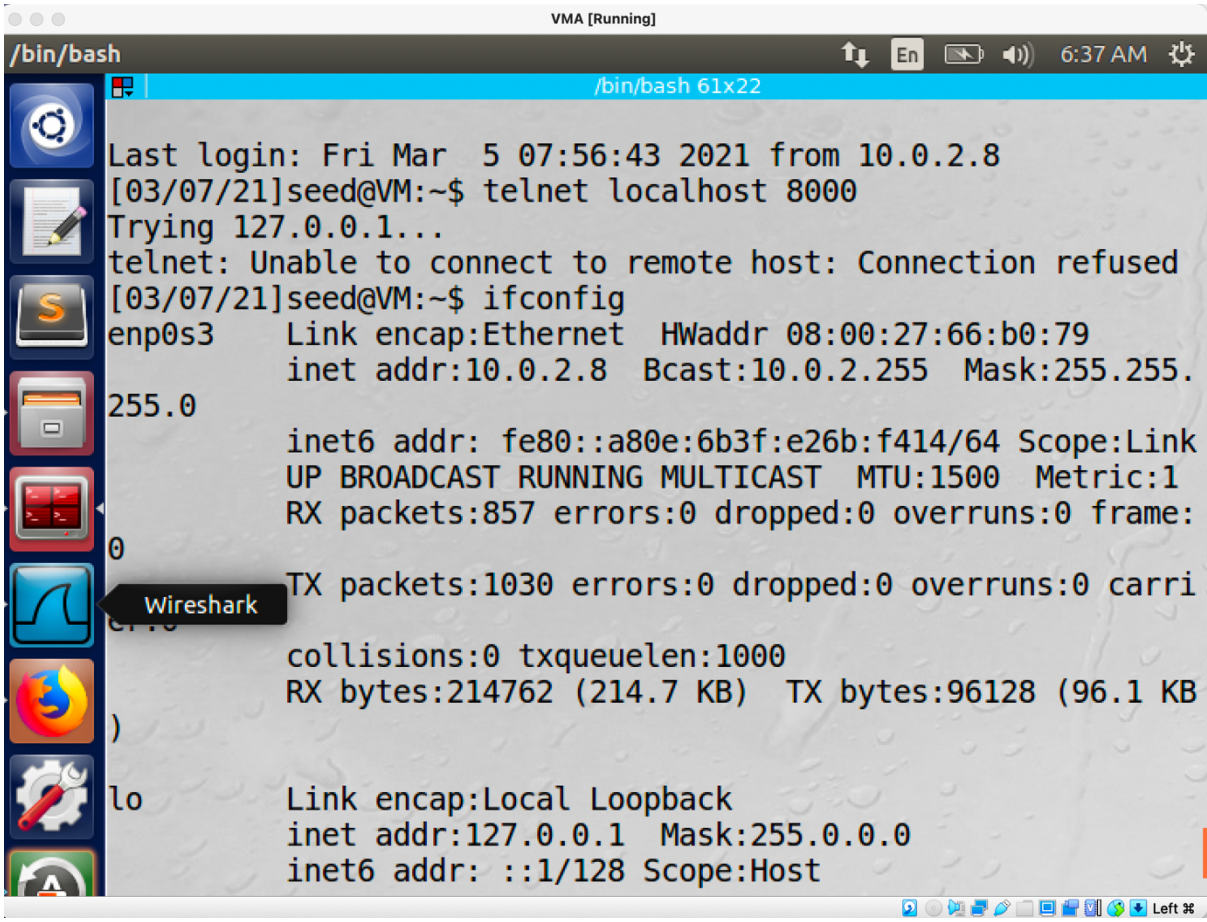
Result:



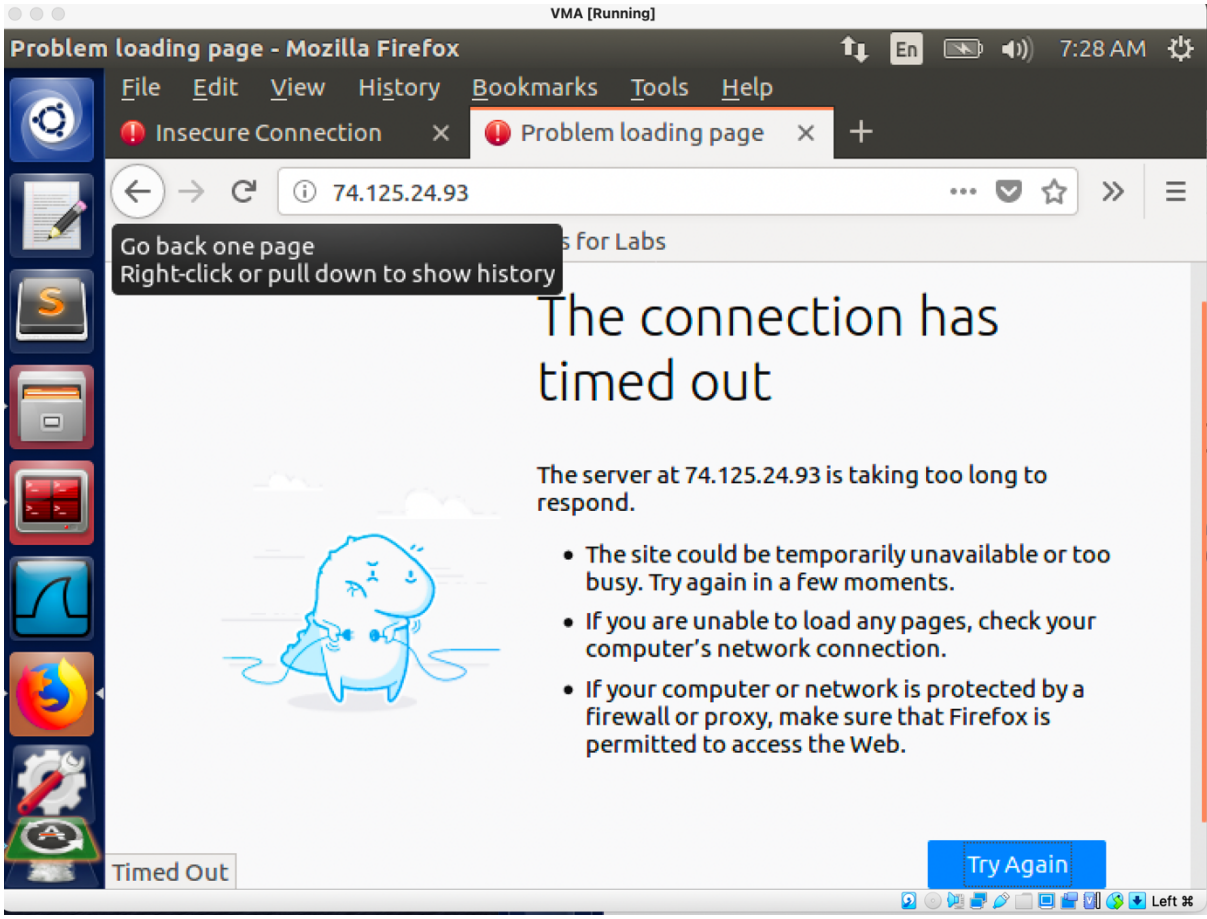
Task 3: Evading Egress Filtering

- 3a:

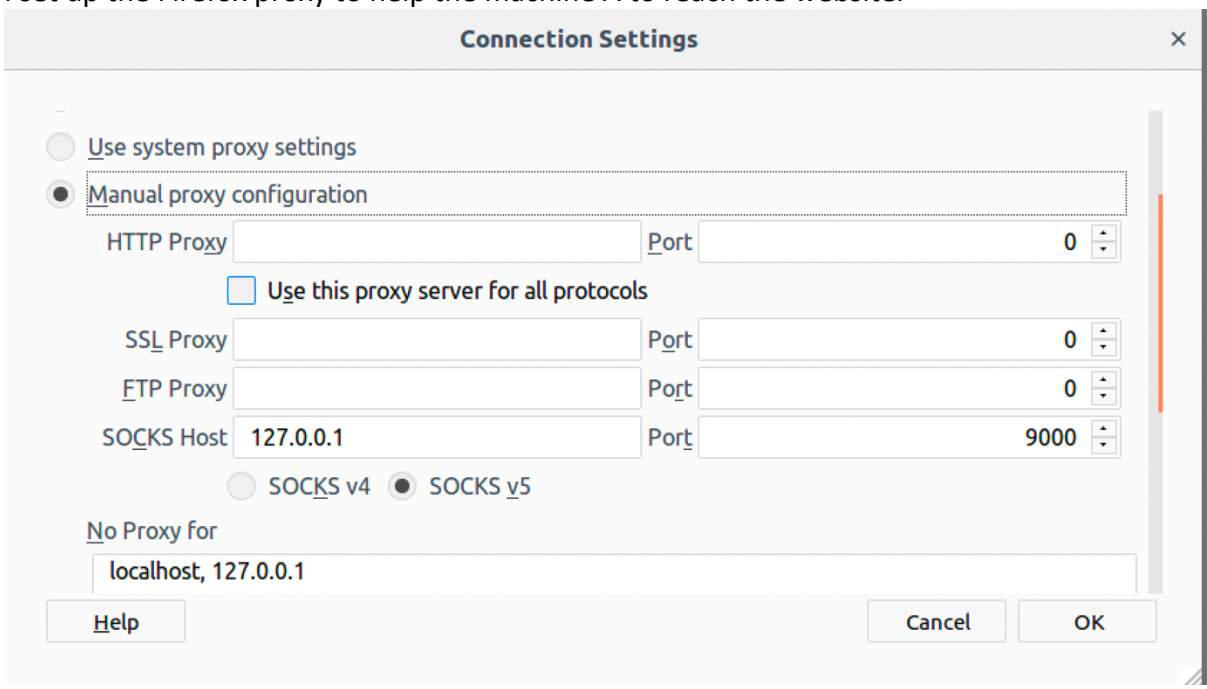
- Step1:
  I add a new iptables rule to block all the telnet request to 10.0.2.8 which is machine B by using the command “sudo iptables -A OUTPUT -p tcp -d 10.0.2.8 –dport 23 -j DROP”. After this new rule is added the telnet command will not work.
- Step2:
  In order to contact machine B, I use the command “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:
- Result:



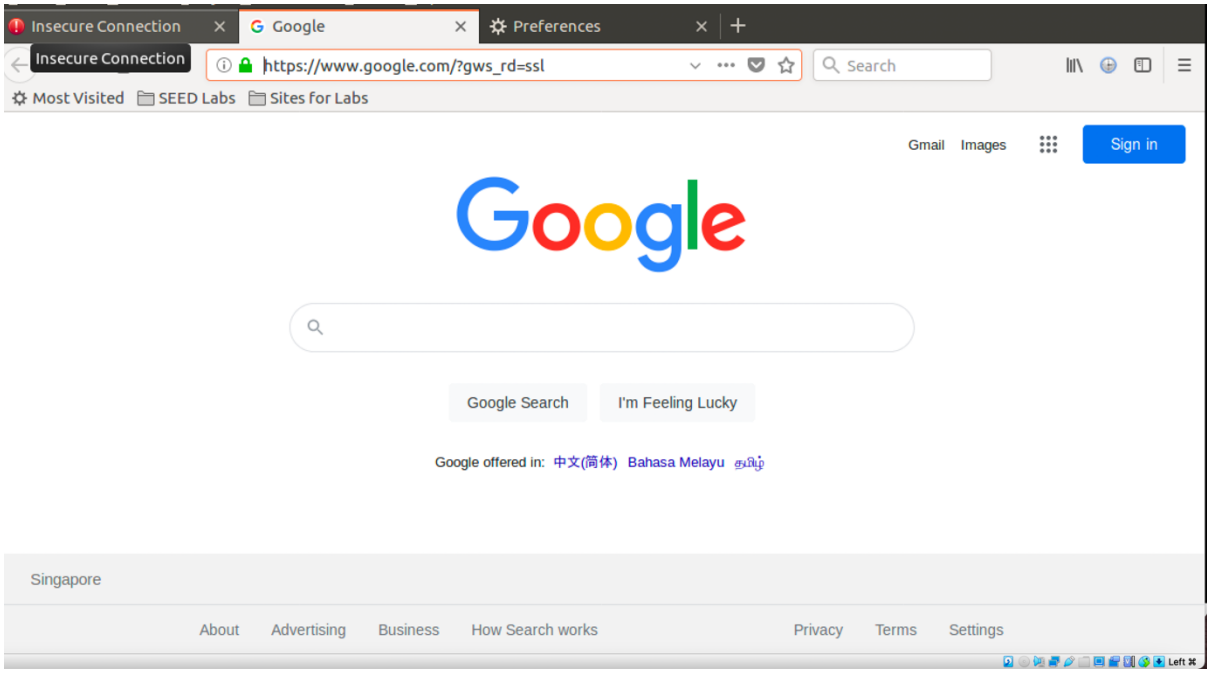
- 3b:
  - Step1:
    I use the command “sudo iptables -A OUTPUT -p tcp -d 74.125.24.93 –dport 80 -j DROP” to add a new iptables rule to block the machine A to visit 74.125.24.93 which is one of the ip address of google. After the rule is added, machine A is not able to visit that website.



- Step2:
I set up the Firefox proxy to help the machine A to reach the website.



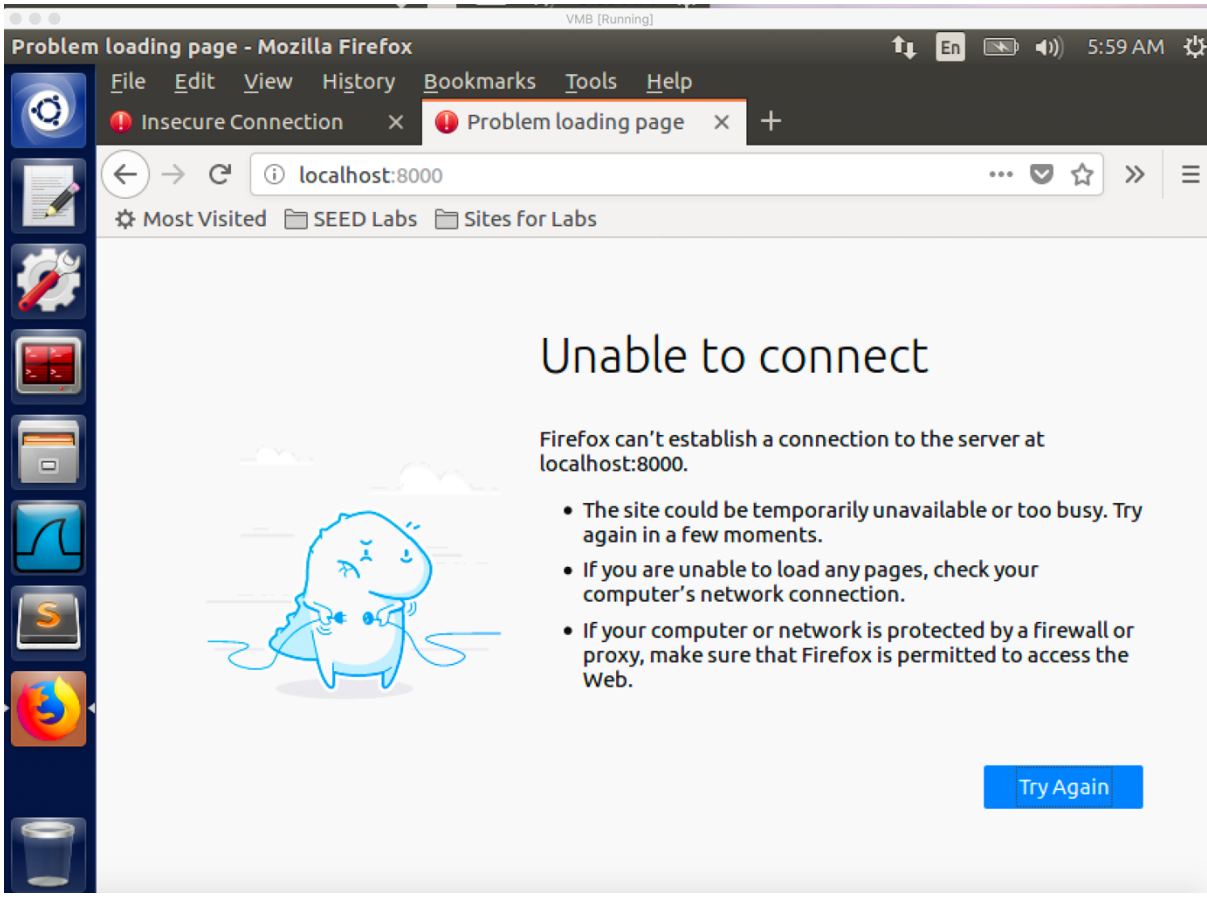
After I set up the proxy, machine A are able to reach the google website.
Result:



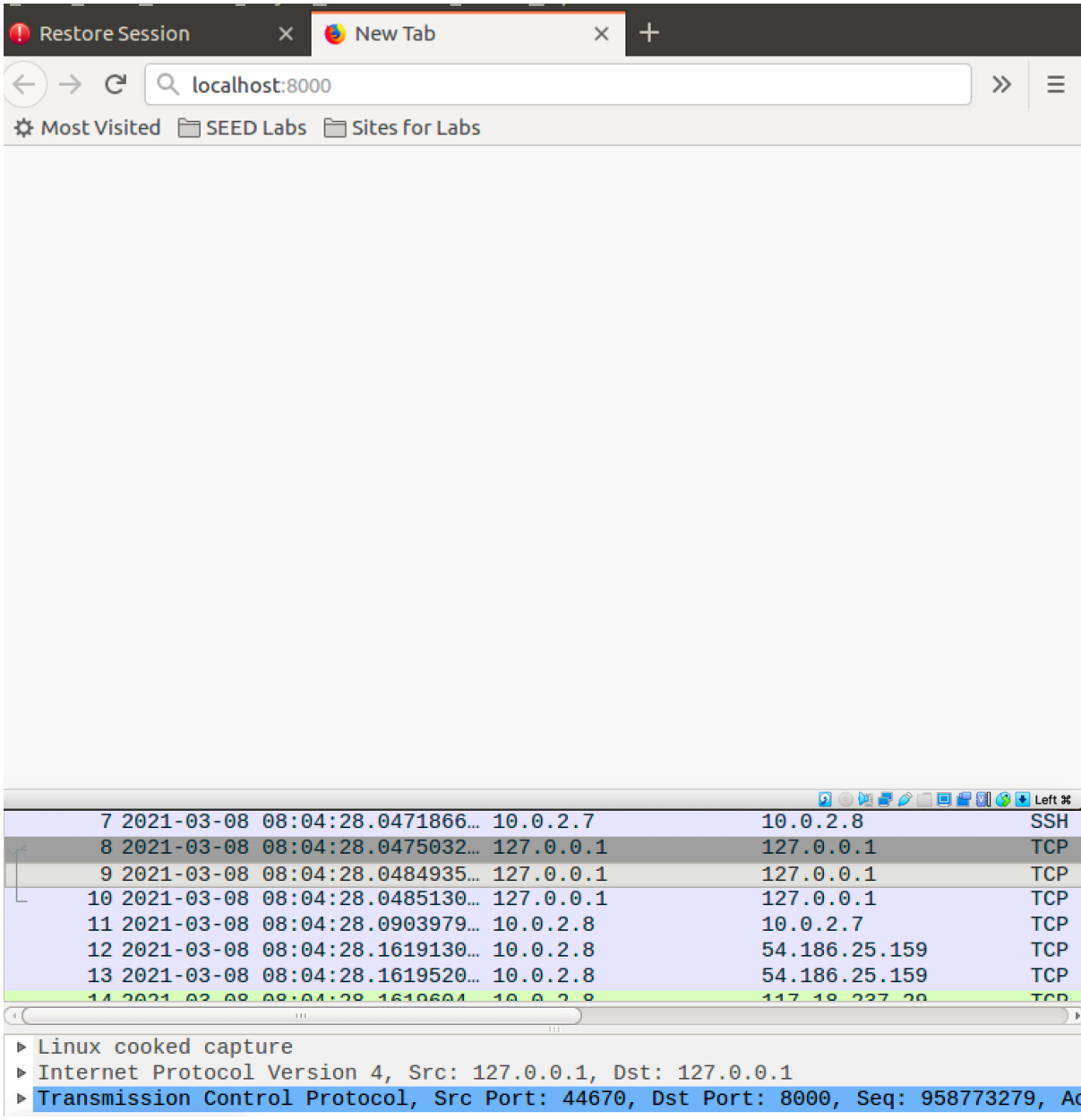
- Task4: Evading Ingress Filtering
  - Step1:
    Initially, I added two iptables rules to block machine B from accessing machine A's port 80 and 22.

```
[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             anywhere
 tcp dpt:http
DROP       tcp  --  anywhere             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.