Firewall Lab

Machine A: 10.0.2.15

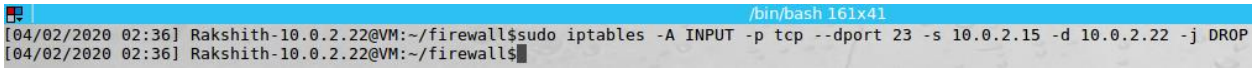
Machine B: 10.0.2.22

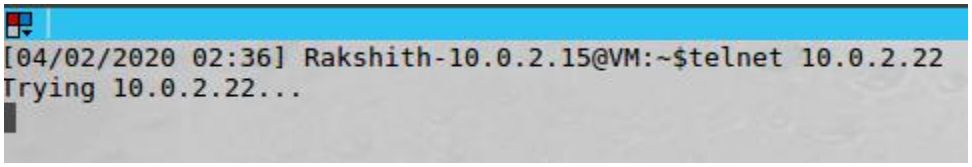
Machine C: 10.0.2.10

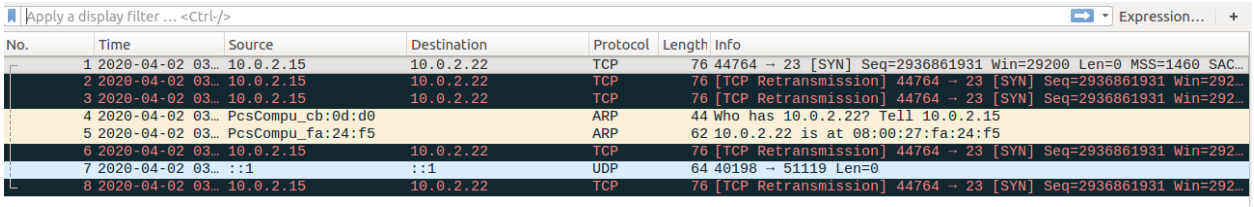
Task 1: Using Firewall

Prevent A from doing telnet to Machine B:

We set iptables rule in the input chain of machine B, the rule drops all TCP traffic on port 23 coming from source 10.0.2.15 towards 10.0.2.22. As seen in the output 10.0.2.15 cannot telnet to 10.0.2.22.

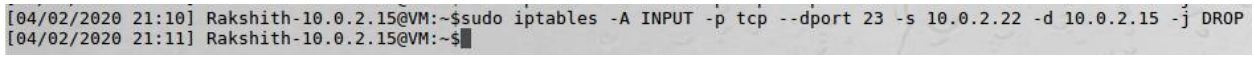


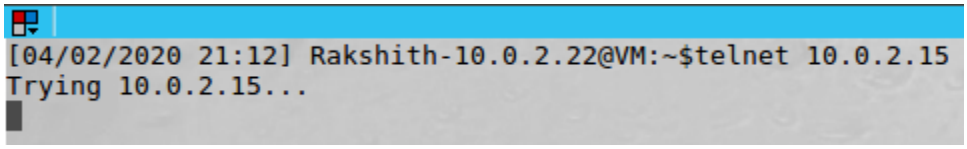


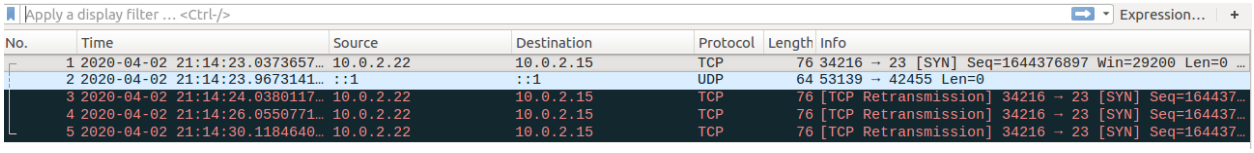


Prevent B from doing telnet to Machine A:

Similar to previous task we have set a rule on the Input chain of machine A, so all TCP traffic on port 23 from source Machine B will be dropped, and same can be seen in wireshark output.







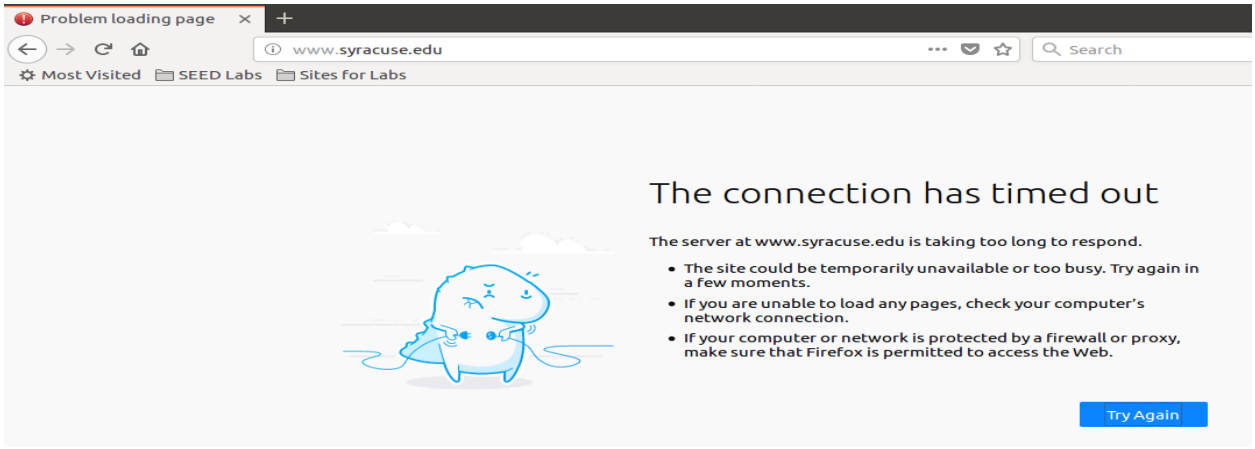
Prevent A from visiting an external website. You can choose any web site that you like to block, but keep in mind; some web servers have multiple IP addresses.

Here we are trying to block the website www.syracuse.edu, this website does not use a CDN to get IP addresses from different range. So, we do a dig on the website and try to obtain the CIDR used. We will get IP addresses from the CIDR assigned / registered for this website whenever we try to connect this website. We will attempt to block this full CIDR block to see if we can block the access to Syracuse.edu.

```
[04/02/2020 03:10] Rakshith-10.0.2.15@VM:~$dig +short www.syracuse.edu
syracuse.edu.
128.230.18.198
[04/02/2020 03:11] Rakshith-10.0.2.15@VM:~$whois 128.230.18.198 | grep -i CIDR
CIDR:           128.230.0.0/16
[04/02/2020 03:11] Rakshith-10.0.2.15@VM:~$
```

As seen above the CIDR obtained is 128.230.0.0/16. We add a iptables filter in the output chain of machine A blocking all outgoing traffic towards 128.230.0.0/16. As we can see in the output the page doesn't load, hence our task is achieved.

```
[04/02/2020 18:48] Rakshith-10.0.2.15@VM:~$sudo iptables -A OUTPUT -d 128.230.0.0/16 -j DROP
[04/02/2020 18:50] Rakshith-10.0.2.15@VM:~$
```




```
[04/02/2020 03:18] Rakshith-10.0.2.15@VM:~$wget www.syracuse.edu
--2020-04-02 03:23:17--  http://www.syracuse.edu/
Resolving www.syracuse.edu (www.syracuse.edu)... 128.230.18.198
Connecting to www.syracuse.edu (www.syracuse.edu)|128.230.18.198|:80... failed: Connection timed out.
Retrying.

--2020-04-02 03:25:33--  (try: 2)  http://www.syracuse.edu/
Connecting to www.syracuse.edu (www.syracuse.edu)|128.230.18.198|:80...
```

56	2020-04-02 03…	10.0.2.15	8.8.8.8	DNS	78 Standard query 0x7c8b A www.syracuse.edu
57	2020-04-02 03…	8.8.8.8	10.0.2.15	DNS	108 Standard query response 0x7c8b A www.syracuse.edu CNAME sy…
58	2020-04-02 03…	10.0.2.15	8.8.8.8	DNS	78 Standard query 0x972f AAAA www.syracuse.edu
59	2020-04-02 03…	8.8.8.8	10.0.2.15	DNS	141 Standard query response 0x972f AAAA www.syracuse.edu CNAME…
60	2020-04-02 03…	23.193.216.83	10.0.2.15	TLSv1.2	87 Encrypted Alert
61	2020-04-02 03…	10.0.2.15	23.193.216.83	TCP	56 33612 → 443 [ACK] Seq=2850722093 Ack=135748 Win=64240 Len=0
62	2020-04-02 03…	23.193.216.83	10.0.2.15	TCP	62 443 → 33612 [FIN, ACK] Seq=135748 Ack=2850722093 Win=32526…
63	2020-04-02 03…	10.0.2.15	23.193.216.83	TLSv1.2	102 Application Data
64	2020-04-02 03…	10.0.2.15	23.193.216.83	TCP	56 33612 → 443 [FIN, ACK] Seq=2850722139 Ack=135749 Win=64240…
65	2020-04-02 03…	23.193.216.83	10.0.2.15	TCP	62 443 → 33612 [ACK] Seq=135749 Ack=2850722140 Win=32479 Len=0
66	2020-04-02 03…	10.0.2.15	8.8.8.8	DNS	88 Standard query 0xb1bb A tiles.services.mozilla.com
67	2020-04-02 03…	::1	::1	UDP	64 40198 → 51119 Len=0
68	2020-04-02 03…	8.8.8.8	10.0.2.15	DNS	169 Standard query response 0xb1bb No such name A tiles.servic…
69	2020-04-02 03…	10.0.2.15	8.8.8.8	DNS	88 Standard query 0x498c AAAA tiles.services.mozilla.com
70	2020-04-02 03…	8.8.8.8	10.0.2.15	DNS	169 Standard query response 0x498c No such name AAAA tiles.ser…
71	2020-04-02 03…	10.0.2.15	8.8.8.8	DNS	105 Standard query 0xa1f1 A tiles.services.mozilla.com.fios-ro…
72	2020-04-02 03…	8.8.8.8	10.0.2.15	DNS	180 Standard query response 0xa1f1 No such name A tiles.servic…
73	2020-04-02 03…	10.0.2.15	8.8.8.8	DNS	105 Standard query 0x1574 AAAA tiles.services.mozilla.com.fios…
74	2020-04-02 03…	8.8.8.8	10.0.2.15	DNS	180 Standard query response 0x1574 No such name AAAA tiles.ser…
75	2020-04-02 03…	10.0.2.15	8.8.8.8	DNS	88 Standard query 0xf33c A tiles.services.mozilla.com
76	2020-04-02 03…	8.8.8.8	10.0.2.15	DNS	169 Standard query response 0xf33c No such name A tiles.servic…
77	2020-04-02 03…	10.0.2.15	8.8.8.8	DNS	88 Standard query 0x1778 AAAA tiles.services.mozilla.com
78	2020-04-02 03…	8.8.8.8	10.0.2.15	DNS	169 Standard query response 0x1778 No such name AAAA tiles.ser…
79	2020-04-02 03…	10.0.2.15	8.8.8.8	DNS	105 Standard query 0xa122 A tiles.services.mozilla.com.fios-ro…

Task 2: Implementing a Simple Firewall

As seen above, iptables is a packet filtering firewall it inspects packets and then makes the decision based on the policies set by administrator. All these changes are facilitated by the changes in the kernel, in the past kernel had to be modified and rebuilt for these changes to take place. Now we use mechanisms called Loadable Kernel Module (LKM) and Netfilter to facilitate the changes in kernel without rebuilding the kernel. LKM allows us to add a new module to the kernel at the runtime.

Netfilter is designed to facilitate the manipulation of packets by authorized users. Netfilter achieves this goal by implementing a number of hooks in the Linux kernel. These hooks are inserted into various places, including the packet incoming and outgoing paths.

Prevent Machine A from doing telnet to Machine B

Below is the code where we allow Netfilter to attach to NF_INET_LOCAL_IN filter this helps us to block all the Ingress telnet traffic between machine A and machine B. This program runs on machine B, hence machine A cannot telnet to machine B as the kernel blocks incoming telnet packets to machine B. We create a makefile to generate kernel module. Once the kernel module is generated we use insmod to add the kernel module to run time of kernel. This helps kernel to apply inspections on packet.

/bin/bash 161x39

```
#include <linux/kernel.h>
#include <linux/module.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->daddr==htonl(167772694) && iph->saddr==htonl(167772687)) {
    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_LOCAL_IN;
//      telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.pf = PF_INET;
```

```
        // Register the hook.
        nf_register_hook(&telnetFilterHook);
        return 0;
}

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

module_init(setUpFilter);
module_exit(removeFilter);

MODULE_LICENSE("GPL");
```

```
[04/02/2020 04:35] Rakshith-10.0.2.22@VM:~/firewall$cat Makefile
obj-m += minifirewall.o
all:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

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

[04/02/2020 04:35] Rakshith-10.0.2.22@VM:~/firewall$
```

```
[04/02/2020 04:31] Rakshith-10.0.2.22@VM:~/firewall$make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/firewall modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/firewall/minifirewall.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/firewall/minifirewall.mod.o
  LD [M]  /home/seed/firewall/minifirewall.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/02/2020 04:31] Rakshith-10.0.2.22@VM:~/firewall$ls
Makefile  minifirewall.c  minifirewall.ko  minifirewall.mod.c  minifirewall.mod.o  minifirewall.o  modules.order  Module.symvers
[04/02/2020 04:31] Rakshith-10.0.2.22@VM:~/firewall$sudo insmod minifirewall.ko
[sudo] password for seed:
```

```
[04/02/2020 04:32] Rakshith-10.0.2.22@VM:~/firewall$dmesg -T | grep -i Telnet
[Thu Apr  2 04:32:41 2020] Registering a Telnet filter.
[04/02/2020 04:34] Rakshith-10.0.2.22@VM:~/firewall$
```

```
[Thu Apr  2 03:31:13 2020] Netfilter messages via NETLINK v0.30.
[Thu Apr  2 04:32:41 2020] Registering a Telnet filter.
[04/02/2020 04:32] Rakshith-10.0.2.22@VM:~/firewall$
```

```
[04/02/2020 04:47] Rakshith-10.0.2.15@VM:~$telnet 10.0.2.22
Trying 10.0.2.22...
```

No.	Time	Source	Destination	Protocol	Length	Info
1	2020-04-02 04…	::1	::1	UDP	64	40198 → 51119 Len=0
2	2020-04-02 04…	10.0.2.15	10.0.2.22	TCP	76	44814 → 23 [SYN] Seq=310670462 Win=29200 Len=0 MSS=1460 SACK…
3	2020-04-02 04…	10.0.2.15	10.0.2.22	TCP	76	[TCP Retransmission] 44814 → 23 [SYN] Seq=310670462 Win=2920…
4	2020-04-02 04…	10.0.2.15	10.0.2.22	TCP	76	[TCP Retransmission] 44814 → 23 [SYN] Seq=310670462 Win=2920…
5	2020-04-02 04…	PcsCompu_cb:0d:d0		ARP	44	Who has 10.0.2.22? Tell 10.0.2.15
6	2020-04-02 04…	PcsCompu_fa:24:f5		ARP	62	10.0.2.22 is at 08:00:27:fa:24:f5
7	2020-04-02 04…	10.0.2.15	10.0.2.22	TCP	76	[TCP Retransmission] 44814 → 23 [SYN] Seq=310670462 Win=2920…

Reverting changes:

Once we remove the kernel module machine A can easily telnet to machine B.

```
[04/02/2020 04:49] Rakshith-10.0.2.22@VM:~/firewall$sudo rmmod minifirewall.ko
[04/02/2020 04:51] Rakshith-10.0.2.22@VM:~/firewall$
```

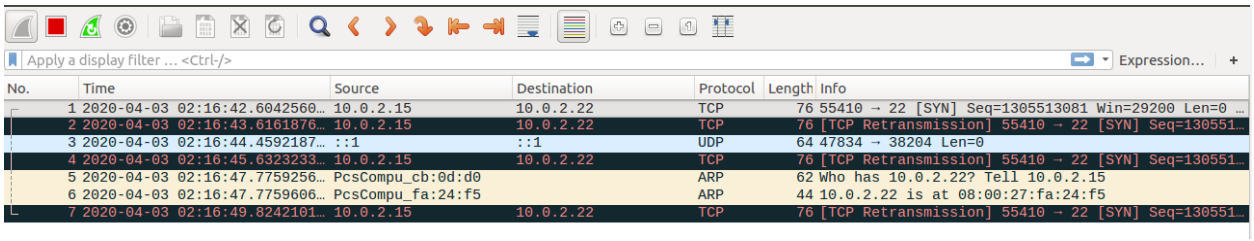
```
[04/02/2020 04:50] Rakshith-10.0.2.15@VM:~$telnet 10.0.2.22
Trying 10.0.2.22...
Connected to 10.0.2.22.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: ^CConnection closed by foreign host.
[04/02/2020 04:51] Rakshith-10.0.2.15@VM:~$
```

Prevent Machine A from doing SSH to Machine B:

We use the same hook, this time we change the port number to 22 (port number of SSH), we then try to do SSH from Machine A to Machine B and kernel blocks all the packets. Remember we have to convert the IP address from dotted decimal format to integer format.

```
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(22)&& iph->daddr==htonl(167772694) && iph->saddr==htonl(167772687)) {
    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_LOCAL_IN;
//      telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.pf = PF_INET;
```

```
[04/03/2020 02:06] Rakshith-10.0.2.15@VM:~$ssh 10.0.2.22
```



Prevent Machine B from doing Telnet to Machine A:

```
/bin/bash 161x41
#include <linux/kernel.h>
#include <linux/module.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->daddr==htonl(167772687) && iph->saddr==htonl(167772694)) {
    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_LOCAL_IN;
//      telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.pf = PF_INET;
        telnetFilterHook.priority = NF_IP_PRI_FIRST;
```

We use the same firewall program in the machine A, we change the source and destination IP to the same code with port number of telnet (port 23). We can see that machine B is unable to telnet to machine A.

```
[04/03/2020 02:23] Rakshith-10.0.2.15@VM:~/firewall$make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/firewall modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/firewall/minifirewall.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/firewall/minifirewall.mod.o
  LD [M]  /home/seed/firewall/minifirewall.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/03/2020 02:23] Rakshith-10.0.2.15@VM:~/firewall$ls
Makefile  minifirewall.c  minifirewall.ko  minifirewall.mod.c  minifirewall.mod.o  minifirewall.o  modules.order  Module.symvers
[04/03/2020 02:23] Rakshith-10.0.2.15@VM:~/firewall$sudo insmod minifirewall.ko
[sudo] password for seed:
[04/03/2020 02:23] Rakshith-10.0.2.15@VM:~/firewall$
```

```
[04/03/2020 02:25] Rakshith-10.0.2.22@VM:~/firewall$telnet 10.0.2.15
Trying 10.0.2.15...
```

8	2020-04-03 02…	10.0.2.22	10.0.2.15	TCP	76 40370 → 23 [SYN] Seq=2889085501 Win=29200 Len=0 MSS=1460 SAC…
9	2020-04-03 02…	10.0.2.22	10.0.2.15	TCP	76 [TCP Retransmission] 40370 → 23 [SYN] Seq=2889085501 Win=292…
10	2020-04-03 02…	10.0.2.22	10.0.2.15	TCP	76 [TCP Retransmission] 40370 → 23 [SYN] Seq=2889085501 Win=292…
11	2020-04-03 02…	10.0.2.22	10.0.2.15	TCP	76 [TCP Retransmission] 40370 → 23 [SYN] Seq=2889085501 Win=292…
12	2020-04-03 02…	::1	::1	UDP	64 60138 → 56544 Len=0
13	2020-04-03 02…	10.0.2.22	10.0.2.15	TCP	76 [TCP Retransmission] 40370 → 23 [SYN] Seq=2889085501 Win=292…
14	2020-04-03 02…	10.0.2.22	10.0.2.15	TCP	76 [TCP Retransmission] 40370 → 23 [SYN] Seq=2889085501 Win=292…
15	2020-04-03 02…	::1	::1	UDP	64 60138 → 56544 Len=0

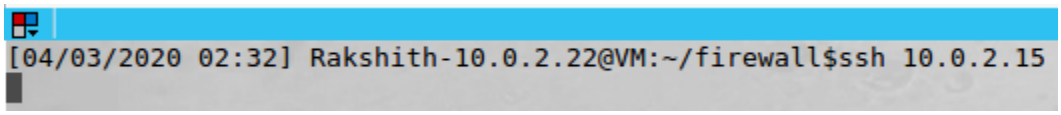
```
[Fri Apr  3 02:23:45 2020] Registering a Telnet filter.
[Fri Apr  3 02:25:58 2020] Dropping telnet packet to 10.0.2.15
[Fri Apr  3 02:25:59 2020] Dropping telnet packet to 10.0.2.15
[Fri Apr  3 02:26:01 2020] Dropping telnet packet to 10.0.2.15
[Fri Apr  3 02:26:05 2020] Dropping telnet packet to 10.0.2.15
[Fri Apr  3 02:26:14 2020] Dropping telnet packet to 10.0.2.15
[Fri Apr  3 02:26:30 2020] Dropping telnet packet to 10.0.2.15
[Fri Apr  3 02:27:03 2020] Dropping telnet packet to 10.0.2.15
[04/03/2020 02:28] Rakshith-10.0.2.15@VM:~/firewall$
```

Prevent Machine B from doing SSH to Machine A:

```c
#include <linux/kernel.h>
#include <linux/module.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(22) && iph->daddr==htonl(167772687) && iph->saddr==htonl(167772694)) {
    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_LOCAL_IN;
//      telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.pf = PF_INET;
```

We use the same code, we just change the port number to 22. We can see that machine B cannot SSH to machine A.

```
[04/03/2020 02:31] Rakshith-10.0.2.15@VM:~/firewall$make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/firewall modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/firewall/minifirewall.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/firewall/minifirewall.mod.o
  LD [M]  /home/seed/firewall/minifirewall.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/03/2020 02:31] Rakshith-10.0.2.15@VM:~/firewall$sudo insmod minifirewall.ko
[04/03/2020 02:31] Rakshith-10.0.2.15@VM:~/firewall$
```

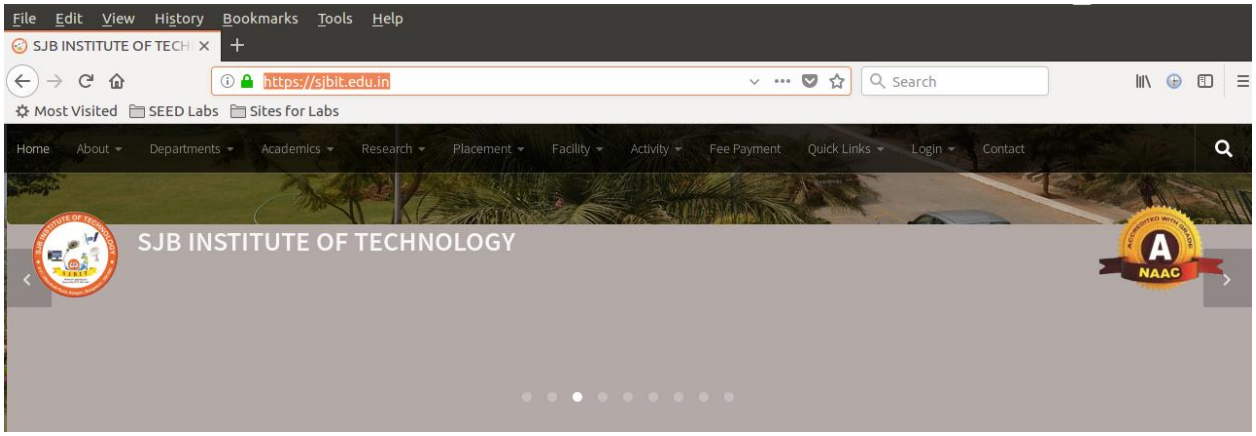


4	2020-04-03 02…	10.0.2.22	10.0.2.15	TCP	76 47586 → 22 [SYN] Seq=1236005549 Win=29200 Len=0 MSS=1460 SAC…
5	2020-04-03 02…	10.0.2.22	10.0.2.15	TCP	76 [TCP Retransmission] 47586 → 22 [SYN] Seq=1236005549 Win=292…
6	2020-04-03 02…	10.0.2.22	10.0.2.15	TCP	76 [TCP Retransmission] 47586 → 22 [SYN] Seq=1236005549 Win=292…
7	2020-04-03 02…	::1	::1	UDP	64 60138 → 56544 Len=0
8	2020-04-03 02…	PcsCompu_fa:24:f5		ARP	62 Who has 10.0.2.15? Tell 10.0.2.22
9	2020-04-03 02…	PcsCompu_cb:0d:d0		ARP	44 10.0.2.15 is at 08:00:27:cb:0d:d0
10	2020-04-03 02…	10.0.2.22	10.0.2.15	TCP	76 [TCP Retransmission] 47586 → 22 [SYN] Seq=1236005549 Win=292…

Block external website access from Machine A:

For this task we change the hook, we use NF_INET_POST_ROUTING, this hook blocks outgoing traffic towards. Here I am trying to block www.sjbit.edu.in, we use dig to identify the IP address of this website, this page has only one IP address and is static, we then block all traffic going towards this website by specifying the IP address in the filter. As we can see once the filter is hooked the web page becomes unreachable.

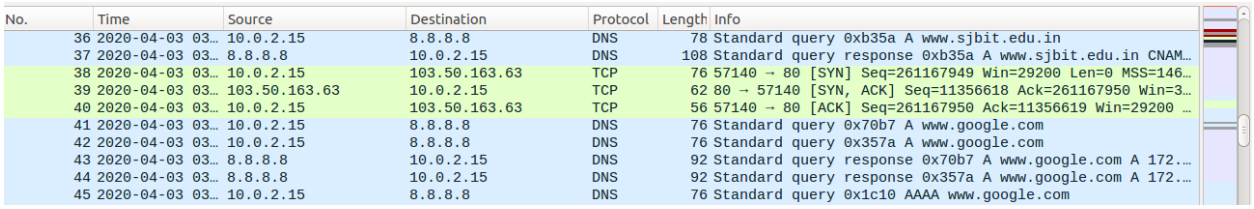
```
[04/03/2020 02:33] Rakshith-10.0.2.15@VM:~/firewall$dig +short www.sjbit.edu.in
sjbit.edu.in.
103.50.163.63
[04/03/2020 02:50] Rakshith-10.0.2.15@VM:~/firewall$
```



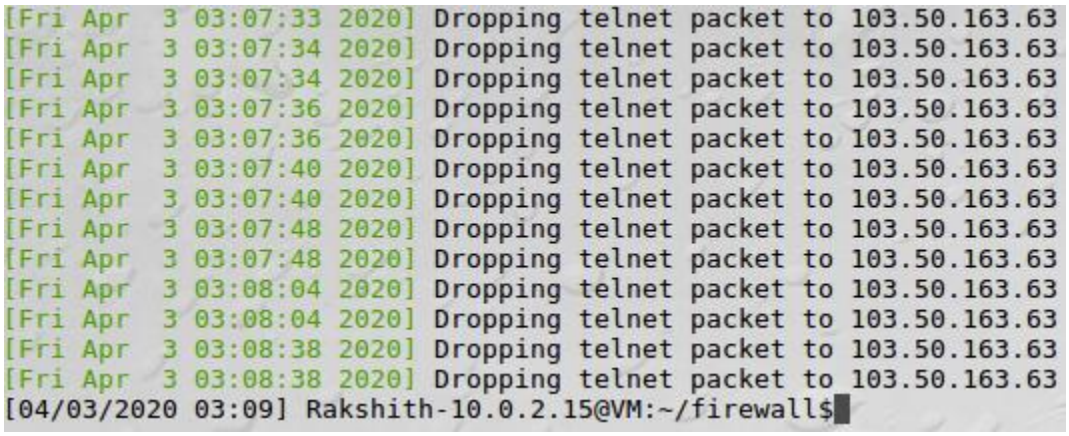

```
[04/03/2020 02:58] Rakshith-10.0.2.15@VM:~/firewall$make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/firewall modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/firewall/minifirewall.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/firewall/minifirewall.mod.o
  LD [M]  /home/seed/firewall/minifirewall.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/03/2020 02:59] Rakshith-10.0.2.15@VM:~/firewall$ls
Makefile  minifirewall.c  minifirewall.ko  minifirewall.mod.c  minifirewall.mod.o  minifirewall.o  modules.order  Module.symvers
[04/03/2020 02:59] Rakshith-10.0.2.15@VM:~/firewall$sudo insmod minifirewall.ko
[04/03/2020 02:59] Rakshith-10.0.2.15@VM:~/firewall$
```

/bin/bash 161x39

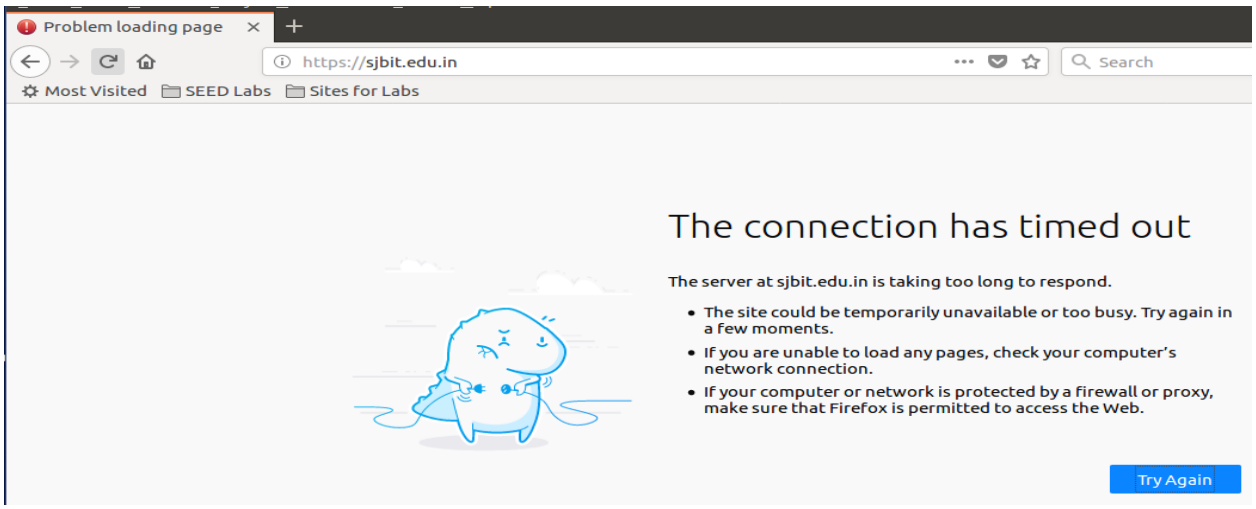
```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/ip.h>
#include <linux/tcp.h>

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(443) && iph->daddr==htonl(1731371839) && iph->saddr==htonl(167772687)) {
    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_LOCAL_IN;
        telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.pf = PF_INET;
```



No.	Time	Source	Destination	Protocol	Length	Info
36	2020-04-03 03…	10.0.2.15	8.8.8.8	DNS	78	Standard query 0xb35a A www.sjbit.edu.in
37	2020-04-03 03…	8.8.8.8	10.0.2.15	DNS	108	Standard query response 0xb35a A www.sjbit.edu.in CNAM…
38	2020-04-03 03…	10.0.2.15	103.50.163.63	TCP	76	57140 → 80 [SYN] Seq=261167949 Win=29200 Len=0 MSS=146…
39	2020-04-03 03…	103.50.163.63	10.0.2.15	TCP	62	80 → 57140 [SYN, ACK] Seq=11356618 Ack=261167950 Win=3…
40	2020-04-03 03…	10.0.2.15	103.50.163.63	TCP	56	57140 → 80 [ACK] Seq=261167950 Ack=11356619 Win=29200 …
41	2020-04-03 03…	10.0.2.15	8.8.8.8	DNS	76	Standard query 0x70b7 A www.google.com
42	2020-04-03 03…	10.0.2.15	8.8.8.8	DNS	76	Standard query 0x357a A www.google.com
43	2020-04-03 03…	8.8.8.8	10.0.2.15	DNS	92	Standard query response 0x70b7 A www.google.com A 172.…
44	2020-04-03 03…	8.8.8.8	10.0.2.15	DNS	92	Standard query response 0x357a A www.google.com A 172.…
45	2020-04-03 03…	10.0.2.15	8.8.8.8	DNS	76	Standard query 0x1c10 AAAA www.google.com



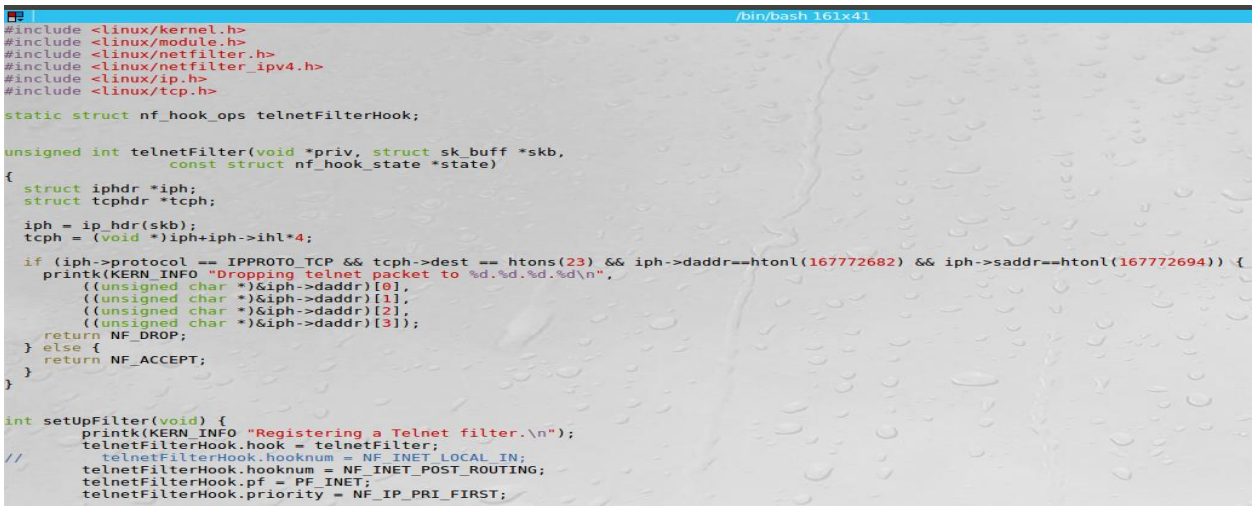
```
[Fri Apr  3 03:07:33 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:07:34 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:07:34 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:07:36 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:07:36 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:07:40 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:07:40 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:07:48 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:07:48 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:08:04 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:08:04 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:08:38 2020] Dropping telnet packet to 103.50.163.63
[Fri Apr  3 03:08:38 2020] Dropping telnet packet to 103.50.163.63
[04/03/2020 03:09] Rakshith-10.0.2.15@VM:~/firewall$
```



Blocking Egress connections:

Block telnet connection between Machine B and Machine C:

Here we use the hook NF_INET_POST_ROUTING to block outgoing telnet connections of Machine B towards Machine C. We have to convert the IP address from dotted decimal format to integer.



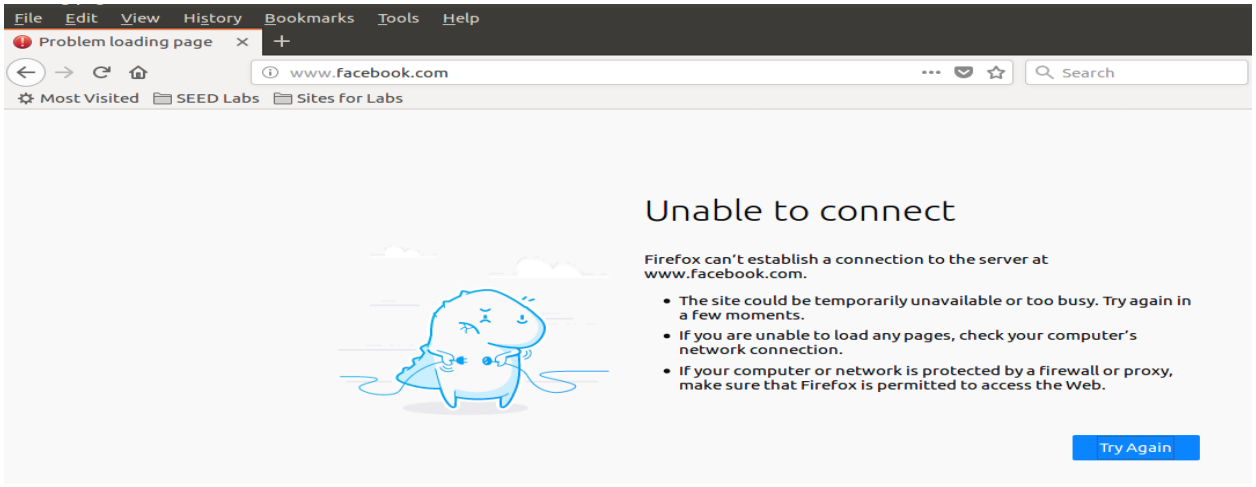

```
[04/03/2020 22:47] Rakshith-10.0.2.22@VM:~/firewall$make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed/firewall modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M]  /home/seed/firewall/minifirewall.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/seed/firewall/minifirewall.mod.o
  LD [M]  /home/seed/firewall/minifirewall.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/03/2020 22:47] Rakshith-10.0.2.22@VM:~/firewall$sudo insmod minifirewall.ko
[sudo] password for seed:
[04/03/2020 22:47] Rakshith-10.0.2.22@VM:~/firewall$ls
Makefile  minifirewall.c  minifirewall.ko  minifirewall.mod.c  minifirewall.mod.o  minifirewall.o  modules.order  Module.symvers
[04/03/2020 22:47] Rakshith-10.0.2.22@VM:~/firewall$telnet 10.0.2.10
Trying 10.0.2.10...
^C
[04/03/2020 22:47] Rakshith-10.0.2.22@VM:~/firewall$
```

Prevent Machine B from accessing www.facebook.com

We use the same hook, all egress connections towards www.facebook.com will be blocked. Facebook.com is load balanced between 31.13.66.35 and 31.13.71.36, so we try to block connections towards both the IPs on port 443.

/bin/bash 161x39

```c
#include <linux/kernel.h>
#include <linux/module.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(443) && (iph->daddr==htonl(520963876)||iph->daddr==htonl(520962595)) && iph->saddr==htonl(167772687)) {
    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_LOCAL_IN;
        telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.pf = PF_INET;
```



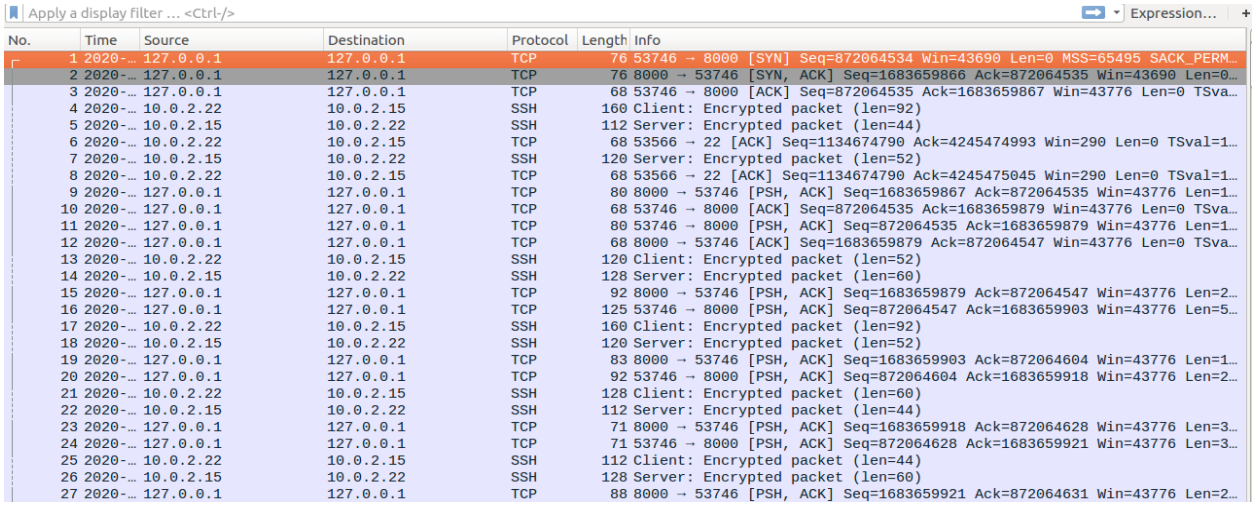
Task 3: Evading Egress Filtering:

In the previous task we have used Netfilter to block the egress traffic between Machine B and Machine C. Now we will use SSH tunneling mechanism to evade the egress filtering. We establish an SSH tunnel between machine B and machine A on port 8000. Once the tunnel is established all tunnel traffic will be encrypted and this allows packets to travel without inspection by kernel. When we telnet to localhost on port 8000 the ssh tunnel will allow us to send traffic from one end of the tunnel to the other end.

As seen in the wireshark output once we telnet to localhost on port 8000 since all telnet data is encypted, a telnet connection will be established between machine B and C, thus our egress filtering is bypassed.

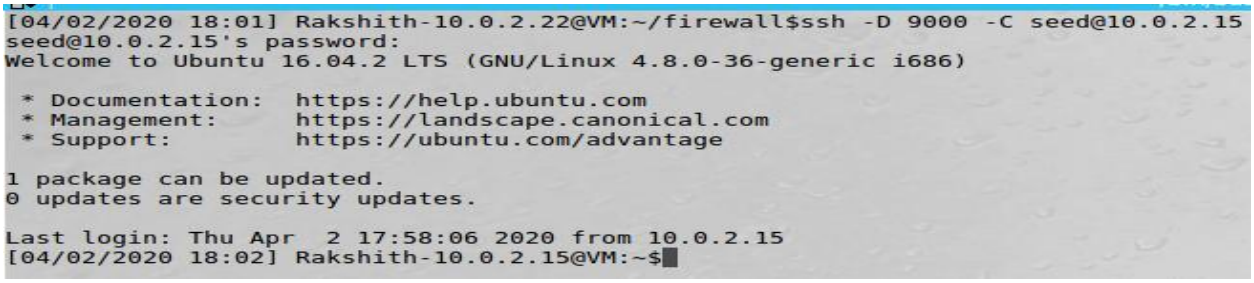
```
[04/03/2020 22:58] Rakshith-10.0.2.22@VM:~/firewall$ssh -L 8000:10.0.2.10:23 seed@10.0.2.15
seed@10.0.2.15's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

Last login: Fri Apr  3 22:54:54 2020 from 10.0.2.10
[04/03/2020 22:59] Rakshith-10.0.2.15@VM:~$
```

```
[04/03/2020 23:03] Rakshith-10.0.2.22@VM:~/firewall$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: Fri Apr  3 23:11:56 EDT 2020 from 10.0.2.15 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[04/03/2020 23:17] Rakshith-10.0.2.10@VM:~$
```

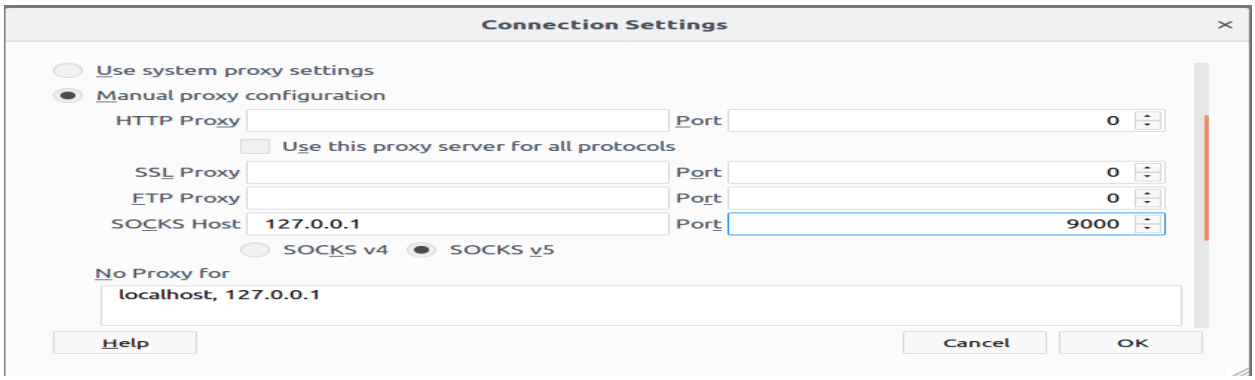


Evading filtering on www.facebook.com

Similar to previous task we are trying to evade filtering on facebook.com, since facebook.com will be managed by their own CDN IPs are dynamically assigned. So we cannot use a static forwarding as we used below. So we have to enable the dynamic port forwarding, when Machine B receives a packet from the tunnel, it will dynamically decide where it should forward the packet to based on the destination information of the packet.

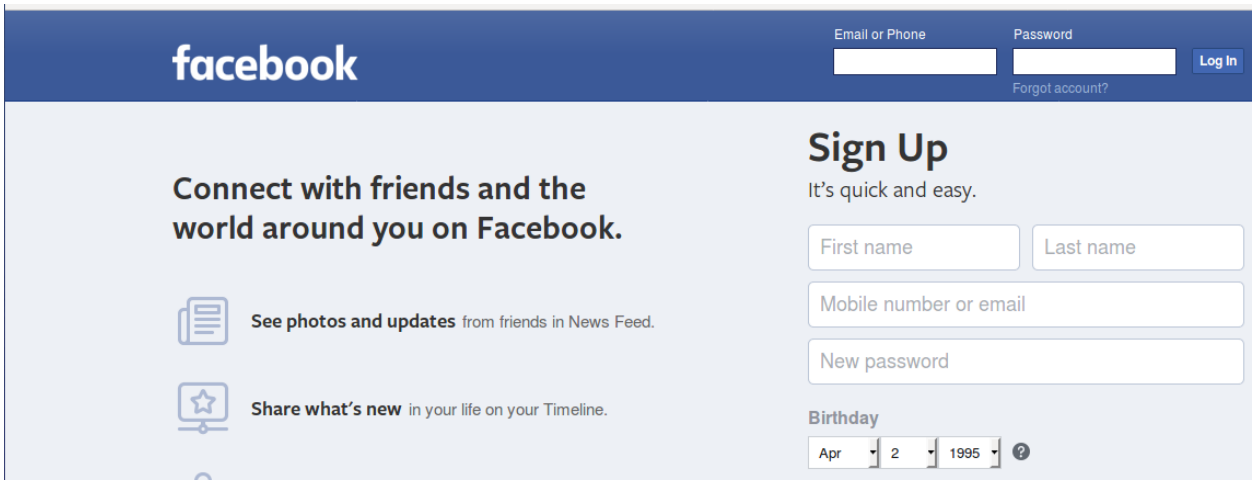


Similar to the telnet program, which connects localhost:9000, we need to ask Firefox to connect to localhost:9000 every time it needs to connect to a web server, so the traffic can go through our SSH tunnel. To achieve that, we can tell Firefox to uselocalhost:9000 as its proxy. To support dynamic port forwarding, we need a special type of proxy called SOCKS proxy, which is supported by most browsers.



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

We can successfully load the facebook page now. As seen in the wireshark output below all the tcp data will be handled by the proxy we configured, the packet information will then be encrypted which kernel cannot inspect and hence we are evading the filtering.

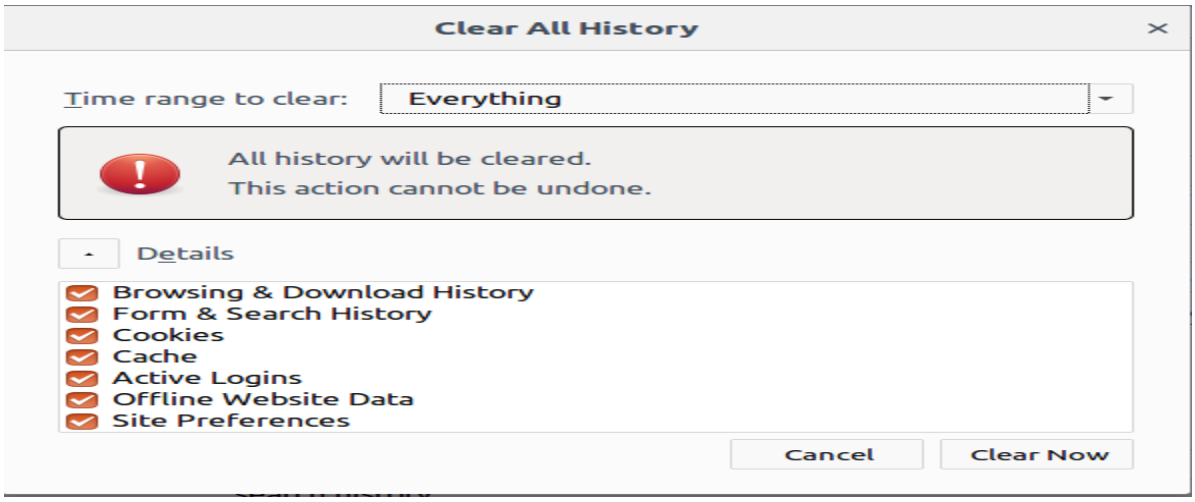


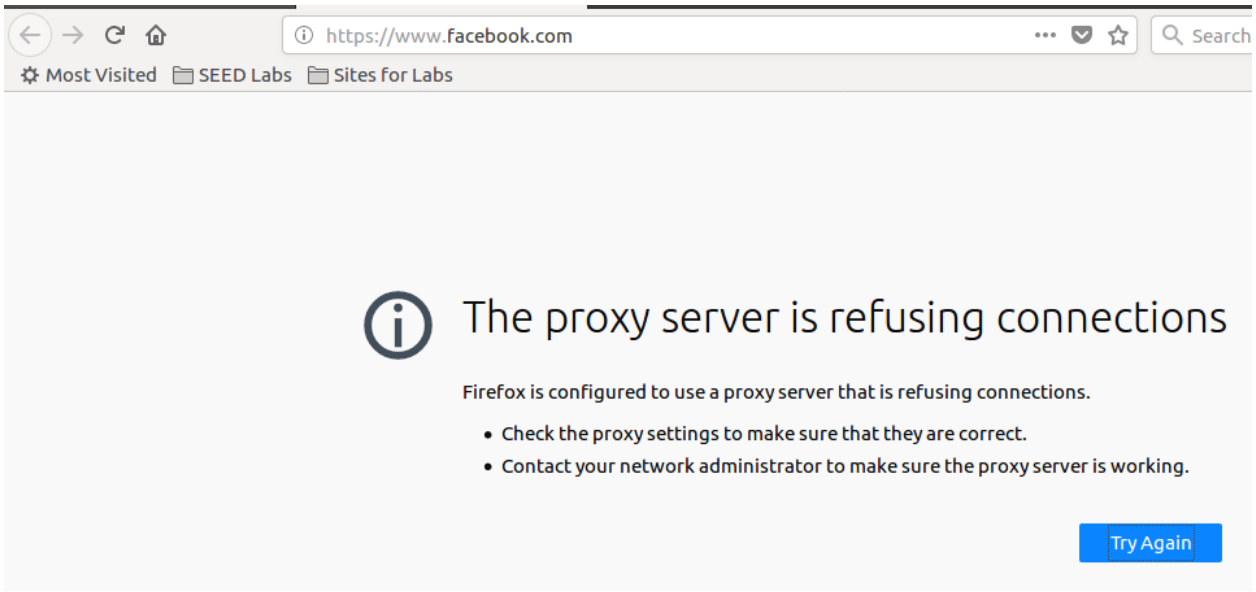
762	2020-04-02 18:07:11.3337439…	10.0.2.22	10.0.2.15	DNS	78	Standard query 0x2249 AAAA www.facebook.com
763	2020-04-02 18:07:11.3495227…	PcsCompu_a1:42:a9		ARP	44	Who has 34.209.18.179? Tell 192.168.60.1
764	2020-04-02 18:07:11.3496123…	PcsCompu_a1:42:a9		ARP	44	Who has 35.164.80.106? Tell 192.168.60.1
765	2020-04-02 18:07:11.3952316…	10.0.2.15	10.0.2.22	DNS	366	Standard query response 0x998c A www.facebook.co…
766	2020-04-02 18:07:12.3733555…	PcsCompu_a1:42:a9		ARP	44	Who has 35.164.80.106? Tell 192.168.60.1
767	2020-04-02 18:07:12.3734435…	PcsCompu_a1:42:a9		ARP	44	Who has 34.209.18.179? Tell 192.168.60.1
768	2020-04-02 18:07:12.9766536…	10.0.2.15	10.0.2.22	DNS	378	Standard query response 0x2249 AAAA www.facebook…
769	2020-04-02 18:07:12.9782772…	127.0.0.1	127.0.0.1	TCP	76	33690 → 9000 [SYN] Seq=3855907960 Win=43690 Len=…
770	2020-04-02 18:07:12.9783134…	127.0.0.1	127.0.0.1	TCP	76	9000 → 33690 [SYN, ACK] Seq=749397912 Ack=385590…
771	2020-04-02 18:07:12.9783480…	127.0.0.1	127.0.0.1	TCP	68	33690 → 9000 [ACK] Seq=3855907961 Ack=749397913 …
772	2020-04-02 18:07:12.9787469…	127.0.0.1	127.0.0.1	TCP	71	33690 → 9000 [PSH, ACK] Seq=3855907961 Ack=74939…
773	2020-04-02 18:07:12.9787742…	127.0.0.1	127.0.0.1	TCP	68	9000 → 33690 [ACK] Seq=749397913 Ack=3855907964 …
774	2020-04-02 18:07:12.9788778…	127.0.0.1	127.0.0.1	TCP	70	9000 → 33690 [PSH, ACK] Seq=749397913 Ack=385590…
775	2020-04-02 18:07:12.9791776…	127.0.0.1	127.0.0.1	TCP	68	33690 → 9000 [ACK] Seq=3855907964 Ack=749397915 …
776	2020-04-02 18:07:12.9792231…	127.0.0.1	127.0.0.1	TCP	78	33690 → 9000 [PSH, ACK] Seq=3855907964 Ack=74939…
777	2020-04-02 18:07:12.9795303…	10.0.2.22	10.0.2.15	SSHv2	120	Client: Encrypted packet (len=52)
778	2020-04-02 18:07:13.0061031…	10.0.2.15	10.0.2.22	SSHv2	112	Server: Encrypted packet (len=44)
779	2020-04-02 18:07:13.0061535…	10.0.2.22	10.0.2.15	TCP	68	55770 → 22 [ACK] Seq=3196196607 Ack=637359090 Wi…
780	2020-04-02 18:07:13.0065408…	127.0.0.1	127.0.0.1	TCP	78	9000 → 33690 [PSH, ACK] Seq=749397915 Ack=385590…
781	2020-04-02 18:07:13.0496721…	127.0.0.1	127.0.0.1	TCP	68	33690 → 9000 [ACK] Seq=3855907974 Ack=749397925 …
782	2020-04-02 18:07:13.0542836…	127.0.0.1	127.0.0.1	TCP	585	33690 → 9000 [PSH, ACK] Seq=3855907974 Ack=74939…
783	2020-04-02 18:07:13.0548268…	10.0.2.22	10.0.2.15	SSHv2	312	Client: Encrypted packet (len=244)
784	2020-04-02 18:07:13.0998797…	10.0.2.15	10.0.2.22	TCP	68	22 → 55770 [ACK] Seq=637359090 Ack=3196196851 Wi…
785	2020-04-02 18:07:13.1013222…	127.0.0.1	127.0.0.1	TCP	68	9000 → 33690 [ACK] Seq=749397925 Ack=3855908491 …
786	2020-04-02 18:07:13.1696725…	10.0.2.15	10.0.2.22	SSHv2	2968	Server: Encrypted packet (len=2900)

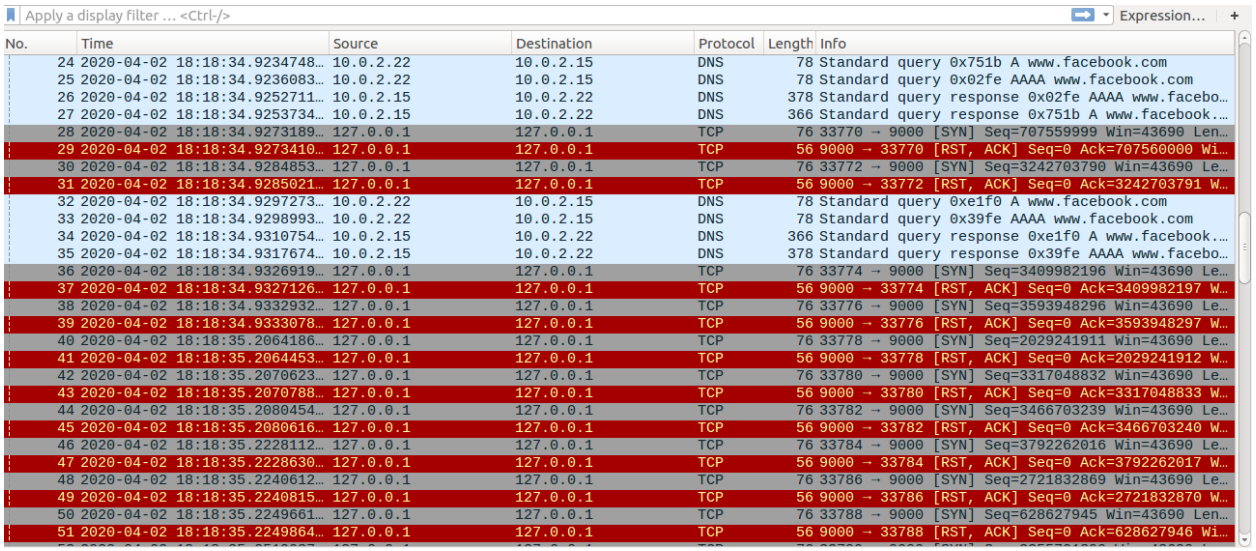
Break the SSH tunnel and clear Firefox cache

Here we break the SSH tunnel, and delete all the cache and cookies from our browser. Our packets are no longer encrypted by our SSH tunnel and hence our kernel starts dropping packets, and as seen in the wireshark output we observe a lot of retransmission due to the missing packets. Hence our Facebook page again is not accessible.

```
[04/02/2020 18:02] Rakshith-10.0.2.15@VM:~$exit
logout


^C[04/02/2020 18:16] Rakshith-10.0.2.22@VM:~/firewall$
```


Clear All History
Time range to clear:
Everything
All history will be cleared.
This action cannot be undone.
Details
Browsing & Download History
Form & Search History
Cookies
Cache
Active Logins
Offline Website Data
Site Preferences
Cancel
Clear Now

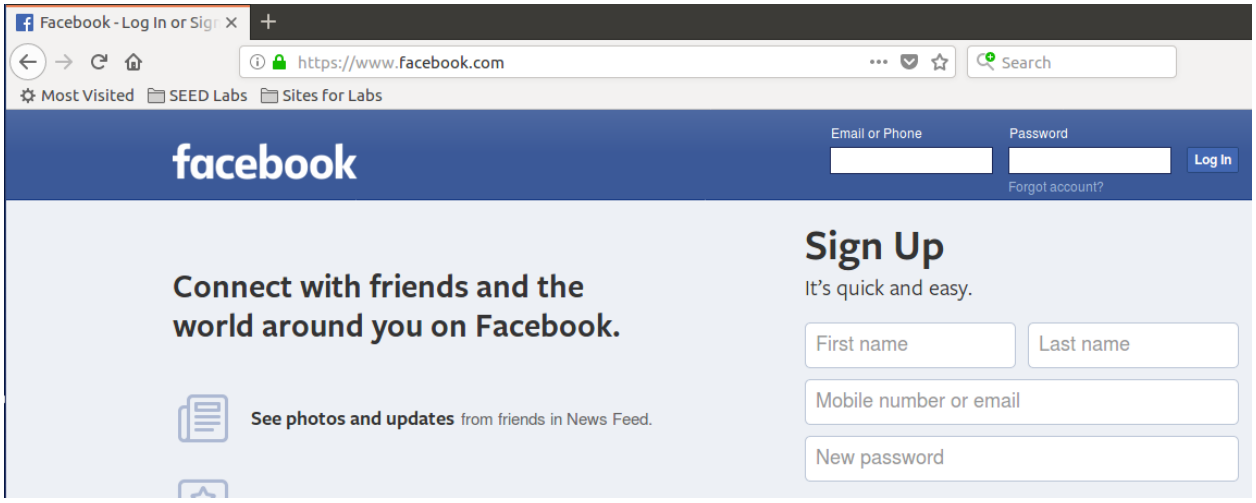

https://www.facebook.com
Search
Most Visited
SEED Labs
Sites for Labs
The proxy server is refusing connections
Firefox is configured to use a proxy server that is refusing connections.
Check the proxy settings to make sure that they are correct.
Contact your network administrator to make sure the proxy server is working.
Try Again


Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
24 2020-04-02 18:18:34.9234748… 10.0.2.22 10.0.2.15 DNS 78 Standard query 0x751b A www.facebook.com
25 2020-04-02 18:18:34.9236083… 10.0.2.22 10.0.2.15 DNS 78 Standard query 0x02fe AAAA www.facebook.com
26 2020-04-02 18:18:34.9252711… 10.0.2.15 10.0.2.22 DNS 378 Standard query response 0x02fe AAAA www.facebo…
27 2020-04-02 18:18:34.9253734… 10.0.2.15 10.0.2.22 DNS 366 Standard query response 0x751b A www.facebook.…
28 2020-04-02 18:18:34.9273189… 127.0.0.1 127.0.0.1 TCP 76 33770 → 9000 [SYN] Seq=707559999 Win=43690 Len…
29 2020-04-02 18:18:34.9273410… 127.0.0.1 127.0.0.1 TCP 56 9000 → 33770 [RST, ACK] Seq=0 Ack=707560000 Wi…
30 2020-04-02 18:18:34.9284853… 127.0.0.1 127.0.0.1 TCP 76 33772 → 9000 [SYN] Seq=3242703790 Win=43690 Le…
31 2020-04-02 18:18:34.9285021… 127.0.0.1 127.0.0.1 TCP 56 9000 → 33772 [RST, ACK] Seq=0 Ack=3242703791 W…
32 2020-04-02 18:18:34.9297273… 10.0.2.22 10.0.2.15 DNS 78 Standard query 0xe1f0 A www.facebook.com
33 2020-04-02 18:18:34.9298993… 10.0.2.22 10.0.2.15 DNS 78 Standard query 0x39fe AAAA www.facebook.com
34 2020-04-02 18:18:34.9310754… 10.0.2.15 10.0.2.22 DNS 366 Standard query response 0xe1f0 A www.facebook.…
35 2020-04-02 18:18:34.9317674… 10.0.2.15 10.0.2.22 DNS 378 Standard query response 0x39fe AAAA www.facebo…
36 2020-04-02 18:18:34.9326919… 127.0.0.1 127.0.0.1 TCP 76 33774 → 9000 [SYN] Seq=3409982196 Win=43690 Le…
37 2020-04-02 18:18:34.9327126… 127.0.0.1 127.0.0.1 TCP 56 9000 → 33774 [RST, ACK] Seq=0 Ack=3409982197 W…
38 2020-04-02 18:18:34.9332932… 127.0.0.1 127.0.0.1 TCP 76 33776 → 9000 [SYN] Seq=3593948296 Win=43690 Le…
39 2020-04-02 18:18:34.9333078… 127.0.0.1 127.0.0.1 TCP 56 9000 → 33776 [RST, ACK] Seq=0 Ack=3593948297 W…
40 2020-04-02 18:18:35.2064186… 127.0.0.1 127.0.0.1 TCP 76 33778 → 9000 [SYN] Seq=2029241911 Win=43690 Le…
41 2020-04-02 18:18:35.2064453… 127.0.0.1 127.0.0.1 TCP 56 9000 → 33778 [RST, ACK] Seq=0 Ack=2029241912 W…
42 2020-04-02 18:18:35.2070623… 127.0.0.1 127.0.0.1 TCP 76 33780 → 9000 [SYN] Seq=3317048832 Win=43690 Le…
43 2020-04-02 18:18:35.2070788… 127.0.0.1 127.0.0.1 TCP 56 9000 → 33780 [RST, ACK] Seq=0 Ack=3317048833 W…
44 2020-04-02 18:18:35.2080454… 127.0.0.1 127.0.0.1 TCP 76 33782 → 9000 [SYN] Seq=3466703239 Win=43690 Le…
45 2020-04-02 18:18:35.2080616… 127.0.0.1 127.0.0.1 TCP 56 9000 → 33782 [RST, ACK] Seq=0 Ack=3466703240 W…
46 2020-04-02 18:18:35.2228112… 127.0.0.1 127.0.0.1 TCP 76 33784 → 9000 [SYN] Seq=3792262016 Win=43690 Le…
47 2020-04-02 18:18:35.2228630… 127.0.0.1 127.0.0.1 TCP 56 9000 → 33784 [RST, ACK] Seq=0 Ack=3792262017 W…
48 2020-04-02 18:18:35.2240612… 127.0.0.1 127.0.0.1 TCP 76 33786 → 9000 [SYN] Seq=2721832869 Win=43690 Le…
49 2020-04-02 18:18:35.2240815… 127.0.0.1 127.0.0.1 TCP 56 9000 → 33786 [RST, ACK] Seq=0 Ack=2721832870 W…
50 2020-04-02 18:18:35.2249661… 127.0.0.1 127.0.0.1 TCP 76 33788 → 9000 [SYN] Seq=628627945 Win=43690 Len…
51 2020-04-02 18:18:35.2249864… 127.0.0.1 127.0.0.1 TCP 56 9000 → 33788 [RST, ACK] Seq=0 Ack=628627946 Wi…

Establish tunnel again:

We enable the SSH tunnel again our kernel gets the missing packets and the page is restored.

```
[04/02/2020 18:23] Rakshith-10.0.2.22@VM:~/firewall$ssh -D 9000 -C seed@10.0.2.15
seed@10.0.2.15's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

Last login: Thu Apr  2 18:02:01 2020 from 10.0.2.22
[04/02/2020 18:23] Rakshith-10.0.2.15@VM:~$
```



No.	Time	Source	Destination	Protocol	Length	Info
66	2020-04-02 18:28:30.0958901…	127.0.0.1	127.0.0.1	TCP	76	33878 → 9000 [SYN] Seq=2132442281 Win=4369…
67	2020-04-02 18:28:30.0959053…	127.0.0.1	127.0.0.1	TCP	56	9000 → 33878 [RST, ACK] Seq=0 Ack=21324422…
68	2020-04-02 18:28:30.0968245…	127.0.0.1	127.0.0.1	TCP	76	33880 → 9000 [SYN] Seq=1318290086 Win=4369…
69	2020-04-02 18:28:30.0968404…	127.0.0.1	127.0.0.1	TCP	56	9000 → 33880 [RST, ACK] Seq=0 Ack=13182900…
70	2020-04-02 18:28:30.0993217…	127.0.0.1	127.0.0.1	TCP	76	33882 → 9000 [SYN] Seq=2323760157 Win=4369…
71	2020-04-02 18:28:30.0993507…	127.0.0.1	127.0.0.1	TCP	56	9000 → 33882 [RST, ACK] Seq=0 Ack=23237601…
72	2020-04-02 18:28:40.6454254…	::1	::1	UDP	64	38866 → 45936 Len=0
73	2020-04-02 18:28:48.1515651…	10.0.2.22	10.0.2.15	TCP	76	56760 → 22 [SYN] Seq=3497192883 Win=29200 …
74	2020-04-02 18:28:48.1523253…	10.0.2.15	10.0.2.22	TCP	76	22 → 56760 [SYN, ACK] Seq=444610953 Ack=34…
75	2020-04-02 18:28:48.1524311…	10.0.2.22	10.0.2.15	TCP	68	56760 → 22 [ACK] Seq=3497192884 Ack=444610…
76	2020-04-02 18:28:48.1542872…	10.0.2.22	10.0.2.15	SSHv2	109	Client: Protocol (SSH-2.0-OpenSSH_7.2p2 Ub…
77	2020-04-02 18:28:48.1548092…	10.0.2.15	10.0.2.22	TCP	68	22 → 56760 [ACK] Seq=444610954 Ack=3497192…
78	2020-04-02 18:28:48.1855872…	10.0.2.15	10.0.2.22	SSHv2	109	Server: Protocol (SSH-2.0-OpenSSH_7.2p2 Ub…
79	2020-04-02 18:28:48.1871292…	10.0.2.22	10.0.2.15	TCP	68	56760 → 22 [ACK] Seq=3497192925 Ack=444610…
80	2020-04-02 18:28:48.1885457…	10.0.2.15	10.0.2.22	SSHv2	1044	Server: Key Exchange Init
81	2020-04-02 18:28:48.1885789…	10.0.2.22	10.0.2.15	TCP	68	56760 → 22 [ACK] Seq=3497192925 Ack=444611…
82	2020-04-02 18:28:48.1939439…	10.0.2.22	10.0.2.15	SSHv2	1404	Client: Key Exchange Init
83	2020-04-02 18:28:48.2387399…	10.0.2.15	10.0.2.22	TCP	68	22 → 56760 [ACK] Seq=444611971 Ack=3497194…
84	2020-04-02 18:28:48.2387953…	10.0.2.22	10.0.2.15	SSHv2	116	Client: Diffie-Hellman Key Exchange Init
85	2020-04-02 18:28:48.2392892…	10.0.2.15	10.0.2.22	TCP	68	22 → 56760 [ACK] Seq=444611971 Ack=3497194…
86	2020-04-02 18:28:48.2644943…	10.0.2.15	10.0.2.22	SSHv2	432	Server: Diffie-Hellman Key Exchange Reply,…
87	2020-04-02 18:28:48.2963068…	10.0.2.22	10.0.2.15	SSHv2	84	Client: New Keys
88	2020-04-02 18:28:48.3433513…	10.0.2.15	10.0.2.22	TCP	68	22 → 56760 [ACK] Seq=444612335 Ack=3497194…
89	2020-04-02 18:28:48.3434039…	10.0.2.22	10.0.2.15	SSHv2	112	Client: Encrypted packet (len=44)
90	2020-04-02 18:28:48.3439699…	10.0.2.15	10.0.2.22	TCP	68	22 → 56760 [ACK] Seq=444612335 Ack=3497194…
91	2020-04-02 18:28:48.3443405…	10.0.2.15	10.0.2.22	SSHv2	112	Server: Encrypted packet (len=44)
92	2020-04-02 18:28:48.3446530…	10.0.2.22	10.0.2.15	SSHv2	128	Client: Encrypted packet (len=60)
93	2020-04-02 18:28:48.3471572…	10.0.2.15	10.0.2.22	SSHv2	120	Server: Encrypted packet (len=52)

Task 4: Evading Ingress Filtering

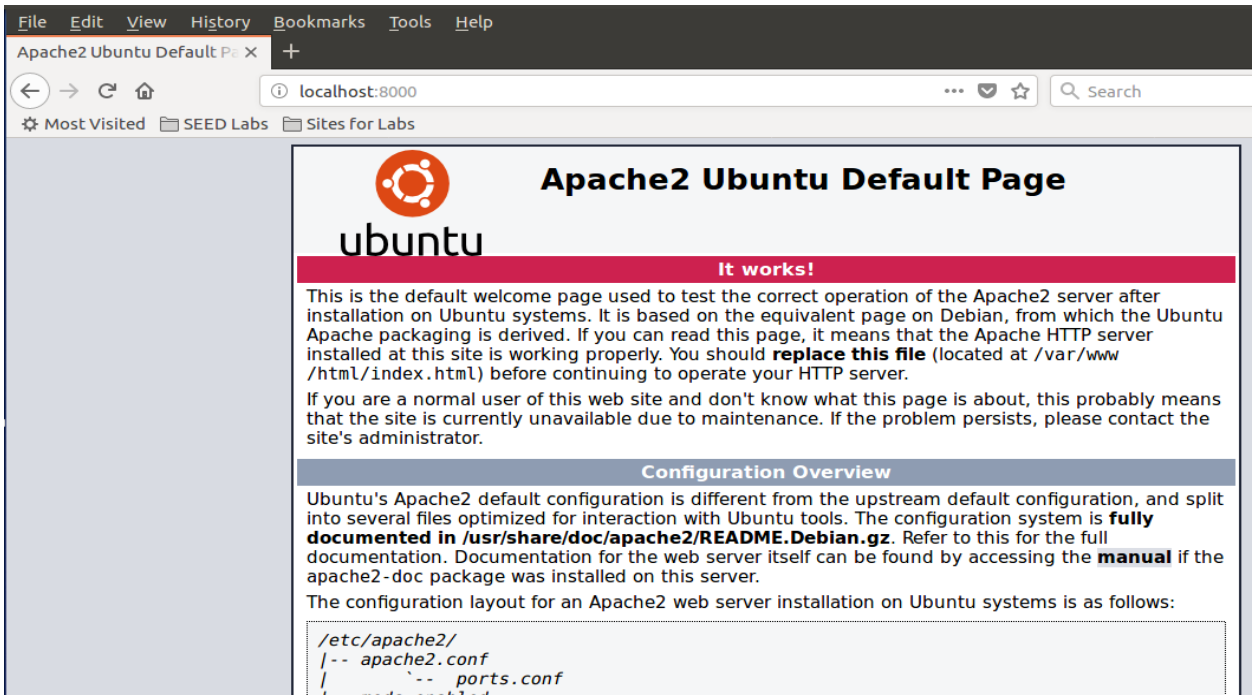
On Machine A, we block Machine B from accessing its port 80(web server) and 22 (SSH server).

```
[04/02/2020 20:24] Rakshith-10.0.2.15@VM:~$sudo iptables -A INPUT -p tcp --dport 80 -s 10.0.2.22 -d 10.0.2.15 -j DROP
[sudo] password for seed:
[04/02/2020 20:49] Rakshith-10.0.2.15@VM:~$sudo iptables -A INPUT -p tcp --dport 22 -s 10.0.2.22 -d 10.0.2.15 -j DROP
[04/02/2020 20:50] Rakshith-10.0.2.15@VM:~$sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -s 10.0.2.22/32 -d 10.0.2.15/32 -p tcp -m tcp --dport 80 -j DROP
-A INPUT -s 10.0.2.22/32 -d 10.0.2.15/32 -p tcp -m tcp --dport 22 -j DROP
[04/02/2020 20:50] Rakshith-10.0.2.15@VM:~$
```

```
[04/02/2020 20:52] Rakshith-10.0.2.22@VM:~$ssh 10.0.2.15
^C
[04/02/2020 20:53] Rakshith-10.0.2.22@VM:~$tcptraceroute 10.0.2.22 80
You do not have enough privileges to use this traceroute method.
socket: Operation not permitted
[04/02/2020 20:53] Rakshith-10.0.2.22@VM:~$sudo tcptraceroute 10.0.2.22 80
[sudo] password for seed:
traceroute to 10.0.2.22 (10.0.2.22), 30 hops max, 60 byte packets
 1  www.seedIoT32.com (10.0.2.22) <syn,ack>  0.078 ms  0.039 ms  0.022 ms
[04/02/2020 20:53] Rakshith-10.0.2.22@VM:~$sudo tcptraceroute 10.0.2.15 80
traceroute to 10.0.2.15 (10.0.2.15), 30 hops max, 60 byte packets
 1  * * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  *
```

We then setup reverse SSH in machine A so that machine B can access files in machine A. This can be demonstrated below with the wireshark output.

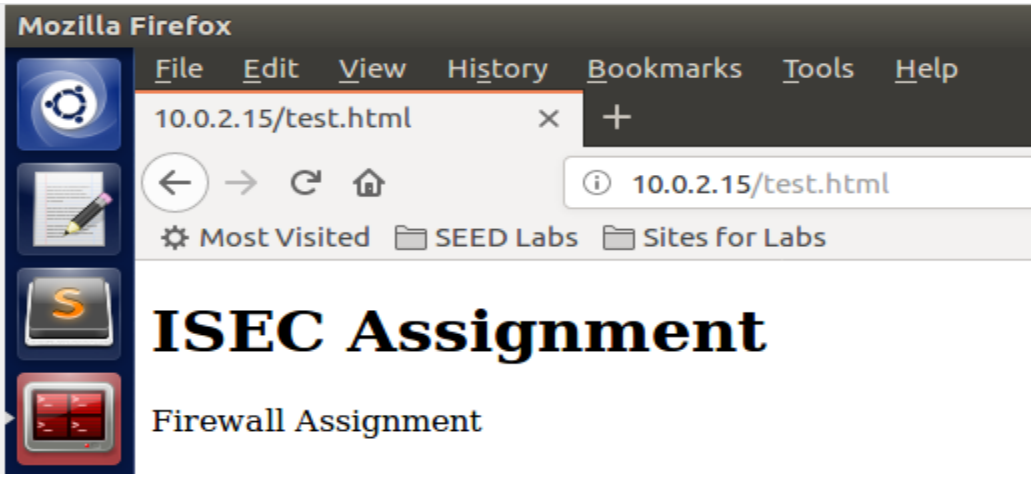
```
[04/02/2020 20:57] Rakshith-10.0.2.15@VM:~$ssh -R 8000:localhost:80 seed@10.0.2.22
seed@10.0.2.22's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

Last login: Thu Apr  2 02:43:21 2020 from 10.0.2.15
[04/02/2020 20:58] Rakshith-10.0.2.22@VM:~$
```



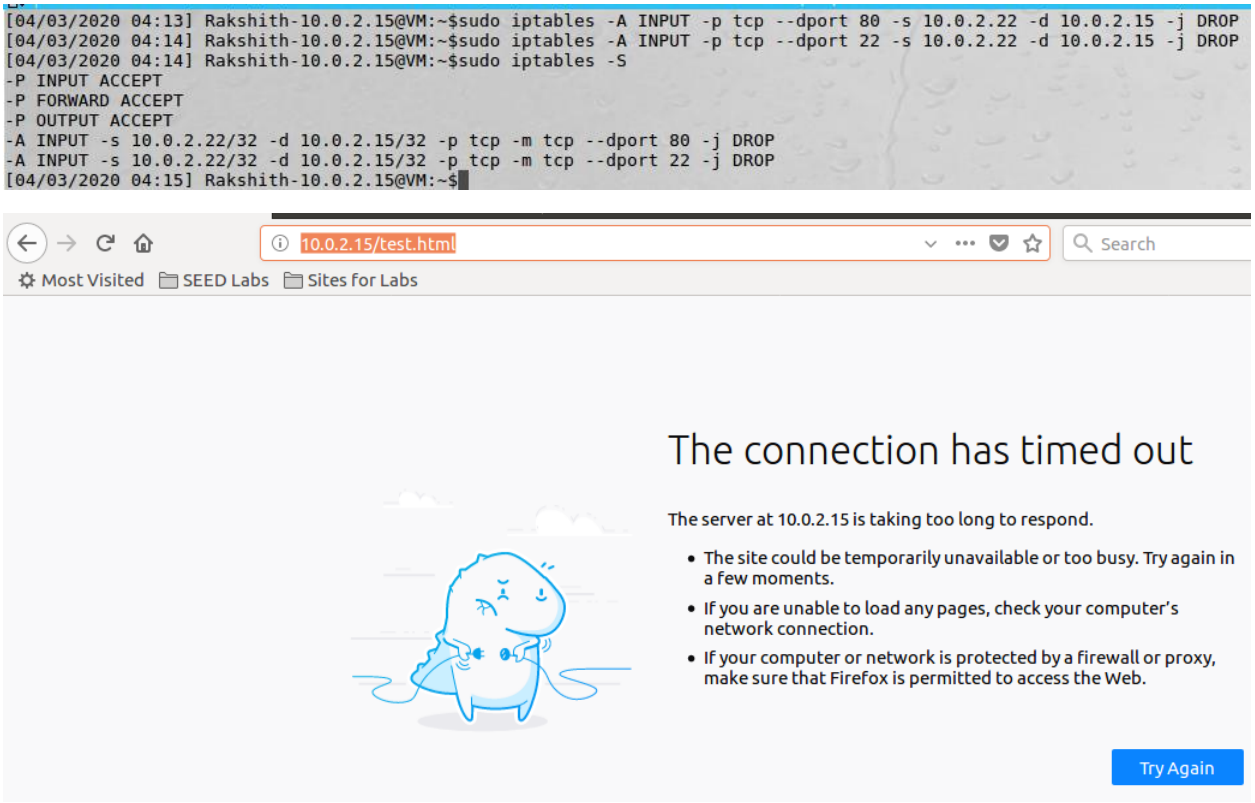

```
513 2020-04-02 21:03:26.0973139… 127.0.0.1   127.0.0.1   HTTP   352 GET /icons/ubuntu-logo.png HTTP/1.1
514 2020-04-02 21:03:26.0973604… 127.0.0.1   127.0.0.1   TCP     68 8000 → 39380 [ACK] Seq=2009054074 Ack=144000
515 2020-04-02 21:03:26.0975388… 10.0.2.22   10.0.2.15   SSH    392 Server: Encrypted packet (len=324)
516 2020-04-02 21:03:26.0983389… 127.0.0.1   127.0.0.1   TCP     68 39380 → 8000 [FIN, ACK] Seq=1440005521 Ack=2
517 2020-04-02 21:03:26.0985543… 10.0.2.22   10.0.2.15   SSH    104 Server: Encrypted packet (len=36)
518 2020-04-02 21:03:26.0999265… 10.0.2.15   10.0.2.22   TCP     68 55356 → 22 [ACK] Seq=4147401270 Ack=31911136
519 2020-04-02 21:03:26.1013339… 10.0.2.15   10.0.2.22   SSH   3728 Client: Encrypted packet (len=3660)
520 2020-04-02 21:03:26.1013815… 10.0.2.22   10.0.2.15   TCP     68 22 → 55356 [ACK] Seq=3191113658 Ack=41474049
521 2020-04-02 21:03:26.1019334… 127.0.0.1   127.0.0.1   HTTP   3692 HTTP/1.1 200 OK  (PNG)
522 2020-04-02 21:03:26.1019813… 127.0.0.1   127.0.0.1   TCP     56 39380 → 8000 [RST] Seq=1440005522 Win=0 Len=
523 2020-04-02 21:03:26.1025635… 10.0.2.15   10.0.2.22   SSH    140 Client: Encrypted packet (len=72)
524 2020-04-02 21:03:26.1025976… 10.0.2.22   10.0.2.15   TCP     68 22 → 55356 [ACK] Seq=3191113658 Ack=41474056
525 2020-04-02 21:03:26.1034369… 10.0.2.22   10.0.2.15   SSH    104 Server: Encrypted packet (len=36)
526 2020-04-02 21:03:26.1455032… 10.0.2.15   10.0.2.22   TCP     68 55356 → 22 [ACK] Seq=4147405002 Ack=31911136
```

```
[04/02/2020 21:00] Rakshith-10.0.2.22@VM:~$sudo tcptraceroute 10.0.2.15 8000
traceroute to 10.0.2.15 (10.0.2.15), 30 hops max, 60 byte packets
 1  10.0.2.15 (10.0.2.15) <rst,ack>  0.539 ms  0.796 ms  0.693 ms
[04/02/2020 21:01] Rakshith-10.0.2.22@VM:~$
```

We create a file in /var/www folder in machine A, we can load the html file from machine B like below.

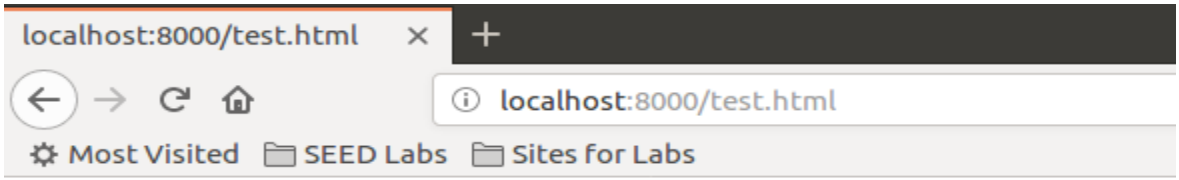


We then block SSH and HTTP traffic in machine A. Now we can see that the page is not accessible from machine B.



We then establish a reverse SSH tunnel to machine B from machine A and connect to localhost:80 on port 8000. Through this tunnel we can again view the webpage from machine B.

```
[04/03/2020 04:15] Rakshith-10.0.2.15@VM:~$ssh -R 8000:localhost:80 seed@10.0.2.22
seed@10.0.2.22's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

Last login: Fri Apr  3 02:05:47 2020 from 10.0.2.15
[04/03/2020 04:16] Rakshith-10.0.2.22@VM:~$
```



ISEC Assignment

Firewall Assignment

57	2020-…	127.0.0.1	127.0.0.1	TCP	76 53056 → 8000 [SYN] Seq=145274272 Win=43690 Len=0 MSS=65495 SACK_PERM…
58	2020-…	127.0.0.1	127.0.0.1	TCP	76 8000 → 53056 [SYN, ACK] Seq=274829001 Ack=145274273 Win=43690 Len=0 …
59	2020-…	127.0.0.1	127.0.0.1	TCP	68 53056 → 8000 [ACK] Seq=145274273 Ack=274829002 Win=43776 Len=0 TSval…
60	2020-…	10.0.2.22	10.0.2.15	SSH	160 Server: Encrypted packet (len=92)
61	2020-…	10.0.2.15	10.0.2.22	TCP	68 55786 → 22 [ACK] Seq=1522014424 Ack=4115116503 Win=290 Len=0 TSval=1…
62	2020-…	10.0.2.15	10.0.2.22	SSH	112 Client: Encrypted packet (len=44)
63	2020-…	10.0.2.22	10.0.2.15	TCP	68 22 → 55786 [ACK] Seq=4115116503 Ack=1522014468 Win=270 Len=0 TSval=4…
64	2020-…	127.0.0.1	127.0.0.1	HTTP	397 GET /test.html HTTP/1.1
65	2020-…	127.0.0.1	127.0.0.1	TCP	68 8000 → 53056 [ACK] Seq=274829002 Ack=145274602 Win=44800 Len=0 TSval…
66	2020-…	10.0.2.22	10.0.2.15	SSH	432 Server: Encrypted packet (len=364)
67	2020-…	10.0.2.15	10.0.2.22	SSH	536 Client: Encrypted packet (len=468)
68	2020-…	10.0.2.22	10.0.2.15	TCP	68 22 → 55786 [ACK] Seq=4115116867 Ack=1522014936 Win=291 Len=0 TSval=4…
69	2020-…	127.0.0.1	127.0.0.1	HTTP	497 HTTP/1.1 200 OK (text/html)
70	2020-…	127.0.0.1	127.0.0.1	TCP	68 53056 → 8000 [ACK] Seq=145274602 Ack=274829431 Win=44800 Len=0 TSval…
71	2020-…	127.0.0.1	127.0.0.1	HTTP	369 GET /favicon.ico HTTP/1.1
72	2020-…	10.0.2.22	10.0.2.15	SSH	408 Server: Encrypted packet (len=340)
73	2020-…	10.0.2.15	10.0.2.22	SSH	608 Client: Encrypted packet (len=540)
74	2020-…	10.0.2.22	10.0.2.15	TCP	68 22 → 55786 [ACK] Seq=4115117207 Ack=1522015476 Win=312 Len=0 TSval=4…
75	2020-…	127.0.0.1	127.0.0.1	HTTP	570 HTTP/1.1 404 Not Found (text/html)
76	2020-…	127.0.0.1	127.0.0.1	TCP	68 53056 → 8000 [ACK] Seq=145274903 Ack=274829933 Win=45952 Len=0 TSval…

Task 5 (SNAT): Use iptables to set up a SNAT. Use Wireshark to prove that your SNAT is working

I am trying to reach machine A through the internal machine inside machine B's network (host V), currently I am not able to reach the machine A because machine A doesn't recognize the source address of host V (192.168.60.101).

```
[04/03/2020 17:36] Rakshith-10.0.2.10@VM:~$ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
^C
--- 10.0.2.15 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3070ms

[04/03/2020 17:37] Rakshith-10.0.2.10@VM:~$
```

Now I setup SNAT in machine B, I am modifying the source address of host V to that of machine B, and I am making these changes just before my packet leaves the interface that's why we use the chain POSTROUTING.

```
[04/03/2020 17:49] Rakshith-10.0.2.22@VM:~/firewall$sudo iptables -t nat -A POSTROUTING -o enp0s3 -j SNAT --to-source 10.0.2.22
[04/03/2020 17:49] Rakshith-10.0.2.22@VM:~/firewall$
```

Now we can successfully ping 10.0.2.15 (Machine A), we can see the results in wireshark as well

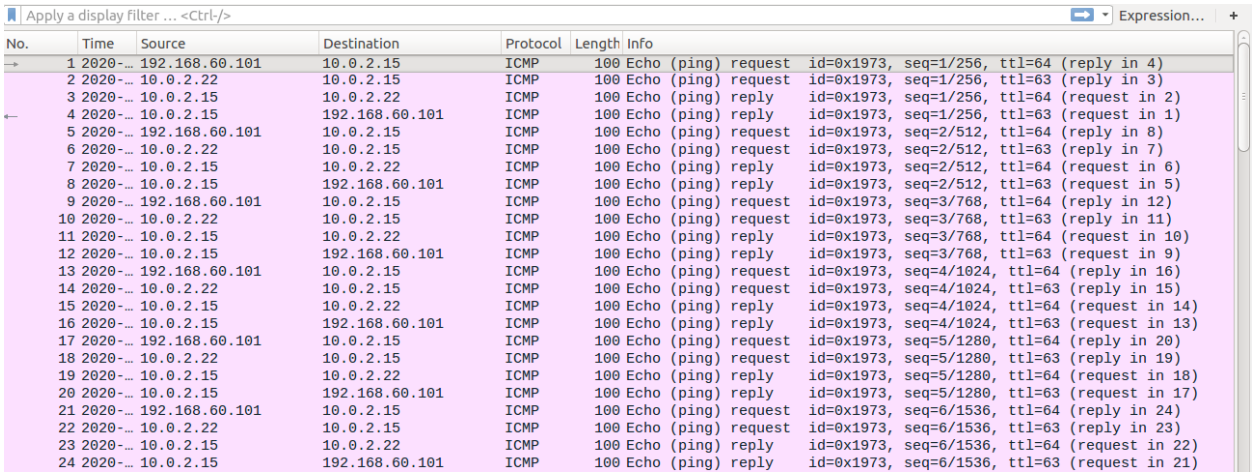
```
[04/03/2020 17:47] Rakshith-10.0.2.10@VM:~$ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=63 time=1.46 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=63 time=1.41 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=63 time=1.36 ms
64 bytes from 10.0.2.15: icmp_seq=4 ttl=63 time=1.38 ms
64 bytes from 10.0.2.15: icmp_seq=5 ttl=63 time=1.81 ms
64 bytes from 10.0.2.15: icmp_seq=6 ttl=63 time=2.28 ms
64 bytes from 10.0.2.15: icmp_seq=7 ttl=63 time=1.29 ms
```

Wireshark output in Host V (192.168.60.101)

Apply a display filter … <Ctrl-/> Expression…

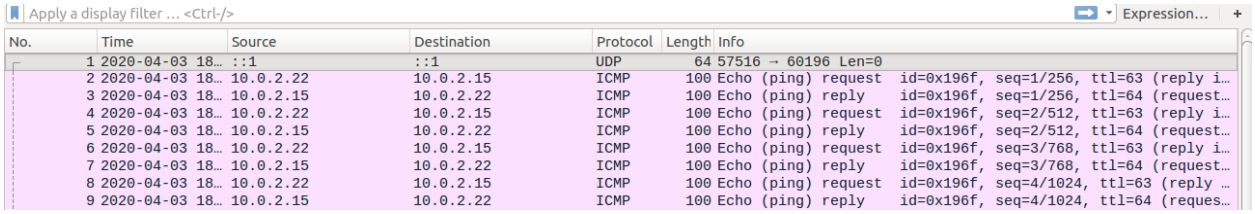
No.	Time	Source	Destination	Protocol	Length	Info
1	2020-04-03…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x194a, seq=9/2304, ttl=64 (reply in 2)
2	2020-04-03…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x194a, seq=9/2304, ttl=63 (request in 1)
3	2020-04-03…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x194a, seq=10/2560, ttl=64 (reply in 4)
4	2020-04-03…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x194a, seq=10/2560, ttl=63 (request in 3)
5	2020-04-03…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x194a, seq=11/2816, ttl=64 (reply in 6)
6	2020-04-03…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x194a, seq=11/2816, ttl=63 (request in 5)
7	2020-04-03…	::1	::1	UDP	64	39210 → 45464 Len=0
8	2020-04-03…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x194a, seq=12/3072, ttl=64 (reply in 9)
9	2020-04-03…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x194a, seq=12/3072, ttl=63 (request in 8)
10	2020-04-03…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x194a, seq=13/3328, ttl=64 (reply in 11)
11	2020-04-03…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x194a, seq=13/3328, ttl=63 (request in 10)
12	2020-04-03…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x194a, seq=14/3584, ttl=64 (reply in 13)
13	2020-04-03…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x194a, seq=14/3584, ttl=63 (request in 12)
14	2020-04-03…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x194a, seq=15/3840, ttl=64 (reply in 15)

Wireshark output in machine B (10.0.2.22); here we can see that request from source 192.168.60.101 is modified and new source IP is 10.0.2.22, and the reply from 10.0.2.15 is then forwarded to 192.168.60.101.



No.	Time	Source	Destination	Protocol	Length	Info
1	2020-…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=1/256, ttl=64 (reply in 4)
2	2020-…	10.0.2.22	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=1/256, ttl=63 (reply in 3)
3	2020-…	10.0.2.15	10.0.2.22	ICMP	100	Echo (ping) reply id=0x1973, seq=1/256, ttl=64 (request in 2)
4	2020-…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x1973, seq=1/256, ttl=63 (request in 1)
5	2020-…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=2/512, ttl=64 (reply in 8)
6	2020-…	10.0.2.22	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=2/512, ttl=63 (reply in 7)
7	2020-…	10.0.2.15	10.0.2.22	ICMP	100	Echo (ping) reply id=0x1973, seq=2/512, ttl=64 (request in 6)
8	2020-…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x1973, seq=2/512, ttl=63 (request in 5)
9	2020-…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=3/768, ttl=64 (reply in 12)
10	2020-…	10.0.2.22	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=3/768, ttl=63 (reply in 11)
11	2020-…	10.0.2.15	10.0.2.22	ICMP	100	Echo (ping) reply id=0x1973, seq=3/768, ttl=64 (request in 10)
12	2020-…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x1973, seq=3/768, ttl=63 (request in 9)
13	2020-…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=4/1024, ttl=64 (reply in 16)
14	2020-…	10.0.2.22	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=4/1024, ttl=63 (reply in 15)
15	2020-…	10.0.2.15	10.0.2.22	ICMP	100	Echo (ping) reply id=0x1973, seq=4/1024, ttl=64 (request in 14)
16	2020-…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x1973, seq=4/1024, ttl=63 (request in 13)
17	2020-…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=5/1280, ttl=64 (reply in 20)
18	2020-…	10.0.2.22	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=5/1280, ttl=63 (reply in 19)
19	2020-…	10.0.2.15	10.0.2.22	ICMP	100	Echo (ping) reply id=0x1973, seq=5/1280, ttl=64 (request in 18)
20	2020-…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x1973, seq=5/1280, ttl=63 (request in 17)
21	2020-…	192.168.60.101	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=6/1536, ttl=64 (reply in 24)
22	2020-…	10.0.2.22	10.0.2.15	ICMP	100	Echo (ping) request id=0x1973, seq=6/1536, ttl=63 (reply in 23)
23	2020-…	10.0.2.15	10.0.2.22	ICMP	100	Echo (ping) reply id=0x1973, seq=6/1536, ttl=64 (request in 22)
24	2020-…	10.0.2.15	192.168.60.101	ICMP	100	Echo (ping) reply id=0x1973, seq=6/1536, ttl=63 (request in 21)

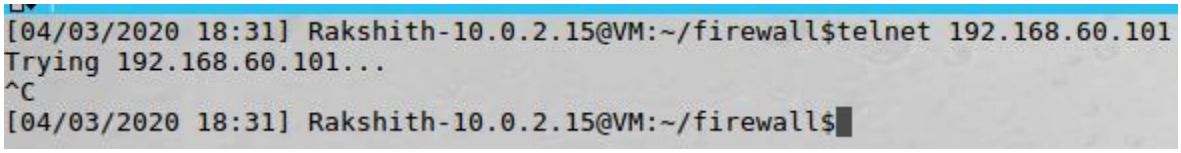
Wireshark output in Machine A (10.0.2.15)



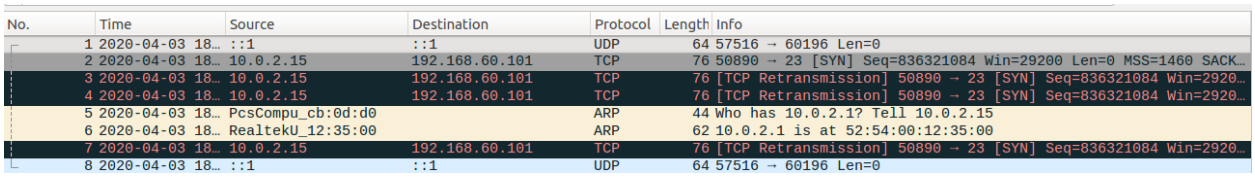
No.	Time	Source	Destination	Protocol	Length	Info
1	2020-04-03 18…	::1	::1	UDP	64	57516 → 60196 Len=0
2	2020-04-03 18…	10.0.2.22	10.0.2.15	ICMP	100	Echo (ping) request id=0x196f, seq=1/256, ttl=63 (reply i…
3	2020-04-03 18…	10.0.2.15	10.0.2.22	ICMP	100	Echo (ping) reply id=0x196f, seq=1/256, ttl=64 (request…
4	2020-04-03 18…	10.0.2.22	10.0.2.15	ICMP	100	Echo (ping) request id=0x196f, seq=2/512, ttl=63 (reply i…
5	2020-04-03 18…	10.0.2.15	10.0.2.22	ICMP	100	Echo (ping) reply id=0x196f, seq=2/512, ttl=64 (request…
6	2020-04-03 18…	10.0.2.22	10.0.2.15	ICMP	100	Echo (ping) request id=0x196f, seq=3/768, ttl=63 (reply i…
7	2020-04-03 18…	10.0.2.15	10.0.2.22	ICMP	100	Echo (ping) reply id=0x196f, seq=3/768, ttl=64 (request…
8	2020-04-03 18…	10.0.2.22	10.0.2.15	ICMP	100	Echo (ping) request id=0x196f, seq=4/1024, ttl=63 (reply …
9	2020-04-03 18…	10.0.2.15	10.0.2.22	ICMP	100	Echo (ping) reply id=0x196f, seq=4/1024, ttl=64 (reques…

Task 6 (DNAT): Use iptables to set up a DNAT for port forwarding. Use Wireshark to prove that your DNAT is working.

Here we are trying to reach webserver inside the network of machine B (10.0.2.22) and currently we are unable to.



```
[04/03/2020 18:31] Rakshith-10.0.2.15@VM:~/firewall$telnet 192.168.60.101
Trying 192.168.60.101...
^C
[04/03/2020 18:31] Rakshith-10.0.2.15@VM:~/firewall$
```



No.	Time	Source	Destination	Protocol	Length	Info
1	2020-04-03 18…	::1	::1	UDP	64	57516 → 60196 Len=0
2	2020-04-03 18…	10.0.2.15	192.168.60.101	TCP	76	50890 → 23 [SYN] Seq=836321084 Win=29200 Len=0 MSS=1460 SACK…
3	2020-04-03 18…	10.0.2.15	192.168.60.101	TCP	76	[TCP Retransmission] 50890 → 23 [SYN] Seq=836321084 Win=2920…
4	2020-04-03 18…	10.0.2.15	192.168.60.101	TCP	76	[TCP Retransmission] 50890 → 23 [SYN] Seq=836321084 Win=2920…
5	2020-04-03 18…	PcsCompu_cb:0d:d0		ARP	44	Who has 10.0.2.1? Tell 10.0.2.15
6	2020-04-03 18…	RealtekU_12:35:00		ARP	62	10.0.2.1 is at 52:54:00:12:35:00
7	2020-04-03 18…	10.0.2.15	192.168.60.101	TCP	76	[TCP Retransmission] 50890 → 23 [SYN] Seq=836321084 Win=2920…
8	2020-04-03 18…	::1	::1	UDP	64	57516 → 60196 Len=0

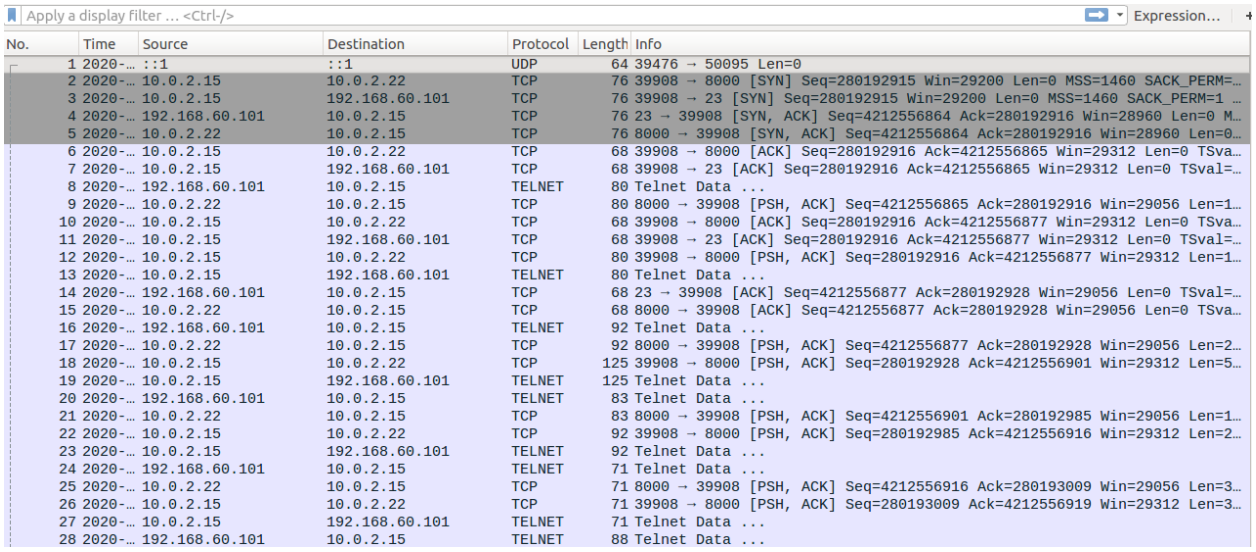
We set up a DNAT on the server machine B (10.0.2.22) with a rule that whenever machine A wants to reach 192.168.60.101 it can telnet to 10.0.2.22 on port 8000. This enables us to do both port forwarding and IP forwarding. We are successfully masking the IP and port number of the webserver from the internet as well.

```
[04/03/2020 18:30] Rakshith-10.0.2.22@VM:~/firewall$sudo iptables -t nat -A PREROUTING -p tcp --dport 8000 -j DNAT --to-destination 192.168.60.101:23
[04/03/2020 18:38] Rakshith-10.0.2.22@VM:~/firewall$
```

Here we are able to telnet to webserver inside 10.0.2.22 (machine B) when we try telnet on port 8000.

```
[04/03/2020 18:31] Rakshith-10.0.2.15@VM:~/firewall$telnet 10.0.2.22 8000
Trying 10.0.2.22...
Connected to 10.0.2.22.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Thu Mar 26 22:48:50 EDT 2020 from 192.168.53.5 on pts/20
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[04/03/2020 18:39] Rakshith-10.0.2.10@VM:~$
```

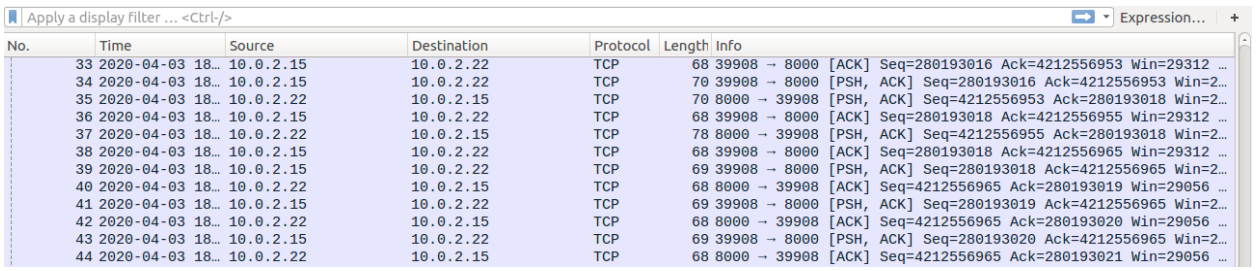
Wireshark output in machine B (server), 10.0.2.15 tries to telnet 10.0.2.22 on port 8000, then 10.0.2.15 is allowed to telnet to 192.168.60.101 on port 23, and the chain continues for each and every packet.



Apply a display filter … <Ctrl-/> Expression… +

No.	Time	Source	Destination	Protocol	Length	Info
1	2020-…	::1	::1	UDP	64	39476 → 50095 Len=0
2	2020-…	10.0.2.15	10.0.2.22	TCP	76	39908 → 8000 [SYN] Seq=280192915 Win=29200 Len=0 MSS=1460 SACK_PERM=…
3	2020-…	10.0.2.15	192.168.60.101	TCP	76	39908 → 23 [SYN] Seq=280192915 Win=29200 Len=0 MSS=1460 SACK_PERM=1 …
4	2020-…	192.168.60.101	10.0.2.15	TCP	76	23 → 39908 [SYN, ACK] Seq=4212556864 Ack=280192916 Win=28960 Len=0 M…
5	2020-…	10.0.2.22	10.0.2.15	TCP	76	8000 → 39908 [SYN, ACK] Seq=4212556864 Ack=280192916 Win=28960 Len=0…
6	2020-…	10.0.2.15	10.0.2.22	TCP	68	39908 → 8000 [ACK] Seq=280192916 Ack=4212556865 Win=29312 Len=0 TSva…
7	2020-…	10.0.2.15	192.168.60.101	TCP	68	39908 → 23 [ACK] Seq=280192916 Ack=4212556865 Win=29312 Len=0 TSval=…
8	2020-…	192.168.60.101	10.0.2.15	TELNET	80	Telnet Data …
9	2020-…	10.0.2.22	10.0.2.15	TCP	80	8000 → 39908 [PSH, ACK] Seq=4212556865 Ack=280192916 Win=29056 Len=1…
10	2020-…	10.0.2.15	10.0.2.22	TCP	68	39908 → 8000 [ACK] Seq=280192916 Ack=4212556877 Win=29312 Len=0 TSva…
11	2020-…	10.0.2.15	192.168.60.101	TCP	68	39908 → 23 [ACK] Seq=280192916 Ack=4212556877 Win=29312 Len=0 TSval=…
12	2020-…	10.0.2.15	10.0.2.22	TCP	80	39908 → 8000 [PSH, ACK] Seq=280192916 Ack=4212556877 Win=29312 Len=1…
13	2020-…	10.0.2.15	192.168.60.101	TELNET	80	Telnet Data …
14	2020-…	192.168.60.101	10.0.2.15	TCP	68	23 → 39908 [ACK] Seq=4212556877 Ack=280192928 Win=29056 Len=0 TSval=…
15	2020-…	10.0.2.22	10.0.2.15	TCP	68	8000 → 39908 [ACK] Seq=4212556877 Ack=280192928 Win=29056 Len=0 TSva…
16	2020-…	192.168.60.101	10.0.2.15	TELNET	92	Telnet Data …
17	2020-…	10.0.2.22	10.0.2.15	TCP	92	8000 → 39908 [PSH, ACK] Seq=4212556877 Ack=280192928 Win=29056 Len=2…
18	2020-…	10.0.2.15	10.0.2.22	TCP	125	39908 → 8000 [PSH, ACK] Seq=280192928 Ack=4212556901 Win=29312 Len=5…
19	2020-…	10.0.2.15	192.168.60.101	TELNET	125	Telnet Data …
20	2020-…	192.168.60.101	10.0.2.15	TELNET	83	Telnet Data …
21	2020-…	10.0.2.22	10.0.2.15	TCP	83	8000 → 39908 [PSH, ACK] Seq=4212556901 Ack=280192985 Win=29056 Len=1…
22	2020-…	10.0.2.15	10.0.2.22	TCP	92	39908 → 8000 [PSH, ACK] Seq=280192985 Ack=4212556916 Win=29312 Len=2…
23	2020-…	10.0.2.15	192.168.60.101	TELNET	92	Telnet Data …
24	2020-…	192.168.60.101	10.0.2.15	TELNET	71	Telnet Data …
25	2020-…	10.0.2.22	10.0.2.15	TCP	71	8000 → 39908 [PSH, ACK] Seq=4212556916 Ack=280193009 Win=29056 Len=3…
26	2020-…	10.0.2.15	10.0.2.22	TCP	71	39908 → 8000 [PSH, ACK] Seq=280193009 Ack=4212556919 Win=29312 Len=3…
27	2020-…	10.0.2.15	192.168.60.101	TELNET	71	Telnet Data …
28	2020-…	192.168.60.101	10.0.2.15	TELNET	88	Telnet Data …

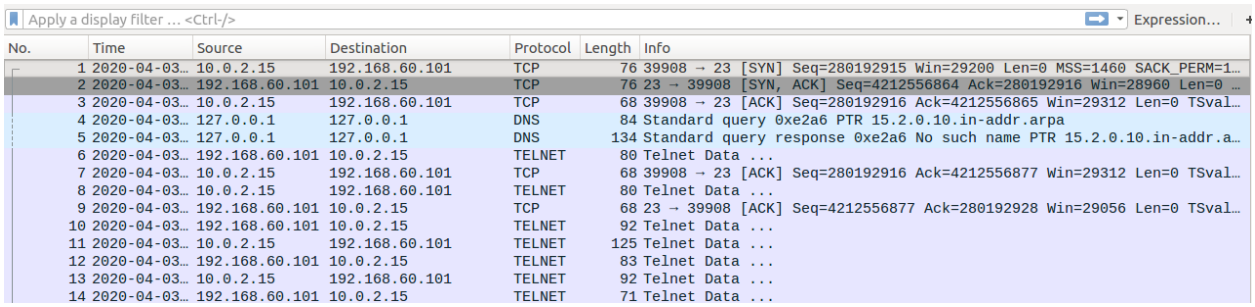
This is the wireshark output in Machine A, it doesn't know the IP address of the actual webserver, it thinks 10.0.2.22 is the webserver.



Apply a display filter … <Ctrl-/> Expression… +

No.	Time	Source	Destination	Protocol	Length	Info
33	2020-04-03 18…	10.0.2.15	10.0.2.22	TCP	68	39908 → 8000 [ACK] Seq=280193016 Ack=4212556953 Win=29312 …
34	2020-04-03 18…	10.0.2.15	10.0.2.22	TCP	70	39908 → 8000 [PSH, ACK] Seq=280193016 Ack=4212556953 Win=2…
35	2020-04-03 18…	10.0.2.22	10.0.2.15	TCP	70	8000 → 39908 [PSH, ACK] Seq=4212556953 Ack=280193018 Win=2…
36	2020-04-03 18…	10.0.2.15	10.0.2.22	TCP	68	39908 → 8000 [ACK] Seq=280193018 Ack=4212556955 Win=29312 …
37	2020-04-03 18…	10.0.2.22	10.0.2.15	TCP	78	8000 → 39908 [PSH, ACK] Seq=4212556955 Ack=280193018 Win=2…
38	2020-04-03 18…	10.0.2.15	10.0.2.22	TCP	68	39908 → 8000 [ACK] Seq=280193018 Ack=4212556965 Win=29312 …
39	2020-04-03 18…	10.0.2.15	10.0.2.22	TCP	69	39908 → 8000 [PSH, ACK] Seq=280193018 Ack=4212556965 Win=2…
40	2020-04-03 18…	10.0.2.22	10.0.2.15	TCP	68	8000 → 39908 [ACK] Seq=4212556965 Ack=280193019 Win=29056 …
41	2020-04-03 18…	10.0.2.15	10.0.2.22	TCP	69	39908 → 8000 [PSH, ACK] Seq=280193019 Ack=4212556965 Win=2…
42	2020-04-03 18…	10.0.2.22	10.0.2.15	TCP	68	8000 → 39908 [ACK] Seq=4212556965 Ack=280193020 Win=29056 …
43	2020-04-03 18…	10.0.2.15	10.0.2.22	TCP	69	39908 → 8000 [PSH, ACK] Seq=280193020 Ack=4212556965 Win=2…
44	2020-04-03 18…	10.0.2.22	10.0.2.15	TCP	68	8000 → 39908 [ACK] Seq=4212556965 Ack=280193021 Win=29056 …

Wireshark output in the webserver

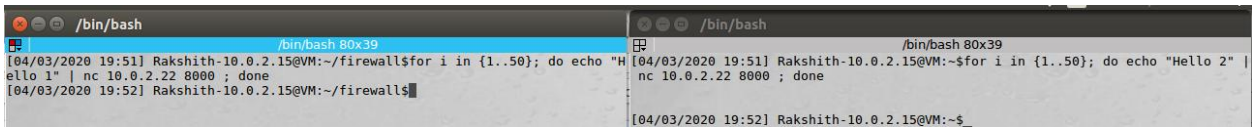


No.	Time	Source	Destination	Protocol	Length	Info
1	2020-04-03…	10.0.2.15	192.168.60.101	TCP	76	39908 → 23 [SYN] Seq=280192915 Win=29200 Len=0 MSS=1460 SACK_PERM=1…
2	2020-04-03…	192.168.60.101	10.0.2.15	TCP	76	23 → 39908 [SYN, ACK] Seq=4212556864 Ack=280192916 Win=28960 Len=0 …
3	2020-04-03…	10.0.2.15	192.168.60.101	TCP	68	39908 → 23 [ACK] Seq=280192916 Ack=4212556865 Win=29312 Len=0 TSval…
4	2020-04-03…	127.0.0.1	127.0.0.1	DNS	84	Standard query 0xe2a6 PTR 15.2.0.10.in-addr.arpa
5	2020-04-03…	127.0.0.1	127.0.0.1	DNS	134	Standard query response 0xe2a6 No such name PTR 15.2.0.10.in-addr.a…
6	2020-04-03…	192.168.60.101	10.0.2.15	TELNET	80	Telnet Data ...
7	2020-04-03…	10.0.2.15	192.168.60.101	TCP	68	39908 → 23 [ACK] Seq=280192916 Ack=4212556877 Win=29312 Len=0 TSval…
8	2020-04-03…	10.0.2.15	192.168.60.101	TELNET	80	Telnet Data ...
9	2020-04-03…	192.168.60.101	10.0.2.15	TCP	68	23 → 39908 [ACK] Seq=4212556877 Ack=280192928 Win=29056 Len=0 TSval…
10	2020-04-03…	192.168.60.101	10.0.2.15	TELNET	92	Telnet Data ...
11	2020-04-03…	10.0.2.15	192.168.60.101	TELNET	125	Telnet Data ...
12	2020-04-03…	192.168.60.101	10.0.2.15	TELNET	83	Telnet Data ...
13	2020-04-03…	10.0.2.15	192.168.60.101	TELNET	92	Telnet Data ...
14	2020-04-03…	192.168.60.101	10.0.2.15	TELNET	71	Telnet Data ...

Task 7 (DNAT): **Use iptables to set up a DNAT for load balancing and demonstrate how it works. In my lecture, I didn't get a perfect load balancing. Can you improve my result?**

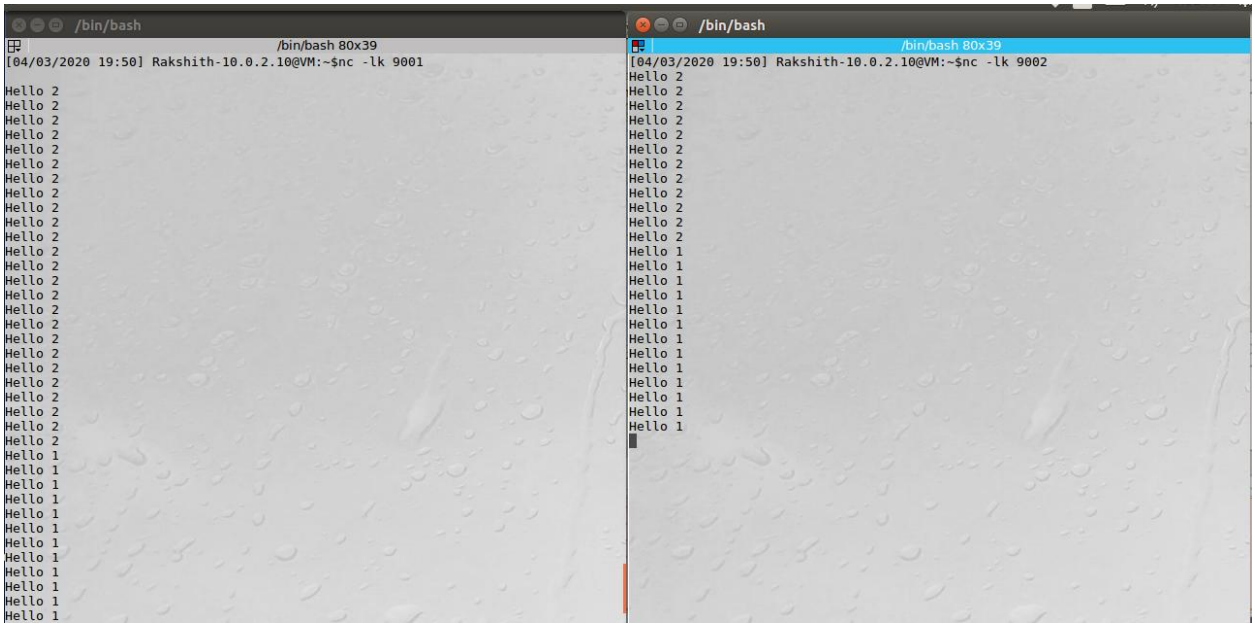
We can use DNAT to set up load balancing, we write two rules in the server as below, we use nth method which enables kernel to use Round robin technique, packet 1 goes to server 1 and packet 2 goes to server 2. When machine A sends packets to server below is the load distribution of packets on two servers in host V.

```
[04/03/2020 19:49] Rakshith-10.0.2.22@VM:~/firewall$sudo iptables -t nat -A PREROUTING -p tcp --dport 8000 -m statistic --mode nth --every 2 --packet 0 -j DNAT --to-destination 192.168.60.101:9001
[04/03/2020 19:49] Rakshith-10.0.2.22@VM:~/firewall$sudo iptables -t nat -A PREROUTING -p tcp --dport 8000 -m statistic --mode nth --every 2 --packet 1 -j DNAT --to-destination 192.168.60.101:9002
[04/03/2020 19:49] Rakshith-10.0.2.22@VM:~/firewall$
```



```
/bin/bash
/bin/bash 80x39
[04/03/2020 19:51] Rakshith-10.0.2.15@VM:~/firewall$for i in {1..50}; do echo "Hello 1" | nc 10.0.2.22 8000 ; done
[04/03/2020 19:52] Rakshith-10.0.2.15@VM:~/firewall$
```

```
/bin/bash
/bin/bash 80x39
[04/03/2020 19:51] Rakshith-10.0.2.15@VM:~$for i in {1..50}; do echo "Hello 2" | nc 10.0.2.22 8000 ; done
[04/03/2020 19:52] Rakshith-10.0.2.15@VM:~$_
```



```
/bin/bash
/bin/bash 80x39
[04/03/2020 19:50] Rakshith-10.0.2.10@VM:~$nc -lk 9001

Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
```

```
/bin/bash
/bin/bash 80x39
[04/03/2020 19:50] Rakshith-10.0.2.10@VM:~$nc -lk 9002
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 2
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
Hello 1
```

Task 8 (Connection Track): Set up a firewall rule based on connections:

First we establish a telnet connection from Machine B to Machine A.

```
[04/03/2020 22:16] Rakshith-10.0.2.22@VM:~/firewall$telnet 10.0.2.15
Trying 10.0.2.15...
Connected to 10.0.2.15.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Fri Apr  3 21:41:07 EDT 2020 from 10.0.2.22 on pts/1
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[04/03/2020 22:16] Rakshith-10.0.2.15@VM:~$
```

```
[04/03/2020 22:18] Rakshith-10.0.2.15@VM:~$sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
[04/03/2020 22:18] Rakshith-10.0.2.15@VM:~$
```

Then we add the iptables rule to accept traffic from established and related connections, and we block all incoming and outgoing TCP traffic.

```
[04/03/2020 22:18] Rakshith-10.0.2.15@VM:~$sudo iptables -A OUTPUT -p tcp -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
[04/03/2020 22:18] Rakshith-10.0.2.15@VM:~$sudo iptables -A OUTPUT -p tcp -j REJECT
[04/03/2020 22:19] Rakshith-10.0.2.15@VM:~$sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A OUTPUT -p tcp -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -j REJECT --reject-with icmp-port-unreachable
[04/03/2020 22:19] Rakshith-10.0.2.15@VM:~$
```

We cannot telnet to Machine B from Machine A, because of the reject rule and the connection was not established before.

```
[04/03/2020 22:19] Rakshith-10.0.2.15@VM:~$telnet 10.0.2.22
Trying 10.0.2.22...
telnet: Unable to connect to remote host: Connection refused
[04/03/2020 22:20] Rakshith-10.0.2.15@VM:~$
```

We can telnet from Machine B to Machine A because our connection was established before, even though all the TCP packets are filtered, we are still able to telnet due to the connection tracking feature.

```
[04/03/2020 22:21] Rakshith-10.0.2.22@VM:~/firewall$telnet 10.0.2.15
Trying 10.0.2.15...
Connected to 10.0.2.15.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Fri Apr  3 22:16:19 EDT 2020 from 10.0.2.22 on pts/1
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[04/03/2020 22:21] Rakshith-10.0.2.15@VM:~$
```