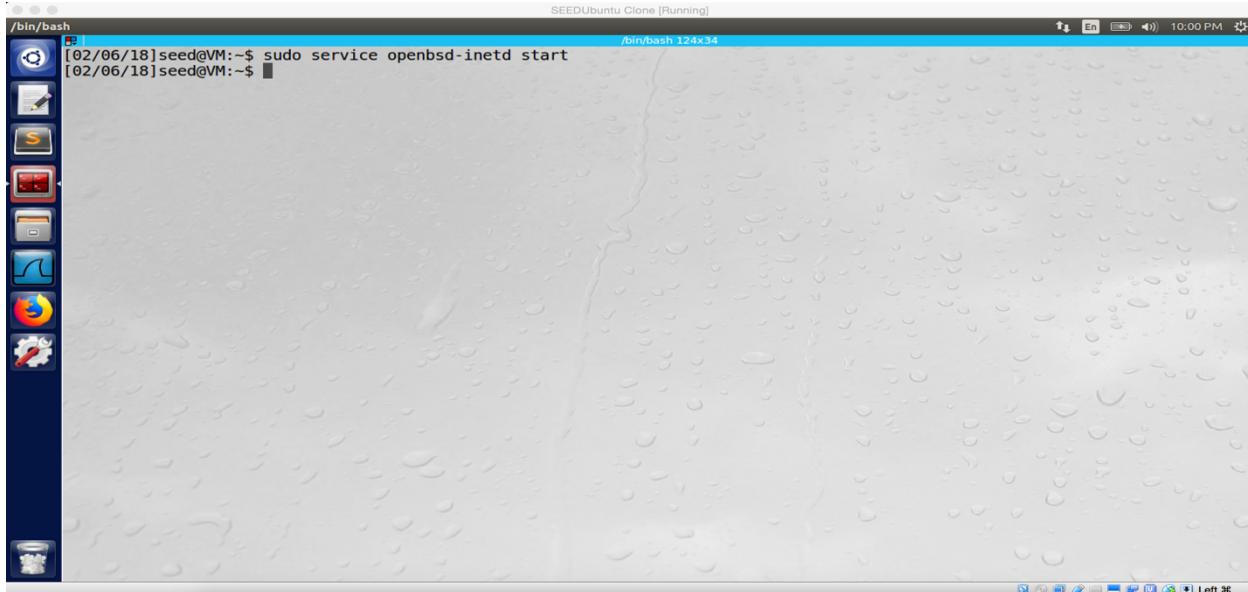


CSE644 LAB2
Yishi Lu
2/13/2018

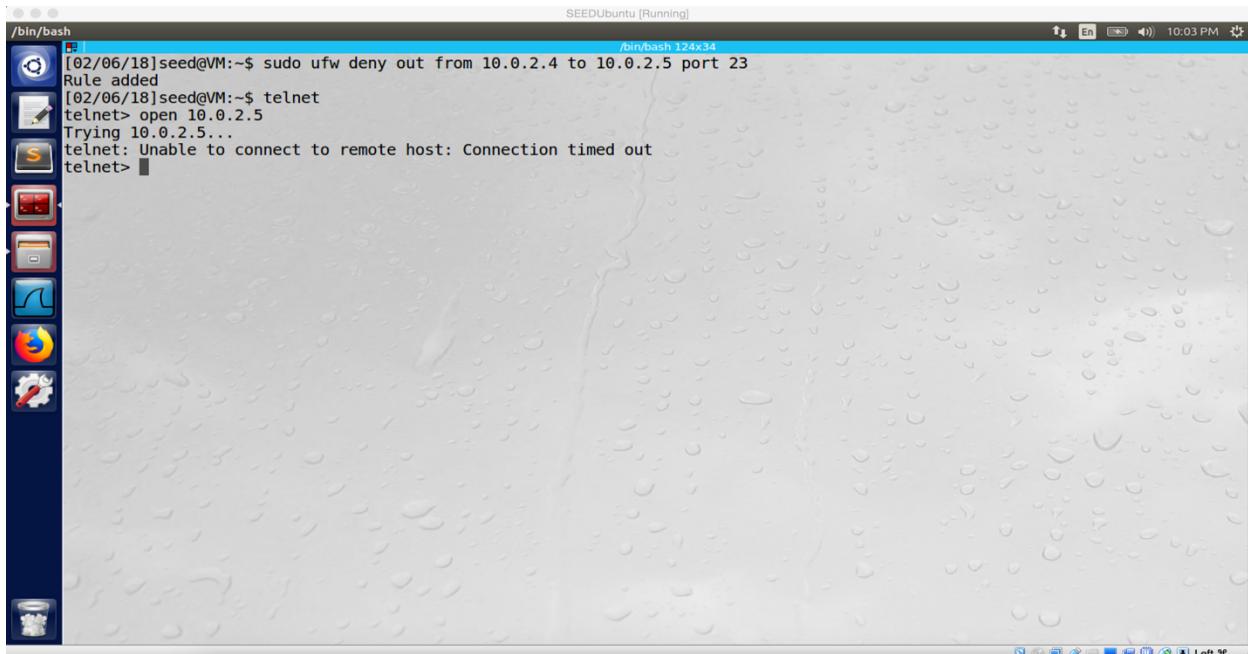
In this lab, I use three machines. The first one is SEEDUbuntu with IP address 10.0.2.4 (VM A), the second one is SEEDUbuntu Clone with IP address 10.0.2.5 (VM B), the third one is SEEDUbuntu Clone2 with IP address 10.0.2.6 (VM C).

Task1

1. Prevent A doing telnet to Machine B

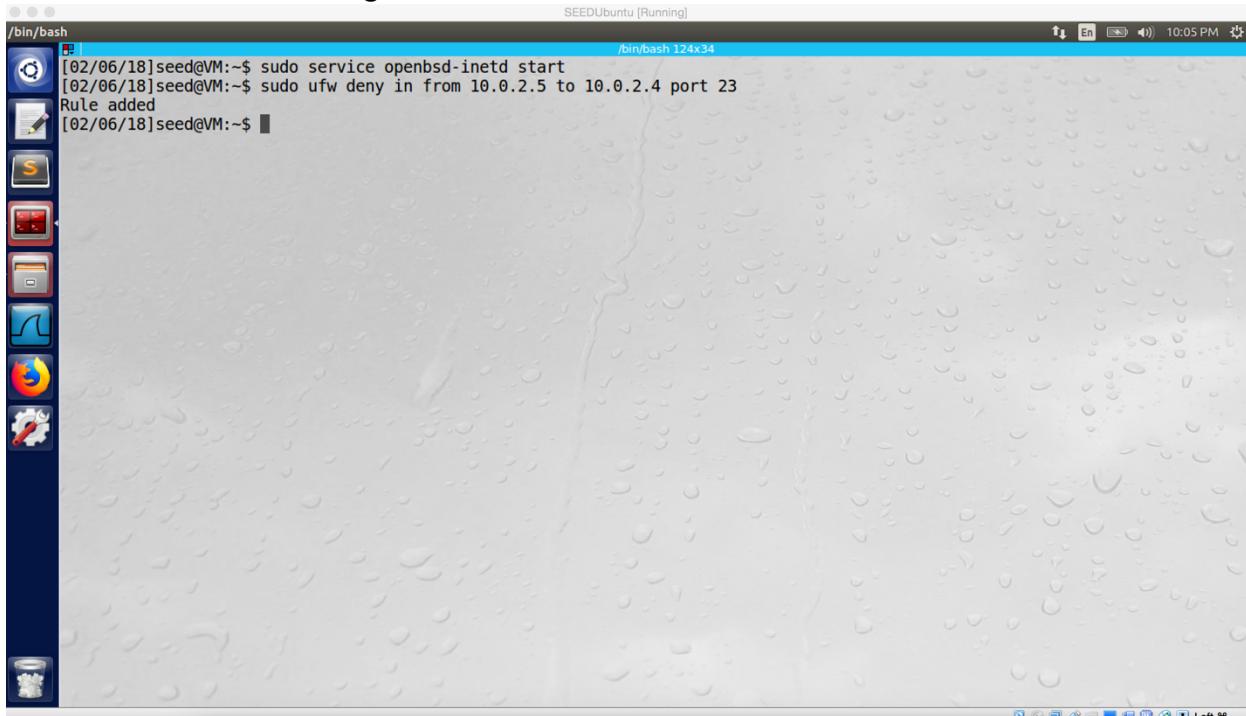


screenshot.1 Start telnet server on VM B



screenshot.2 On VM A. Add rule to prevent VM A doing telnet to VM2 successfully

2. Prevent B from doing telnet to Machine A.

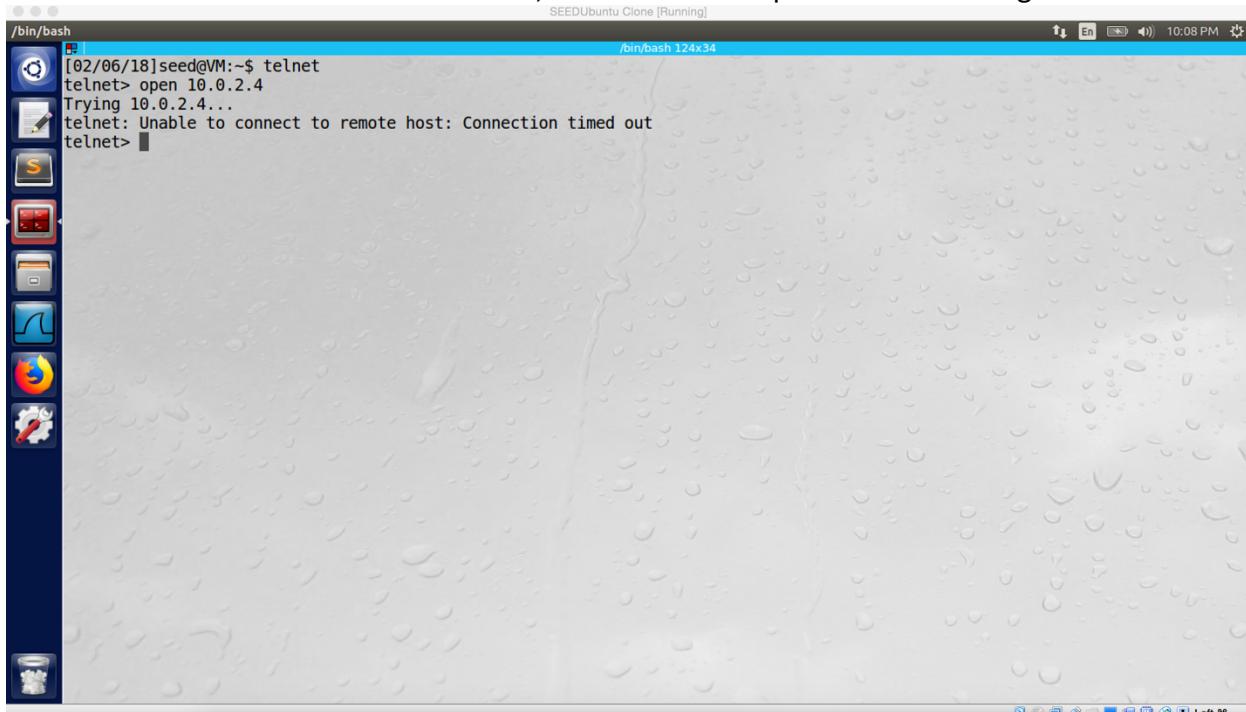


SEEDUbuntu [Running] /bin/bash 124x34

```
[02/06/18]seed@VM:~$ sudo service openbsd-inetd start
[02/06/18]seed@VM:~$ sudo ufw deny in from 10.0.2.5 to 10.0.2.4 port 23
Rule added
[02/06/18]seed@VM:~$
```

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "SEEDUbuntu [Running] /bin/bash 124x34". The command history shows the user starting the "openbsd-inetd" service and then using the "ufw" command to deny incoming traffic on port 23 from IP address 10.0.2.5 to 10.0.2.4. The desktop interface includes a vertical application menu on the left and a dock at the bottom.

screenshot.3 On VM A. Start telnet sever, and I add rule to prevent VM B doing telnet to VM A.



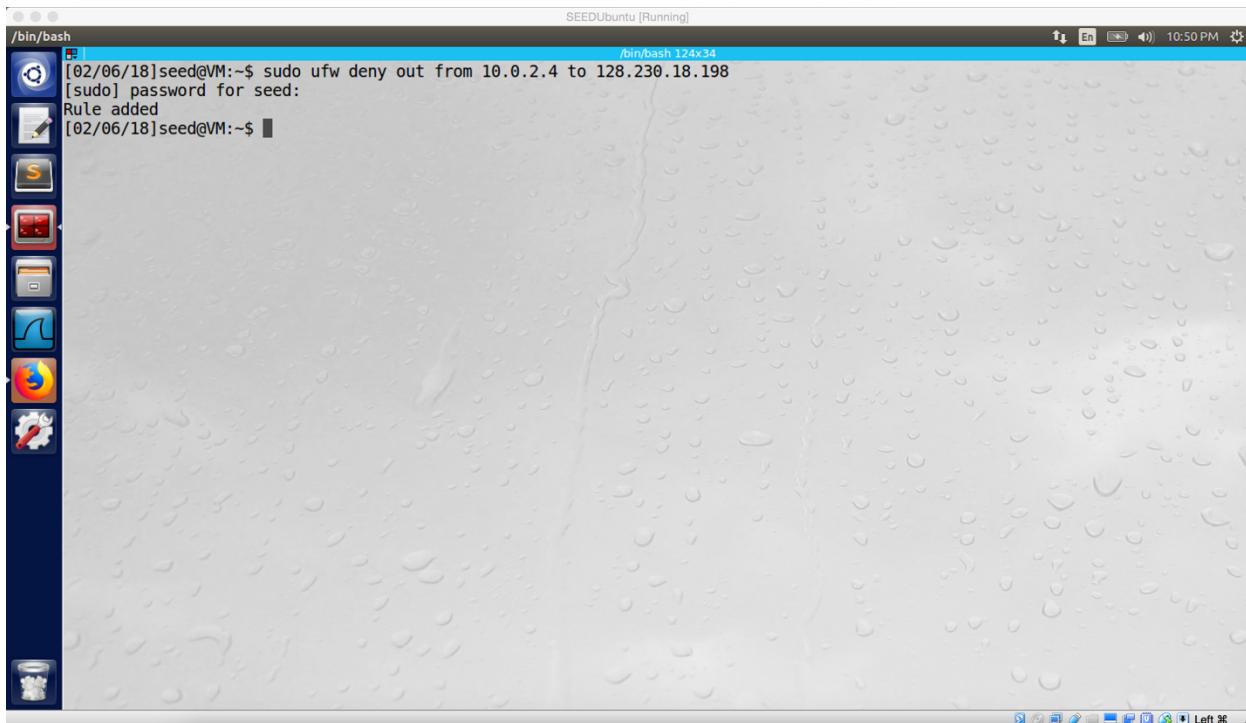
SEEDUbuntu Clone [Running] /bin/bash 124x34

```
[02/06/18]seed@VM:~$ telnet
telnet> open 10.0.2.4
Trying 10.0.2.4...
telnet: Unable to connect to remote host: Connection timed out
telnet>
```

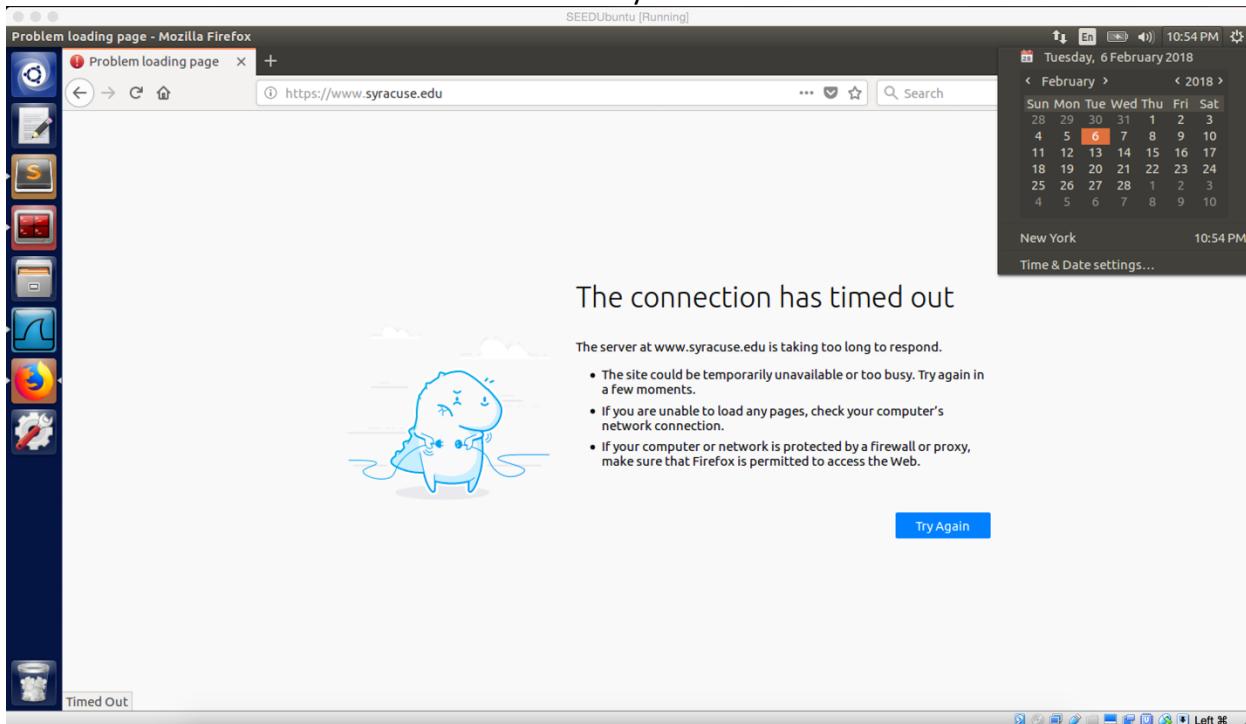
The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "SEEDUbuntu Clone [Running] /bin/bash 124x34". The user runs a "telnet" command and attempts to connect to IP address 10.0.2.4. The connection fails with the message "telnet: Unable to connect to remote host: Connection timed out". The desktop interface includes a vertical application menu on the left and a dock at the bottom.

screenshot.4 The rule is added successfully, VM B cannot telnet to VM A

3. Prevent A from visiting <https://www.syracuse.edu>



screenshot.5 On VM A. Add rule to block www.syracuse.edu



screenshot.6 Successfully blocked this website.

Observation:

For this task, I use ufw to add three rules on the firewall of VM A. The first rule is to prevent VM A doing telnet to VM B (as screenshot 1 and 2 show). The second rule is to prevent

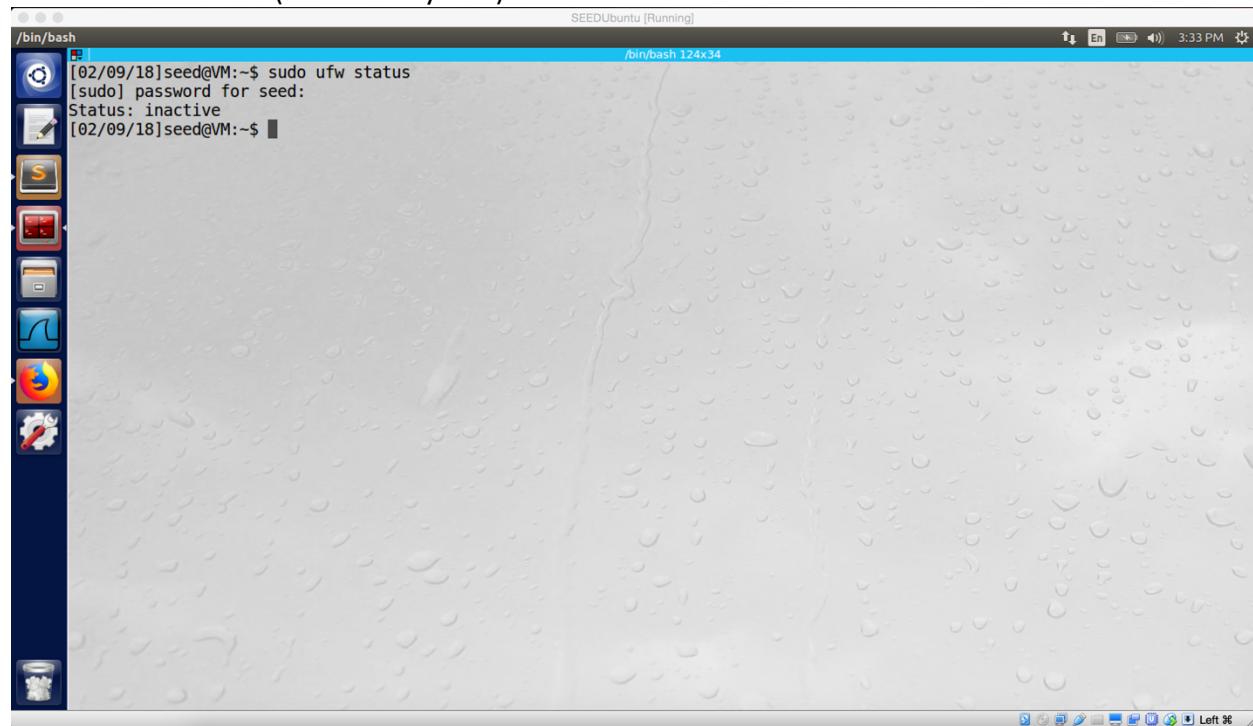
VM B doing telnet to VM A (as screenshot 3 and 4 show). And the last rule is to prevent VM A to visit www.syracuse.edu (as screenshot 5 and 6 show).

Explanation:

Firewall is a part of computer system, and it has two main functions. The first one is egress. Egress, when the firewall is used to prevent internal user to reach outside. In step 1 and step 3, I did egress. In step 1, I add rule of “sudo ufw deny out from 10.0.2.4 to 10.0.2.5 port 23”, it means ufw prevent user on 10.0.2.4 (VM A) to visit 10.0.2.5 (VM B) by port 23. Because telnet use port 23 by default, it rejects telnetting from VM A to VM B. In step 3, I add rule of “sudo ufw deny out from 10.0.2.4 to 128.230.18.198”, this means ufw prevent user on 10.0.2.4 (VM A) to visit 128.230.18.198 (IP address of SU main website). Ingress, when the firewall is used to prevent attack or connection from outside. In step 2, I did ingress. In this part, I add rule of “sudo ufw deny in from 10.0.2.5 to 10.0.2.4 port 23”, compare to step 1, I change out to in and swap 10.0.2.4 with 10.0.2.5. This rule will reject any telnetting from VM B with port 23.

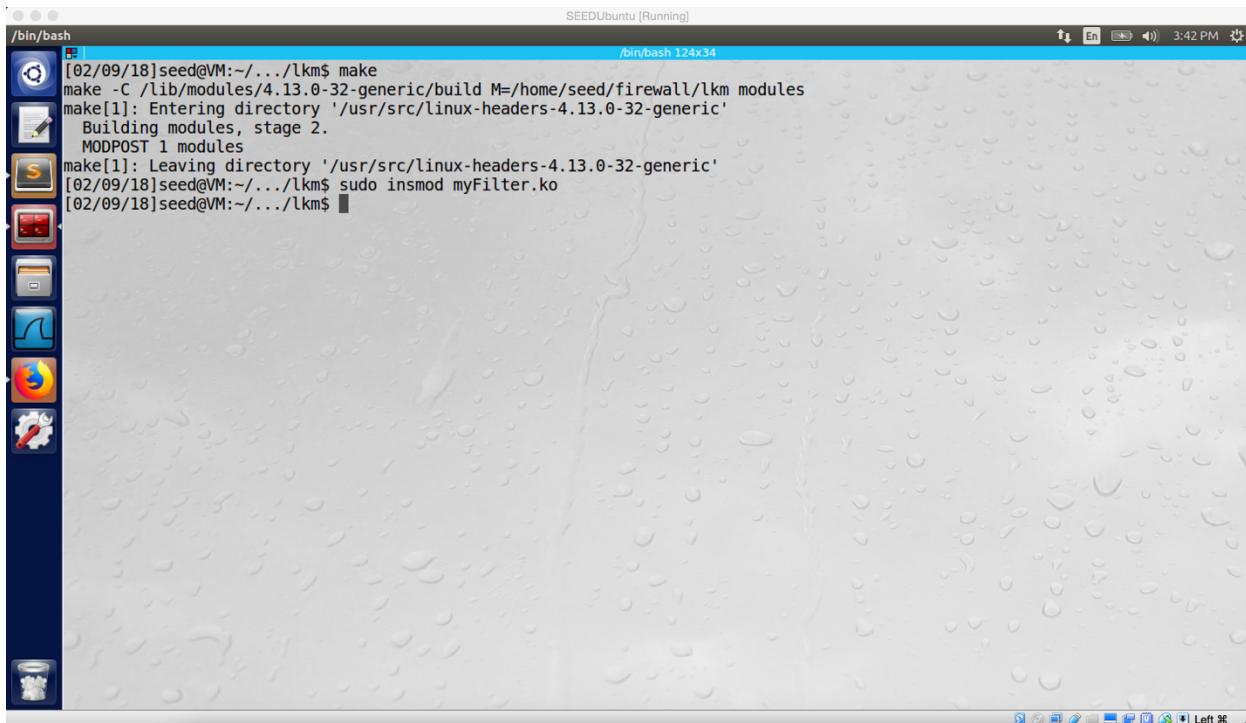
Task2

In this task, I created a module which contains 5 rules. First, prevent VM A do telnet VM B. Second, prevent VM B to telnet VM A. Third, prevent VM A to visit SU main website (www.syracuse.edu). Forth, dropping any ICMP packet from VM B. Last, prevent VM A to visit CUNY main website (www2.cuny.edu).



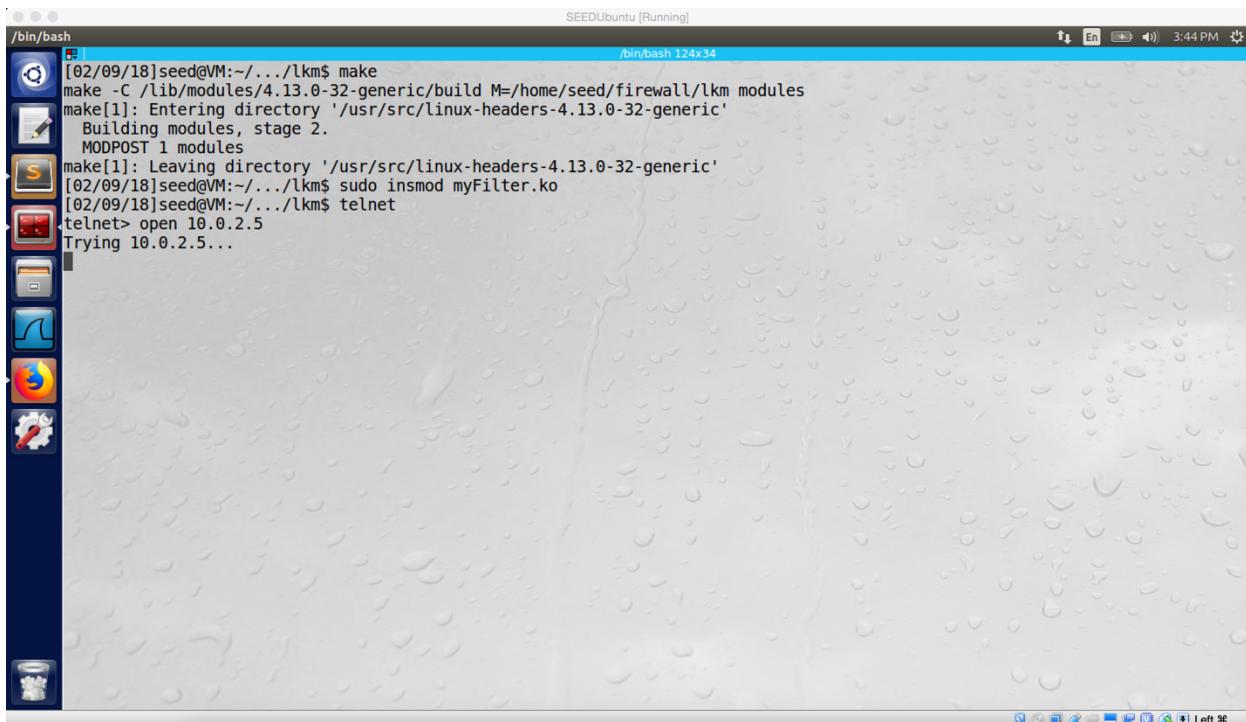
```
SEEDUbuntu [Running]
/bin/bash 124x34
[02/09/18]seed@VM:~$ sudo ufw status
[sudo] password for seed:
Status: inactive
[02/09/18]seed@VM:~$
```

For this task, I disable the ufw firewall firstly



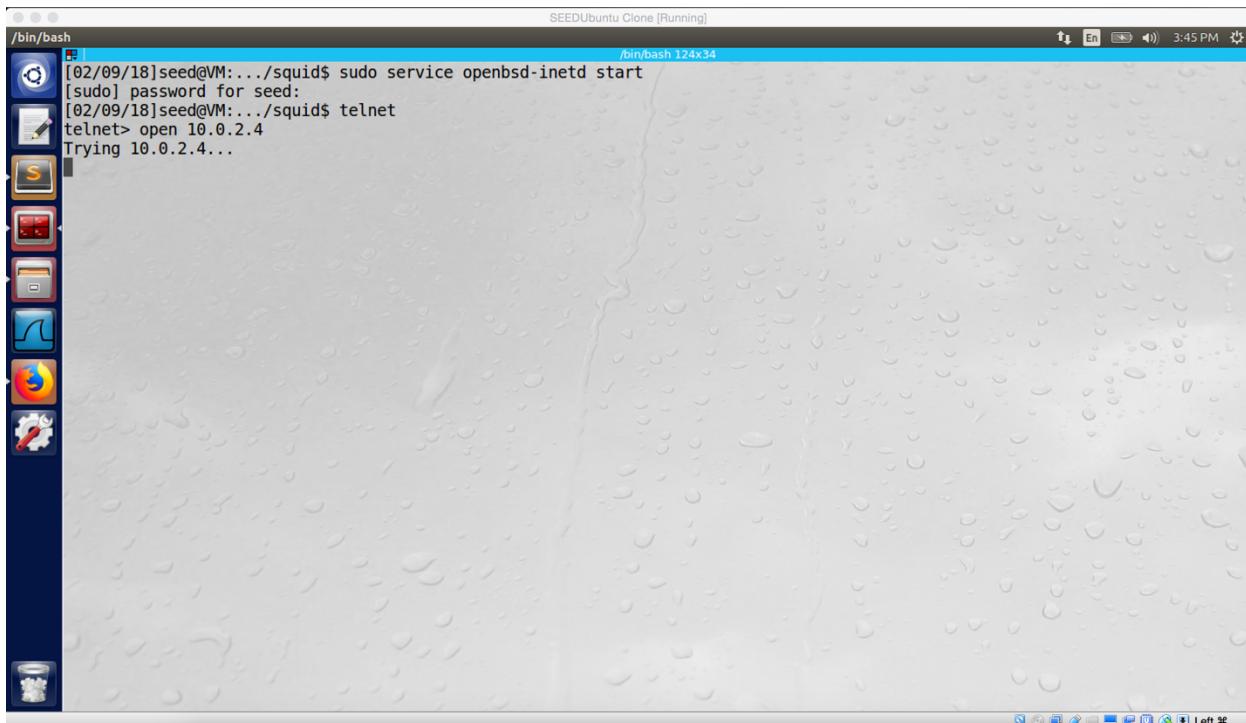
```
SEEDUbuntu [Running]
/bin/bash 124x34
[02/09/18]seed@VM:~/.../lkm$ make
make -C /lib/modules/4.13.0-32-generic/build M=/home/seed/firewall/lkm modules
make[1]: Entering directory '/usr/src/linux-headers-4.13.0-32-generic'
  Building modules, stage 2.
    MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.13.0-32-generic'
[02/09/18]seed@VM:~/.../lkm$ sudo insmod myFilter.ko
[02/09/18]seed@VM:~/.../lkm$
```

Insert my module to kernel

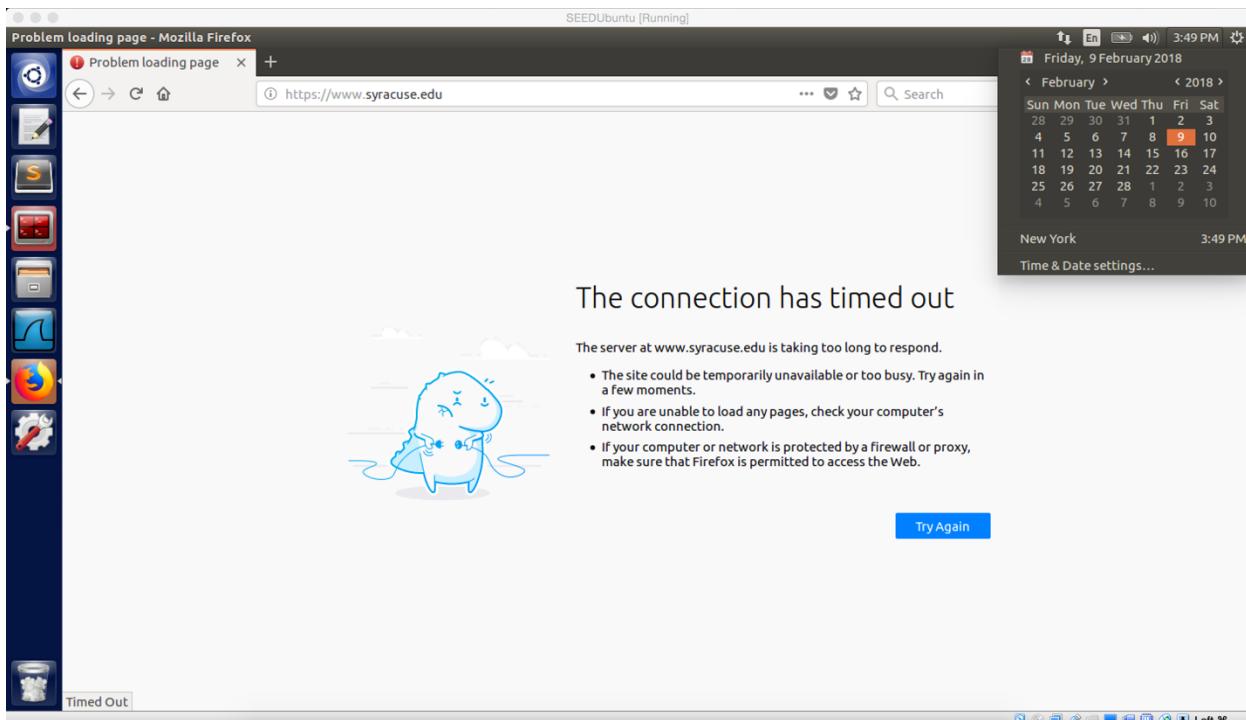


```
SEEDUbuntu [Running]
/bin/bash 124x34
[02/09/18]seed@VM:~/.../lkm$ make
make -C /lib/modules/4.13.0-32-generic/build M=/home/seed/firewall/lkm modules
make[1]: Entering directory '/usr/src/linux-headers-4.13.0-32-generic'
  Building modules, stage 2.
    MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.13.0-32-generic'
[02/09/18]seed@VM:~/.../lkm$ sudo insmod myFilter.ko
[02/09/18]seed@VM:~/.../lkm$ telnet
telnet: open 10.0.2.5...
Trying 10.0.2.5...
```

For rule 1, the host machine VM A cannot telnet VM B



For rule 2, the VM B cannot telnet VM A



For rule 3, successfully prevent VM A to visit SU main website

SEEDUbuntu Clone [Running] /bin/bash 124x34

```
[02/09/18]seed@VM:.../squid$ sudo service openbsd-inetd start
[sudo] password for seed:
[02/09/18]seed@VM:.../squid$ telnet
telnet> open 10.0.2.4
Trying 10.0.2.4...
telnet: Unable to connect to remote host: Connection timed out
telnet> ^Z
[4]+  Stopped                  telnet
[02/09/18]seed@VM:.../squid$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
```

SEEDUbuntu Clone [Running] *any 3:52 PM

Apply a display filter... <Ctrl>

No.	Time	Source	Destination	Protocol	Length	Info
1	2018-02-09 15:52:25...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=1/256, ttl=64 (no response found!)
2	2018-02-09 15:52:26...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=2/512, ttl=64 (no response found!)
3	2018-02-09 15:52:27...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=3/768, ttl=64 (no response found!)
4	2018-02-09 15:52:28...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=4/1024, ttl=64 (no response found!)
5	2018-02-09 15:52:29...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=5/1280, ttl=64 (no response found!)
6	2018-02-09 15:52:30...	PcsCompu...		ARP	64	Who has 10.0.2.4? Tell 10.0.2.5
7	2018-02-09 15:52:30...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=6/1536, ttl=64 (no response found!)
8	2018-02-09 15:52:30...	PcsCompu...		ARP	62	10.0.2.4 is at 08:00:27:06:26:a9
9	2018-02-09 15:52:31...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=7/1792, ttl=64 (no response found!)
10	2018-02-09 15:52:32...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=8/2048, ttl=64 (no response found!)
11	2018-02-09 15:52:33...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=9/2304, ttl=64 (no response found!)
12	2018-02-09 15:52:34...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=10/2560, ttl=64 (no response found!)
13	2018-02-09 15:52:35...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=11/2816, ttl=64 (no response found!)
14	2018-02-09 15:52:36...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=12/3072, ttl=64 (no response found!)
15	2018-02-09 15:52:37...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=13/3328, ttl=64 (no response found!)
16	2018-02-09 15:52:38...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=14/3584, ttl=64 (no response found!)
17	2018-02-09 15:52:39...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=0x0927, seq=15/3840, ttl=64 (no response found!)
18	2018-02-09 15:52:40...	10.0.2.5	10.0.2.4	ICMP	100	Echo (ping) request id=16/4096, ttl=64 (no response found!)

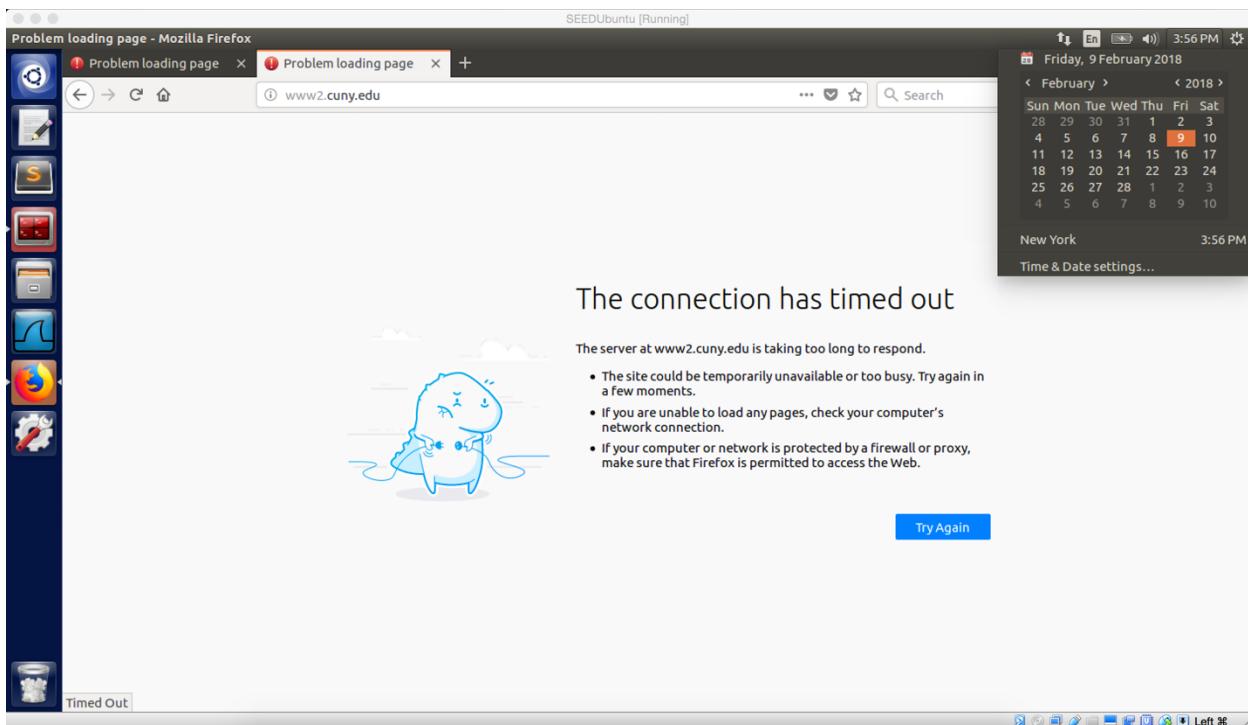
Frame 13: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0
 ► Linux cooked capture
 ► Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.4
 ► Internet Control Message Protocol

0000 00 04 00 01 00 06 08 00 27 74 a9 f5 00 00 08 00'.....
0010 45 00 00 54 20 ca 40 00 01 01 d7 0a 00 02 05	E.T. @. @.....
0020 0a 00 02 04 08 00 8e 42 09 27 00 0b 13 0a 7e 5aB.....Z
0030 d8 23 0c 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13	.#.....
0040 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 !#
0050 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33	\$%`()*+, ./0123
0060 34 35 36 37	4567

/tmp/wireshark_any_20180209155219_7v1xPC.pcapng (2440 bytes)

Packets: 18 - Displayed: 18 (100.0%) Profile: Default

For rule 4, ICMP request from VM B to VM A does not success, there is no reply send back from VM A.



For rule 5, VM A cannot visit www2.cuny.edu

Observation:

Firstly, I disable the ufw firewall, and then I insert my module into kernel by LKM. As the screenshots showed above. The rule 1, rule 2 and rule 3 have the same effect as the ufw firewall rules which we did in task 1. Rule 4 is a new rule; this rule will prevent ICMP packet from VM B. The last rule is similar to rule 3, it blocks www2.cuny.edu, so the user on the host machine cannot visit this website.

Explanation:

This task is done by two main mechanisms, LKM (loadable kernel module) and Netfilter. The LKM allows us to add new module into kernel. Netfilter provides several hooks in incoming and outgoing paths, so we can insert our module (by LKM) into these hooks, and then we can manipulate packets when they pass these hooks. For rule 1, I block any TCP packet with port 23 and destination IP address 10.0.2.5, so VM A is not able to telnet VM B; moreover, we should set hooknum to NF_INET_POST_ROUTING. For rule 2, I block any TCP packet with port 23, and source IP address 10.0.2.5, so VM B cannot telnet the VM A; moreover, we should set hooknum to NF_INET_PRE_ROUTING. For rule 3, I block packets with IP address of 128.230.18.198 (SU website address), so the host machine cannot visit SU website; moreover, we should set hooknum to NF_INET_POST_ROUTING. For rule 4, I block any ICMP packet with source IP address of 10.0.2.5, so any ICMP packet from VM B will be discarded; moreover, we should set hooknum to NF_INET_PRE_ROUTING. For rule 5, I block packets with IP address of 128.228.0.52, so any packet from such IP address will be blocked; moreover, we should set hooknum to NF_INET_POST_ROUTING.

setUpFilter, and removeFilter are same as professor's code. However, we may change hooknum when we want to apply filter. Because we should insert modules into different hook by doing different filtering (ingress, egress).

```
//get source IP address in char array
sprintf(ip_dst, "%d.%d.%d.%d", ((unsigned char *)&iph->daddr)[0], ((unsigned char *)&iph->daddr)[1],
        ((unsigned char *)&iph->daddr)[2], ((unsigned char *)&iph->daddr)[3]);
//get destination IP address in char array
sprintf(ip_src, "%d.%d.%d.%d", ((unsigned char *)&iph->saddr)[0], ((unsigned char *)&iph->saddr)[1],
        ((unsigned char *)&iph->saddr)[2], ((unsigned char *)&iph->saddr)[3]);

//prevent VM A telnet VM B
if (iph->protocol == IPPROTO_TCP && (tcpiph->dest == htons(23) || tcpiph->source == htons(23))&&
strcmp(ip_dst, "10.0.2.5") == 0) {
    printk(KERN_INFO "Dropping telnet packet to %s\n", ip_dst);
    return NF_DROP;
}

//prevent VM B telnet VM A
if (iph->protocol == IPPROTO_TCP && (tcpiph->dest == htons(23) || tcpiph->source == htons(23))&&
strcmp(ip_src, "10.0.2.5") == 0) {
    printk(KERN_INFO "Dropping telnet packet from %s\n", ip_src);
    return NF_DROP;
}

//prevent host machine to visit www.syracuse.edu
if (strcmp(ip_dst, "128.230.18.198") == 0) {
    printk(KERN_INFO "Dropping packet to %s\n", ip_dst);
    return NF_DROP;
}

//prevent ICMP request/reply from VM B (10.0.2.5)
if (iph->protocol == IPPROTO_ICMP && strcmp(ip_src, "10.0.2.5") == 0) {
    printk(KERN_INFO "Dropping ICMP packet from VM B %s", ip_src);
    return NF_DROP;
}

//prevent host machine to visit www2.cuny.edu
if (strcmp(ip_dst, "128.228.0.52") == 0) {
    printk(KERN_INFO "Dropping packet to %s\n", ip_dst);
    return NF_DROP;
}
```

Question 1:

Netfilter defines five hooks for IPv4, they are

NF_IP_PRE_ROUTING: All incoming packets will pass this hook before any routing decision is made.

NF_IP_LOCAL_IN: When the routing decides that the following packet is for the host machine, and then the packet will pass this hook.

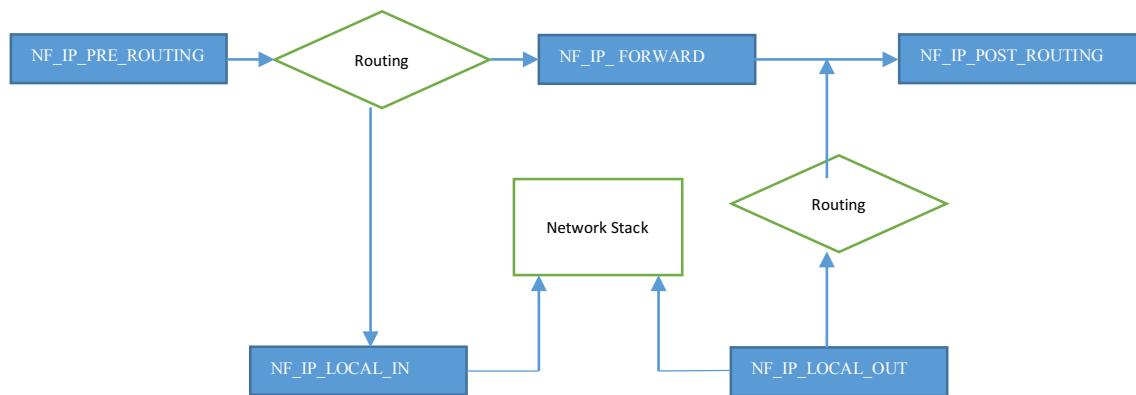
NF_IP_FORWARD: If the following packet is not for the host, and then the packet will forward to other hosts and reach this hook.

NF_IP_LOCAL_OUT: Packets generated by the local host will sent to this hook first.

NF_IP_POST_ROUTING: When a packet is going out of the host, it will pass this hook.

We can insert our module program into these hooks. When packets pass these hooks, our module program will get involved. As a result, we can modify or filter these packets.

We can insert our program into these hooks. Therefore, when a packet is arrived in the hook, our program will be involved. As a result, we can block or modify the packet.



Question 2:

For ingress, we want to inspect incoming packet, so we should place a hook before the packet is being routed or processed, such as **NF_IP_PRE_ROUTING**.

For egress, we want to inspect outgoing packet, so we should place a hook after a packet is being routed or processed, such as **NF_IP_POST_ROUTING**.

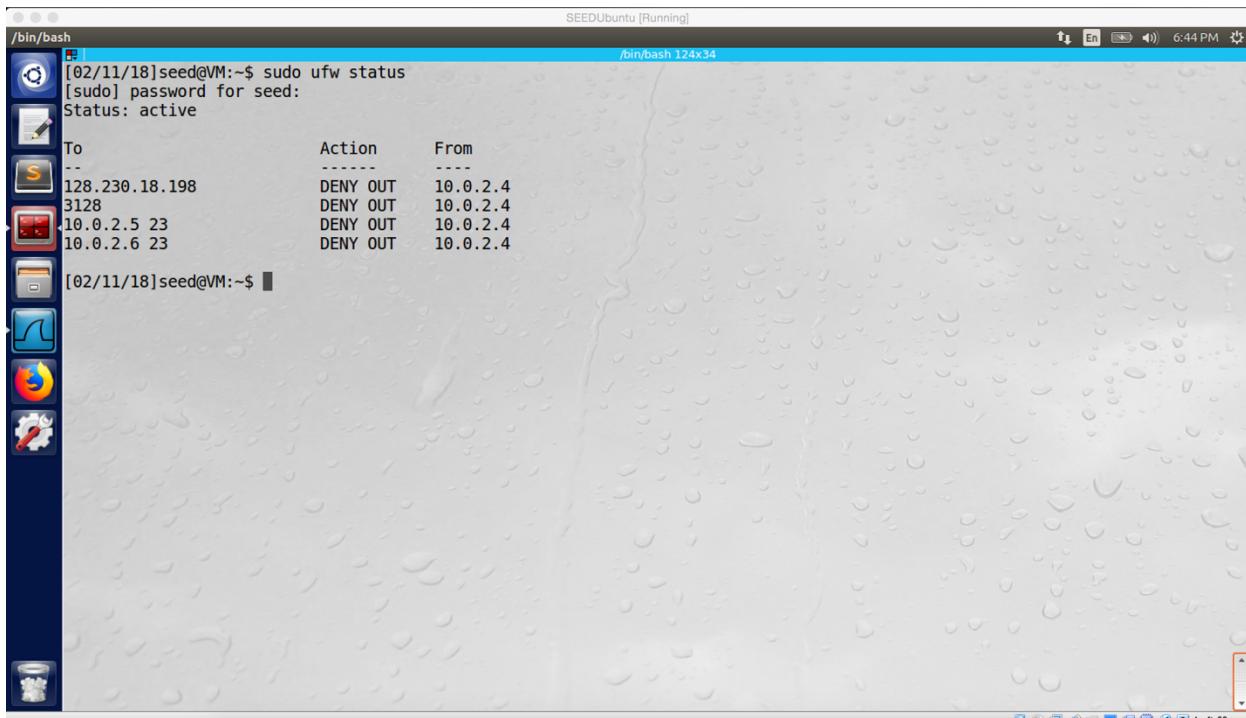
Question 3:

Yes, by placing our filter program on hooks, we can block/filtering packets, and we also can modify packets.

Task3

For this task, I choose to use a website with static IP address which is www.syracuse.com

Task 3.a



SEEDUbuntu [Running] /bin/bash 124x34

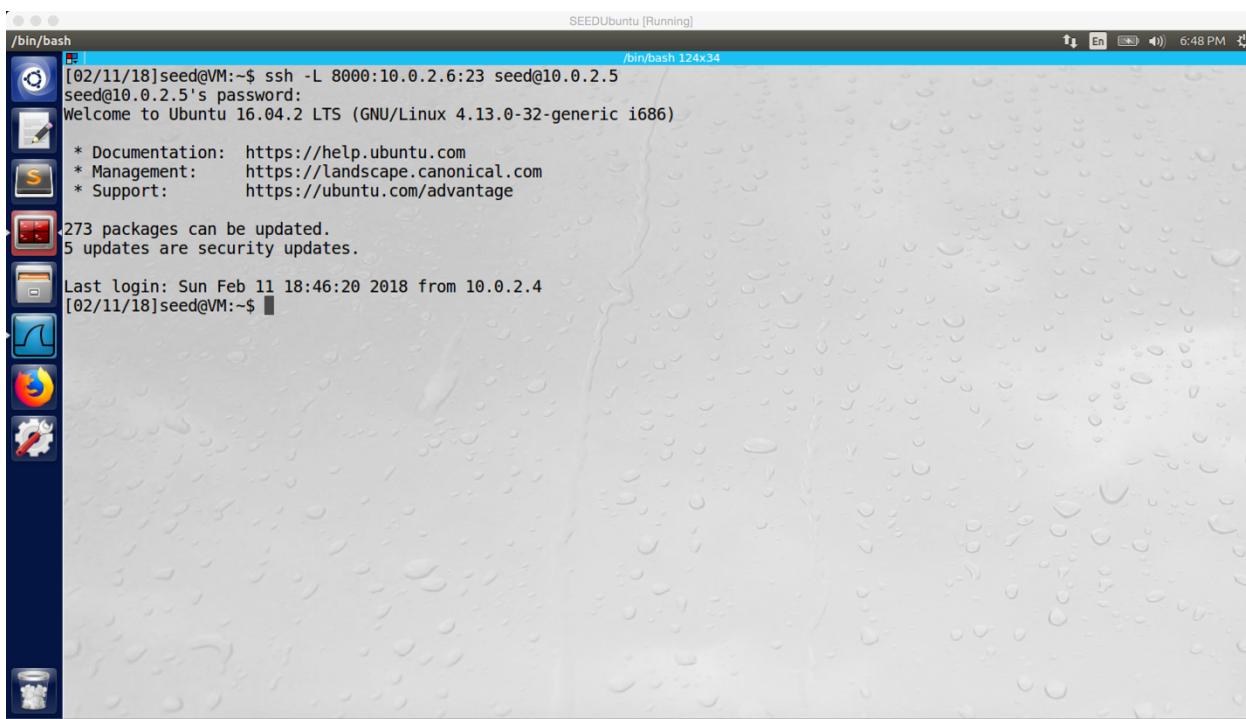
```
[02/11/18]seed@VM:~$ sudo ufw status
[sudo] password for seed:
Status: active

To                         Action      From
--                         --          --
128.230.18.198            DENY OUT   10.0.2.4
3128                      DENY OUT   10.0.2.4
10.0.2.5 23               DENY OUT   10.0.2.4
10.0.2.6 23               DENY OUT   10.0.2.4

[02/11/18]seed@VM:~$
```

The screenshot shows a desktop environment with a terminal window open. The terminal window title is 'SEEDUbuntu [Running]' and the command run is 'sudo ufw status'. The output shows that the firewall is active and denies traffic from 10.0.2.4 to 128.230.18.198 on port 3128 and port 23. The desktop interface includes a vertical application menu on the left and a dock with icons for various applications at the bottom.

Screenshot 1 on VM A. Adding ufw rule to block VM A telnet VM C successfully



SEEDUbuntu [Running] /bin/bash 124x34

```
[02/11/18]seed@VM:~$ ssh -L 8000:10.0.2.6:23 seed@10.0.2.5
seed@10.0.2.5's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-32-generic i686)

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

273 packages can be updated.
5 updates are security updates.

Last login: Sun Feb 11 18:46:20 2018 from 10.0.2.4
[02/11/18]seed@VM:~$
```

The screenshot shows a desktop environment with a terminal window open. The terminal window title is 'SEEDUbuntu [Running]' and the command run is 'ssh -L 8000:10.0.2.6:23 seed@10.0.2.5'. The output shows the connection to VM B (seed@10.0.2.5) and the system information of VM B (Ubuntu 16.04.2 LTS). The desktop interface includes a vertical application menu on the left and a dock with icons for various applications at the bottom.

Screenshot 2 on VM A. Successfully build SSH tunnel from VM A to VM B

```
/bin/bash
[02/11/18]seed@VM:~$ ssh -L 8000:10.0.2.6:23 seed@10.0.2.5
seed@10.0.2.5's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-32-generic i686)

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

273 packages can be updated.
5 updates are security updates.

Last login: Sun Feb 11 18:46:20 2018 from 10.0.2.4
[02/11/18]seed@VM:~$ telnet localhost 8000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sun Feb 11 18:46:46 EST 2018 from 10.0.2.5 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-32-generic i686)

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

273 packages can be updated.
5 updates are security updates.

[02/11/18]seed@VM:~$
```

Screenshot 3 on VM A. Successfully telnetting from VM A to VM C by SSH tunnel

Wireshark Screenshot showing network traffic for an SSH session. The traffic consists of numerous encrypted packets (SSHv2) exchanged over port 22. The interface is 'mon0' and the packet count is 249.

No.	Time	Source	Destination	Protocol	Length	Info
17	2018-02-11 18:48:13...	10.0.2.4	10.0.2.5	SSHv2	112	Client: Encrypted packet (len=44)
18	2018-02-11 18:48:13...	10.0.2.5	10.0.2.4	SSHv2	112	Server: Encrypted packet (len=44)
20	2018-02-11 18:48:13...	10.0.2.4	10.0.2.5	SSHv2	128	Client: Encrypted packet (len=60)
21	2018-02-11 18:48:13...	10.0.2.5	10.0.2.4	SSHv2	120	Server: Encrypted packet (len=52)
23	2018-02-11 18:48:15...	10.0.2.4	10.0.2.5	SSHv2	152	Client: Encrypted packet (len=84)
24	2018-02-11 18:48:15...	10.0.2.5	10.0.2.4	SSHv2	96	Server: Encrypted packet (len=28)
26	2018-02-11 18:48:15...	10.0.2.4	10.0.2.5	SSHv2	180	Client: Encrypted packet (len=112)
28	2018-02-11 18:48:15...	10.0.2.5	10.0.2.4	SSHv2	1008	Server: Encrypted packet (len=940)
30	2018-02-11 18:48:15...	10.0.2.4	10.0.2.5	SSHv2	112	Client: Encrypted packet (len=44)
32	2018-02-11 18:48:15...	10.0.2.4	10.0.2.5	SSHv2	512	Client: Encrypted packet (len=444)
34	2018-02-11 18:48:15...	10.0.2.5	10.0.2.4	SSHv2	176	Server: Encrypted packet (len=108)
35	2018-02-11 18:48:15...	10.0.2.5	10.0.2.4	SSHv2	440	Server: Encrypted packet (len=372)
37	2018-02-11 18:48:15...	10.0.2.4	10.0.2.5	SSHv2	128	Server: Encrypted packet (len=60)
40	2018-02-11 18:48:28...	10.0.2.4	10.0.2.5	SSHv2	112	Client: Encrypted packet (len=44)
41	2018-02-11 18:48:28...	10.0.2.5	10.0.2.4	SSHv2	128	Server: Encrypted packet (len=60)
43	2018-02-11 18:48:28...	10.0.2.4	10.0.2.5	SSHv2	184	Client: Encrypted packet (len=36)
50	2018-02-11 18:48:28...	10.0.2.5	10.0.2.4	SSHv2	128	Server: Encrypted packet (len=60)
51	2018-02-11 18:48:28...	10.0.2.5	10.0.2.4	SSHv2	128	Server: Encrypted packet (len=60)
57	2018-02-11 18:48:28...	10.0.2.5	10.0.2.4	SSHv2	184	Server: Encrypted packet (len=36)
58	2018-02-11 18:48:28...	10.0.2.5	10.0.2.4	SSHv2	128	Server: Encrypted packet (len=60)
61	2018-02-11 18:48:28...	10.0.2.5	10.0.2.4	SSHv2	184	Server: Encrypted packet (len=36)

Frame 17: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface 0

Linux cooked capture

Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.5

Transmission Control Protocol, Src Port: 57736, Dst Port: 22, Seq: 836190218, Ack: 1132917638, Len: 44

SSH Protocol

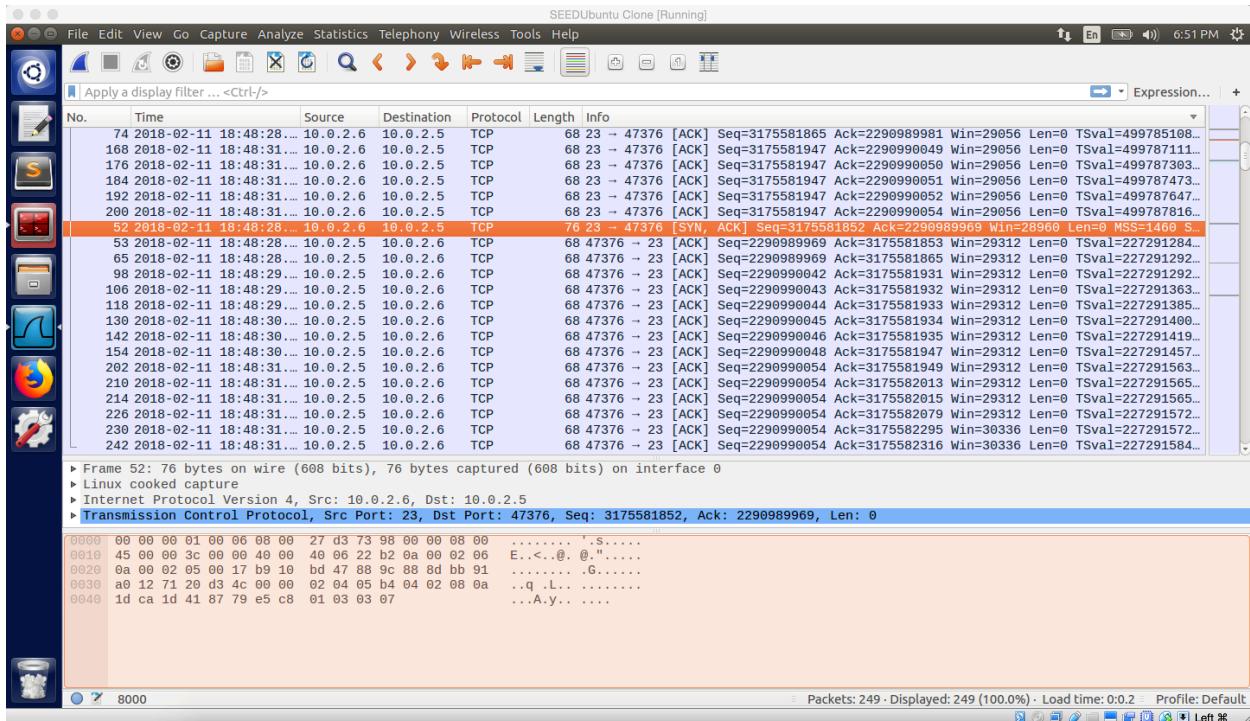
Hex Dump (0000-00FF): 00 00 00 01 00 00 08 00 27 06 26 a9 00 00 08 00&....
0010 45 00 00 60 31 e5 40 00 40 06 f0 aa 00 02 04 E. 1.0. @....
0020 00 00 02 05 1e 88 00 16 31 d7 40 04 86 f3 86 1.0.C...
0030 88 18 01 03 5d ab 00 00 01 01 08 00 f7 82 47 76]....Gv
0040 82 0c 44 d7 17 45 13 4e e0 20 c3 57 51 14 52 e1 ..D.E.N. .WQ.R.
0050 32 98 ab c2 59 56 49 78 71 a5 08 a3 9f 1a 94 2...YVIX q....
0060 86 1a 12 6f a5 60 0e ce 8a 7f 92 75 e4 bb ff cc ..o.j.u....

Invalid filter: "10.0.2.4" is neither a field nor a protocol name.

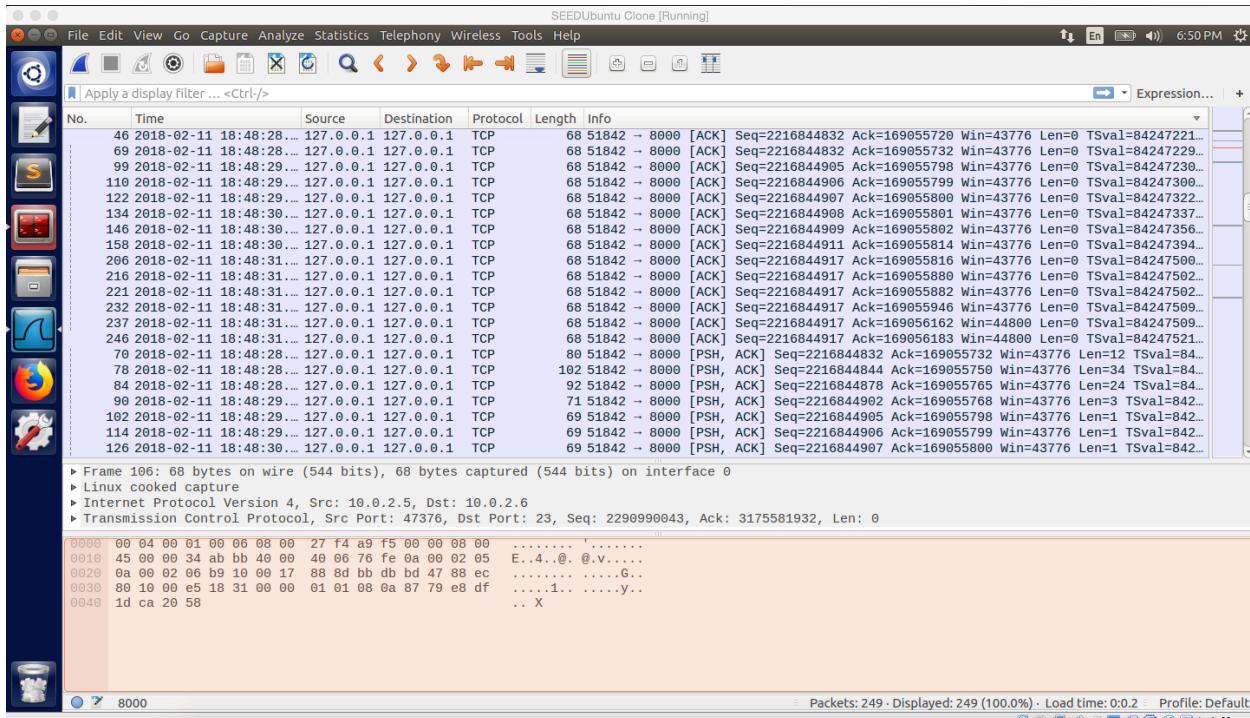
Packets: 249 - Displayed: 249 (100.0%)

Profile: Default

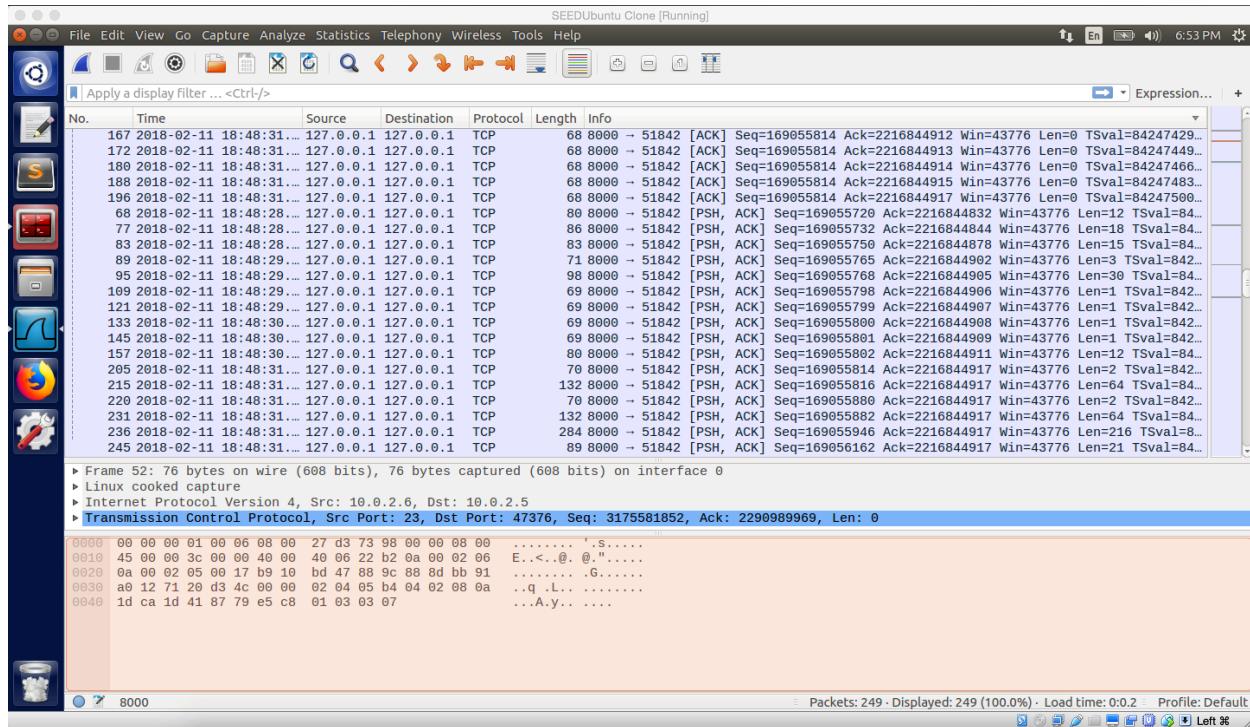
screenshot 4. SSH tunnel is established between VM A and VM B.



screenshot 5.1. TCP packets transfer between VM B and VM C. VM C received and send out TCP packets by port 23



screenshot 5.2. TCP packet go through port 8000



screenshot 5.3, TCP packets are going through port 8000.

Observation and Explanation:

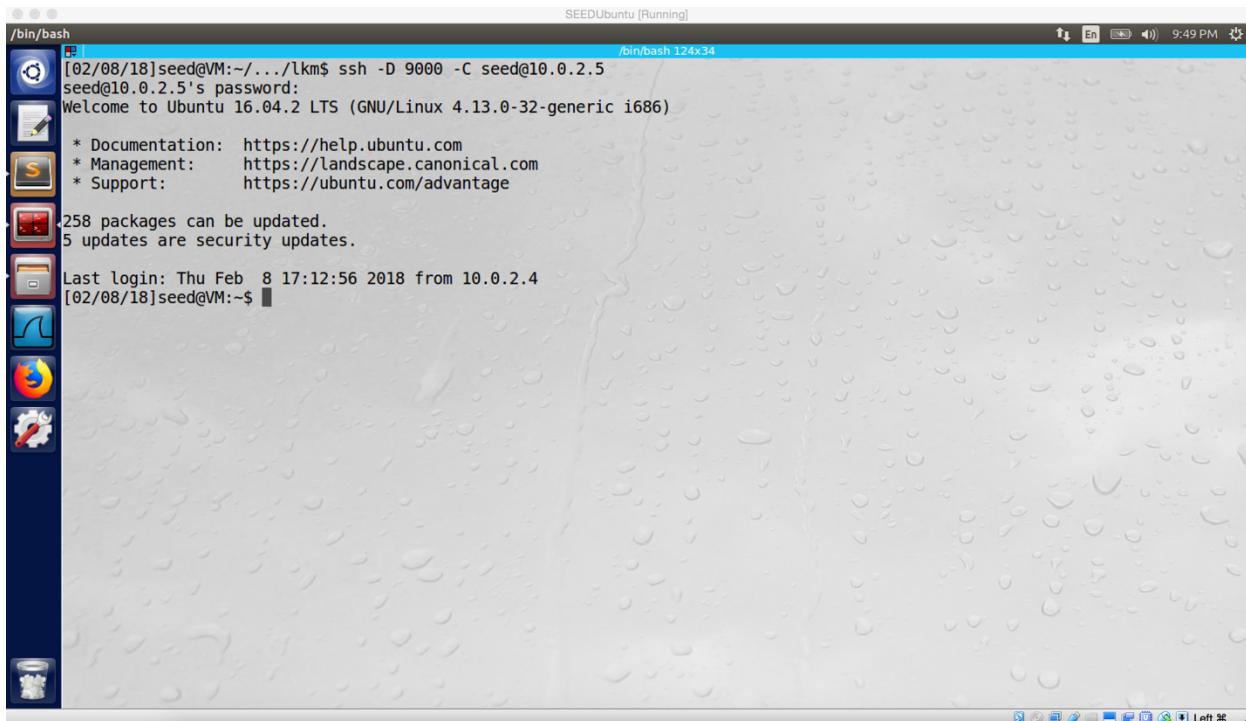
I did this task before I finished task 2, so I use ufw to finish this task. First, I start ufw firewall and add rule of “sudo ufw deny out from 10.0.2.4 to 10.0.2.6 port 23” to it, this rule will prevent packet out from VM A to VM C in port 23(screenshot 1). Because telnet uses port 23 by default, telnet is blocked.

And then I typed “ssh -L 8000:10.0.2.6:23 seed@10.0.2.5”, this command established a SSH tunnel between VM A and VM B (screenshot 2). So using this tunnel, VM A will send out packets by port 8000 to VM B; after VM B received these packets, it will forward these packets to port 23 of VM C. After VM C received packets, it can send packets back on the same path with reverse order.

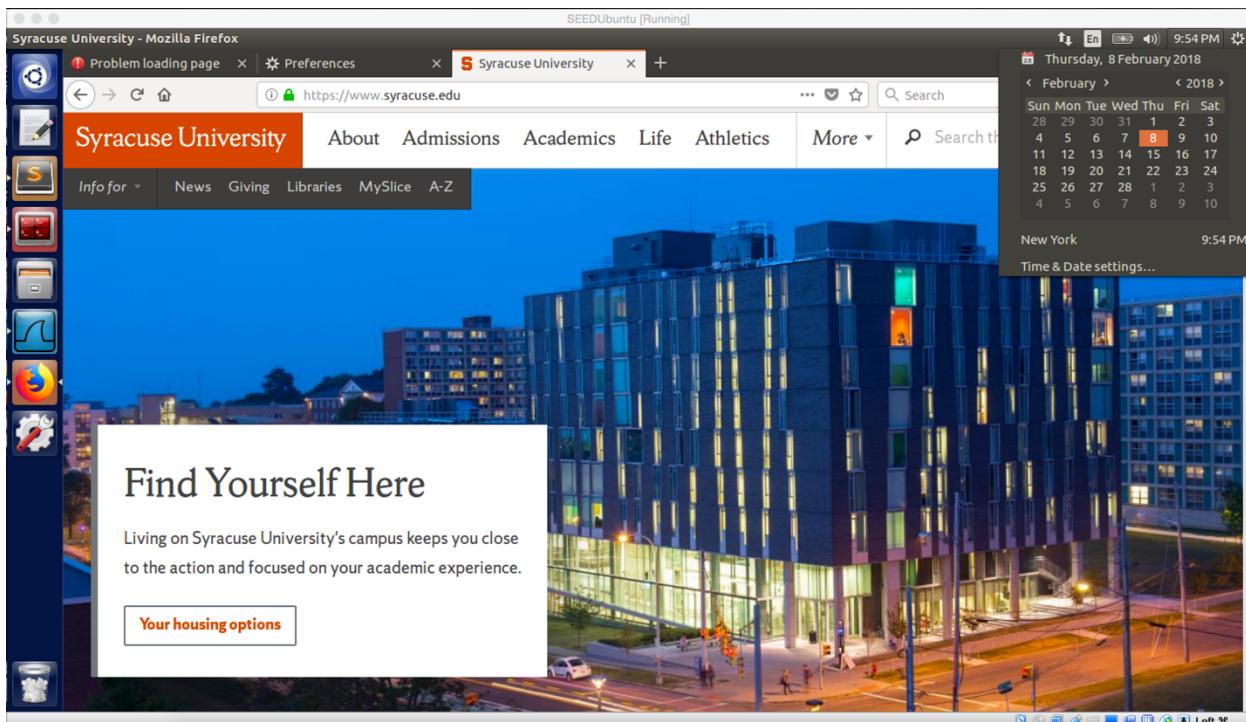
After the SSH tunnel is established, we can do telnet from VM A to VM C by command “telnet localhost 8000”, as screenshot 3 shows we successfully telnet from VM A to VM C. The screenshot 4 and 5 show the whole traffic, we can see the SSH tunnel is established between VM A and VM B (SSH protocol), and they use this tunnel to transfer packets (screenshot 4). Furthermore, we can see TCP packets are go through port 8000 (screenshot 5.2 and 5.3), and finally received by VM C in port 23; on the other hand, VM C also send packet out via port 23 (screenshot 5.1)

Task 3.b

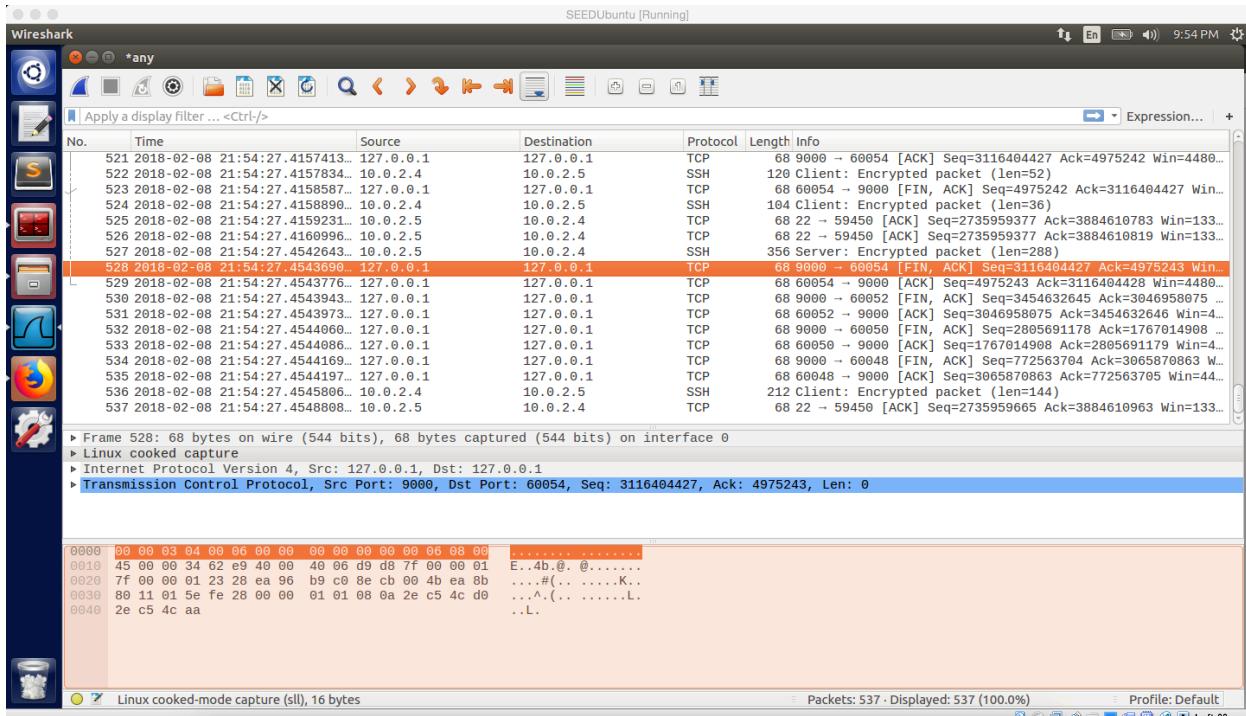
Because Facebook is hard to block, I decide to block SU main website (www.syracuse.edu) in this task.



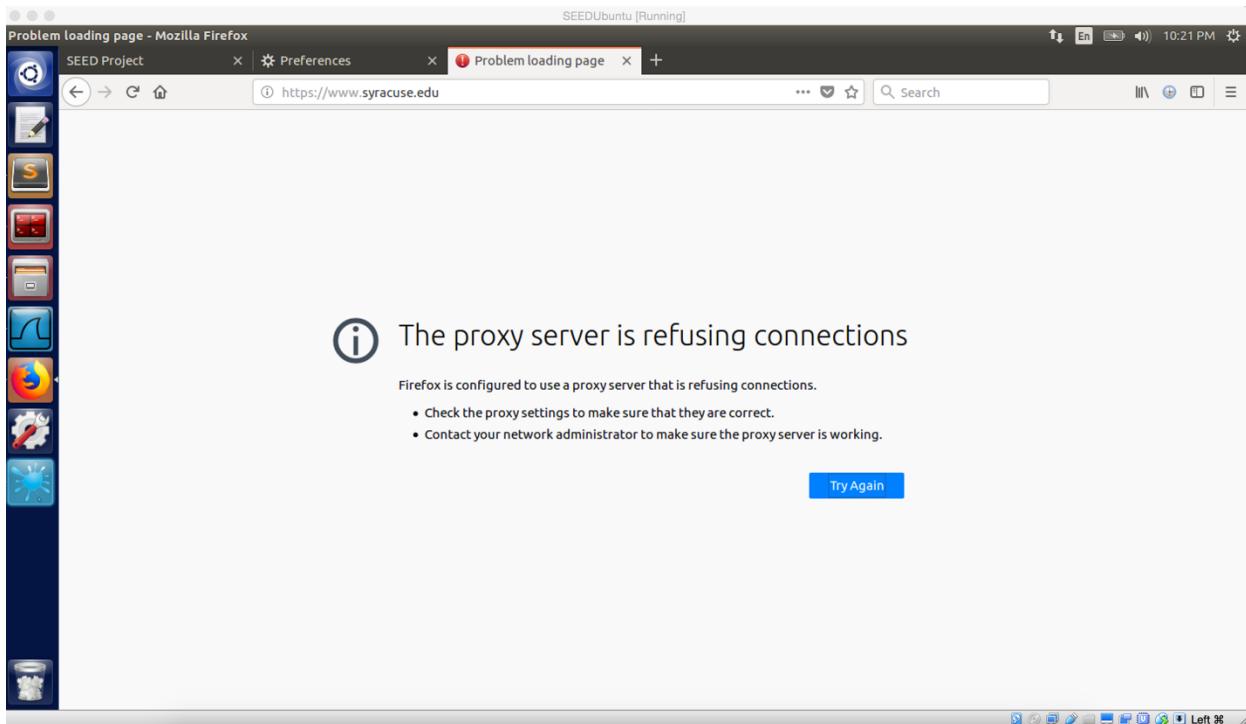
Screenshot 1 on VM A. Successfully establish SSH tunnel between VM A and VM B

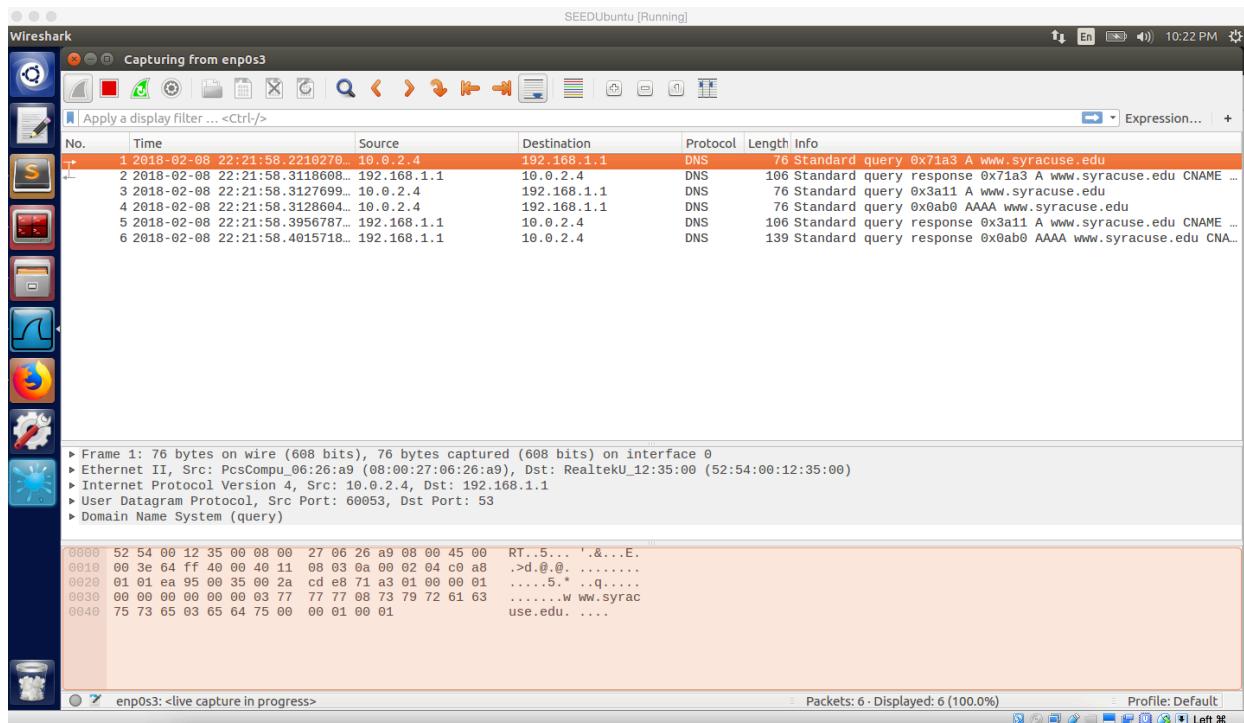


Screenshot 2 on VM A. I can visit SU website

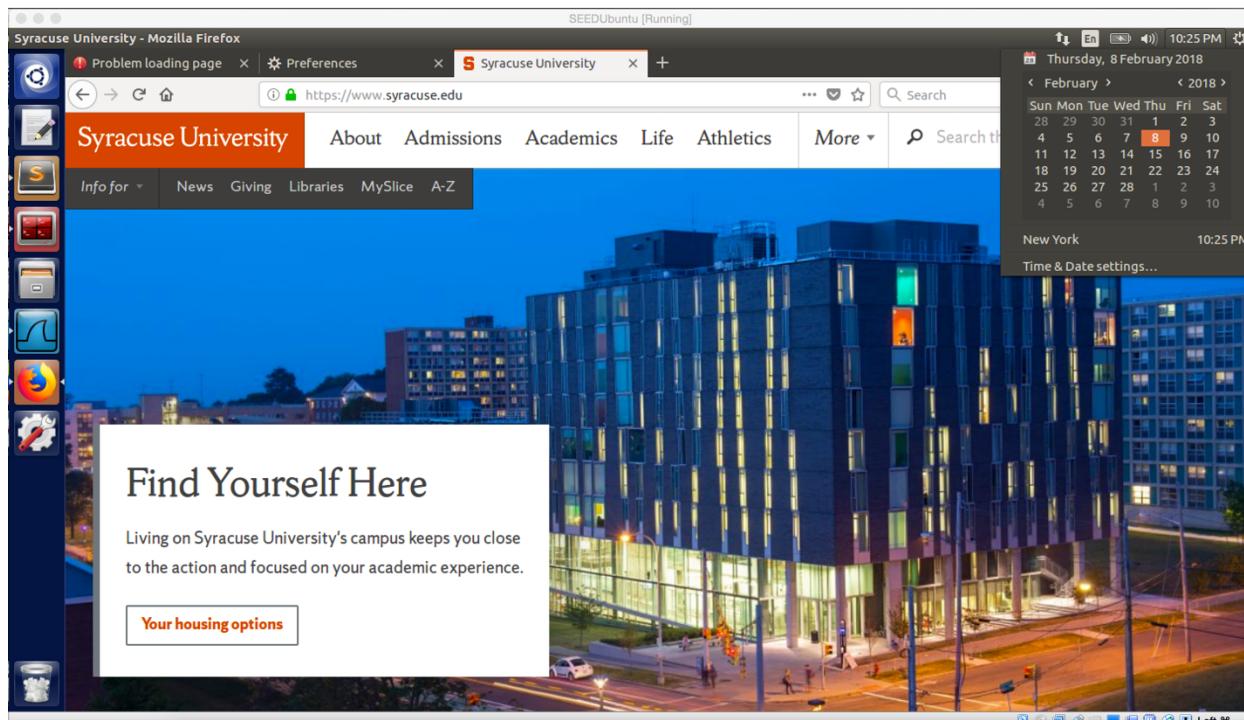


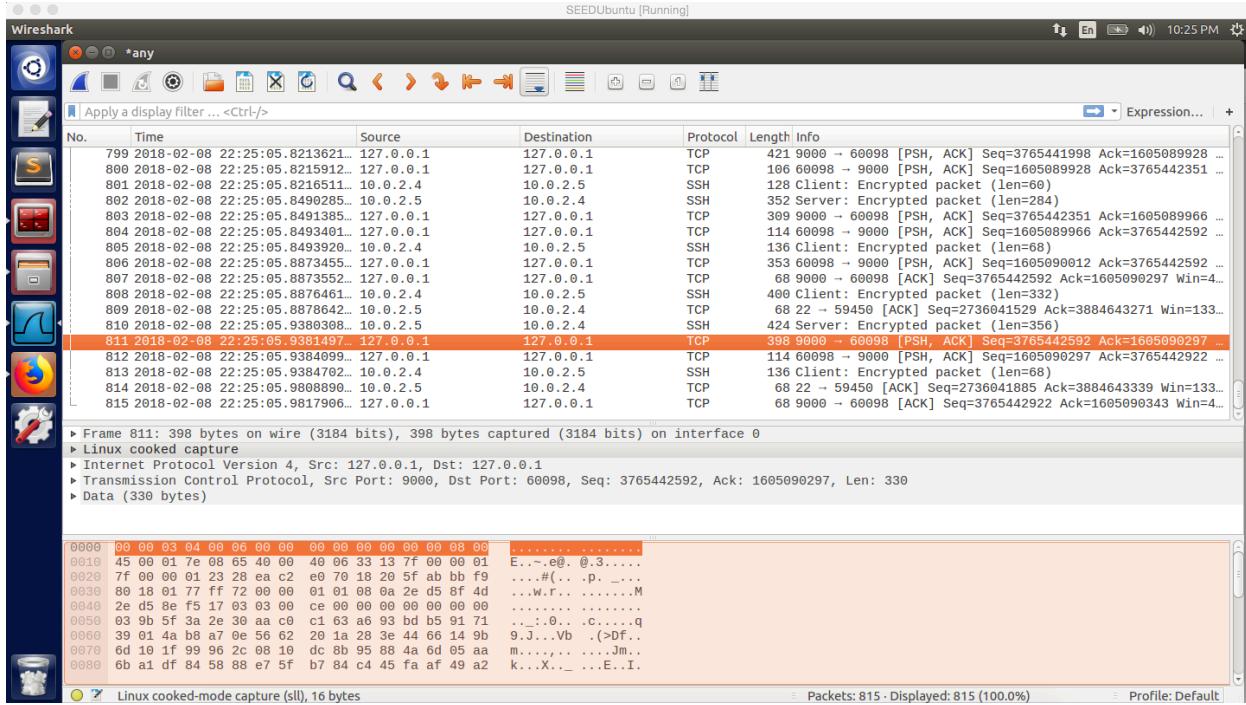
Screenshot 3 on VM A. SSH tunnel is established between VM A and VM B (SSH protocol packet), and Packets go through port 9000. Successfully connect to SU main website, even my netFilter program is still running.





Screenshot 5 on VM A. After I broke the SSH tunnel, I cannot access to SU main website, and the Firefox shows “The proxy server is refusing connections”





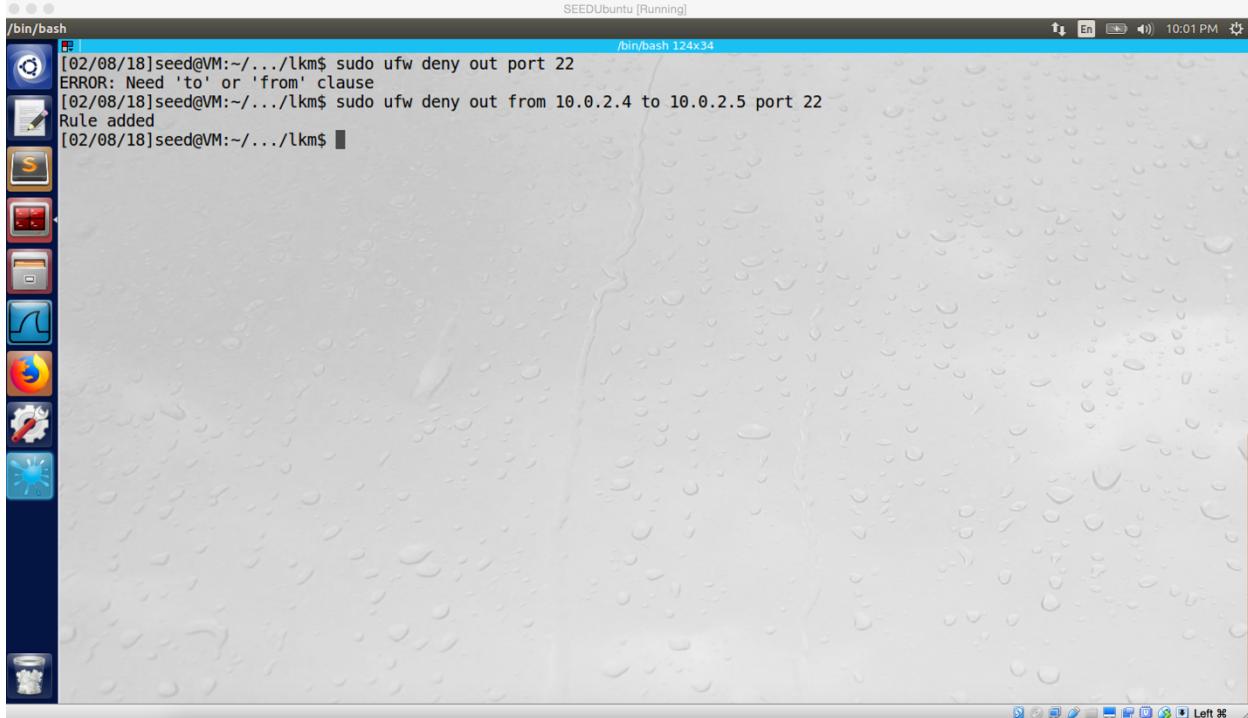
Screenshot 6 on VM A. I can access to SU main website again after I built SSH tunnel from VM A to VM B.

Observation and Explanation:

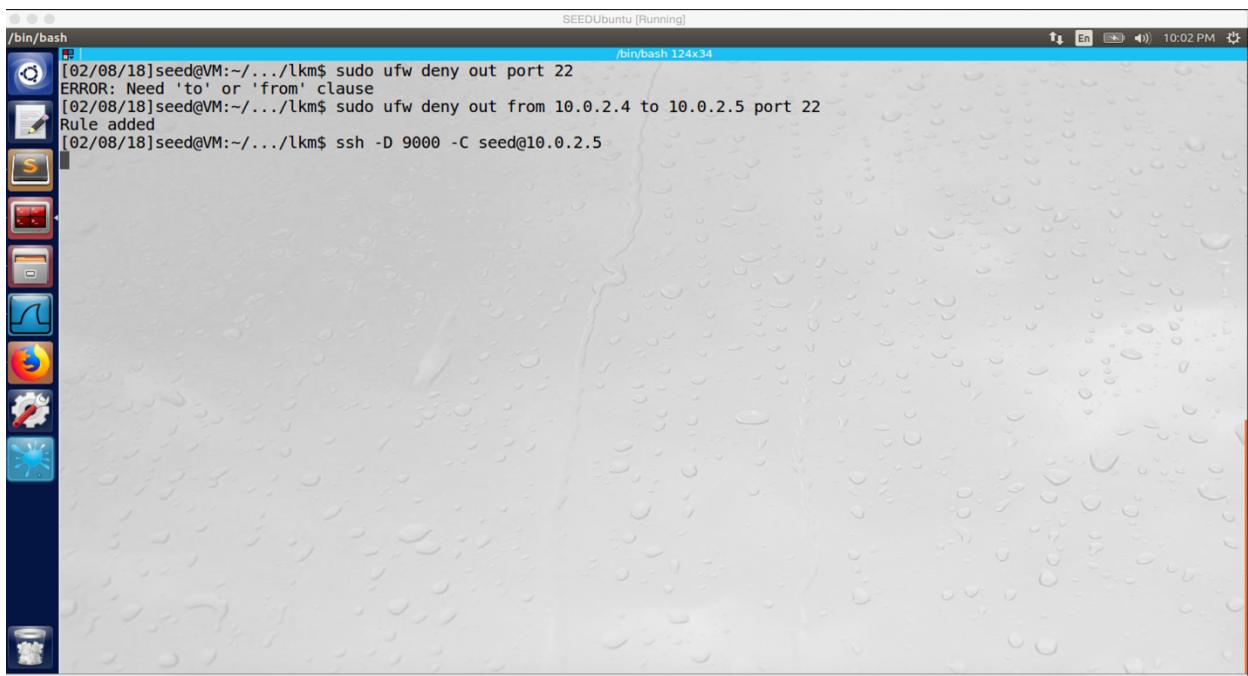
1. Since the SU website is blocked by my module; before I use SOCKS proxy and establish SSH tunnel, the SU main website is blocked, and I cannot visit it on VM A. After I established SSH tunnel between VM A and VM B by command “ssh -D 9000 -C seed@10.0.2.5” (screenshot 1). This command will establish a SSH tunnel between VM A and VM B, and the VM A will use port 9000 to receive and send packet. However, because this time we didn’t specify a destination, we need to set SOCKS proxy to make sure whatever the destination is, it will go through port 9000 (to use the SSH tunnel). After I configured SOCKS proxy on the browser, I can use VM A to connect SU website again (screenshot 2). As the screenshot 3 shows, the SSH tunnel is established between VM A and VM B, they use this tunnel to transport packets (SSH protocol); and the TCP packet is transported via port 9000.
2. After I broke the SSH tunnel, I cannot visit SU website, and the error message shows “The proxy server is refusing connections”. SOCKS is used to forward data to port 9000 and make sure these data can go through the SSH tunnel. However, the tunnel is broken now, so the SOCKS cannot work now. As screenshot 5 shows, there is no SSH tunnel now.
3. After reestablished the SSH tunnel, I can visit the website again. Packets go through the SSH tunnel (packet between 10.0.2.5 and 10.0.2.4 with SSH protocol) and port 9000 as before (screenshot 6).

Question 4

The default SSH port is 22, and it is assigned for contacting SSH servers (SSH remote login protocol). Therefore, after I add rule to ufw to block port 22, the SSH tunnel cannot be established anymore (as following screenshots show). However, if we change SSH port, then we may still establish tunnel to evade egress filtering.

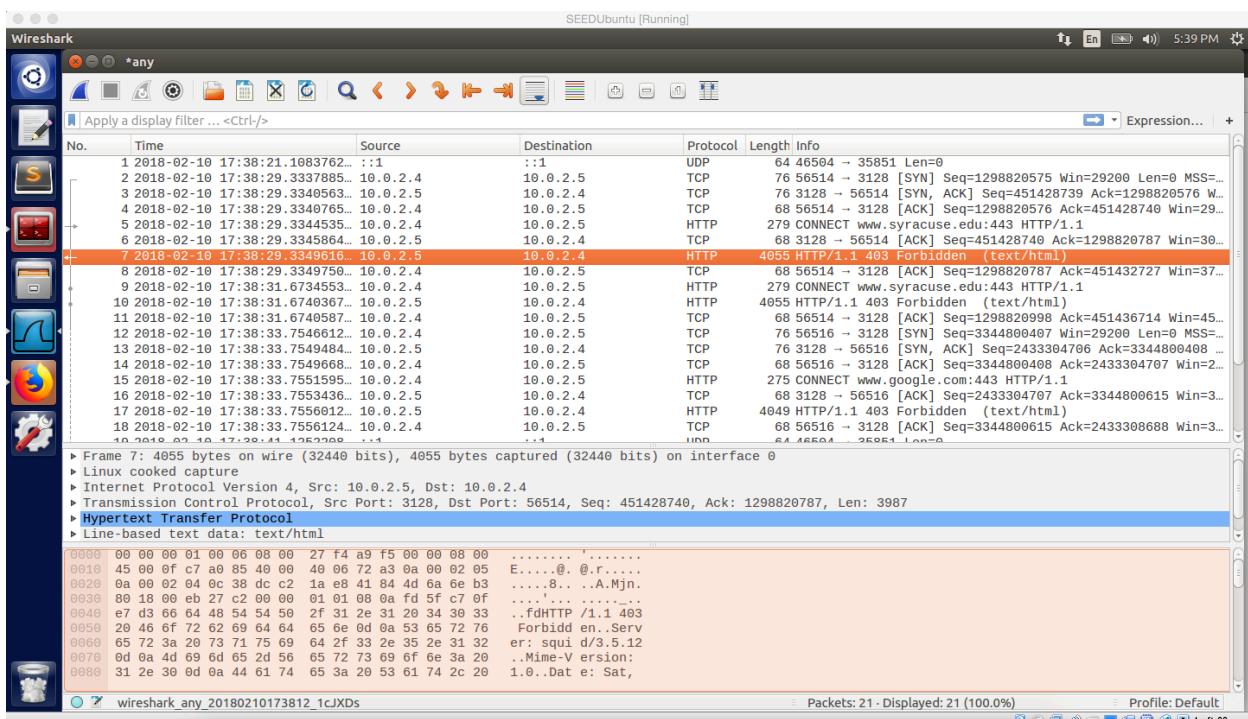
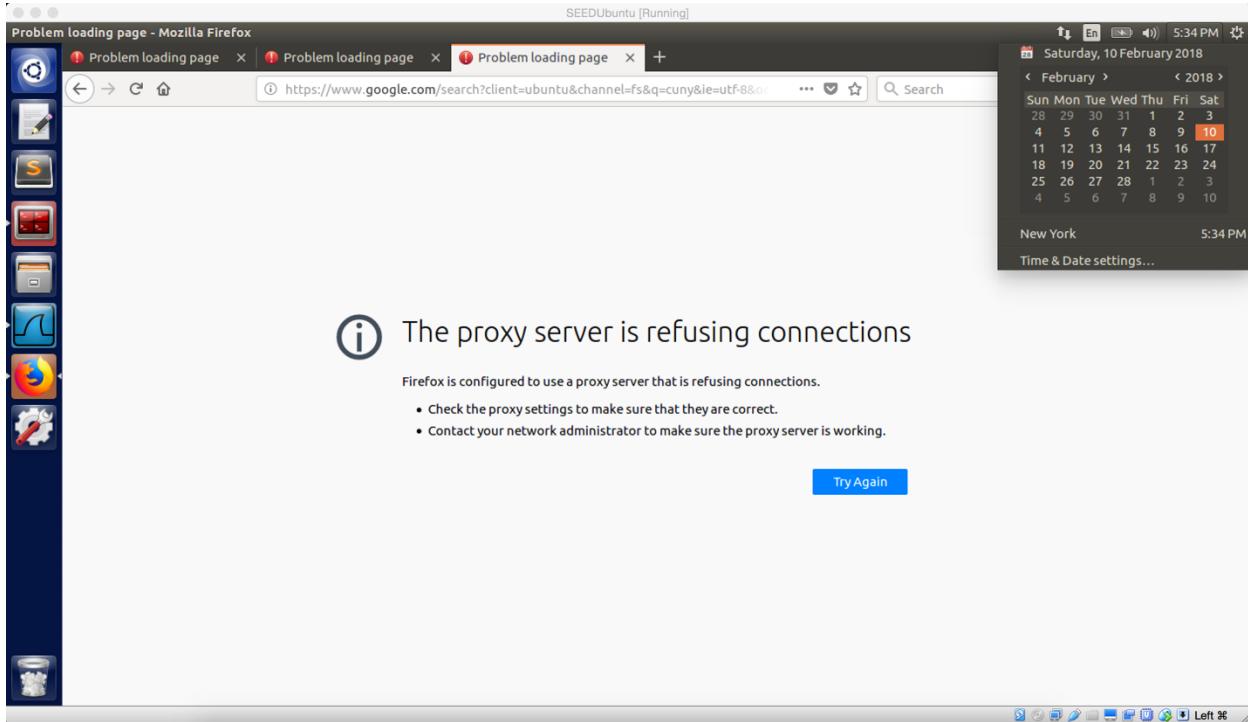


block port 22 from VM A to VM B

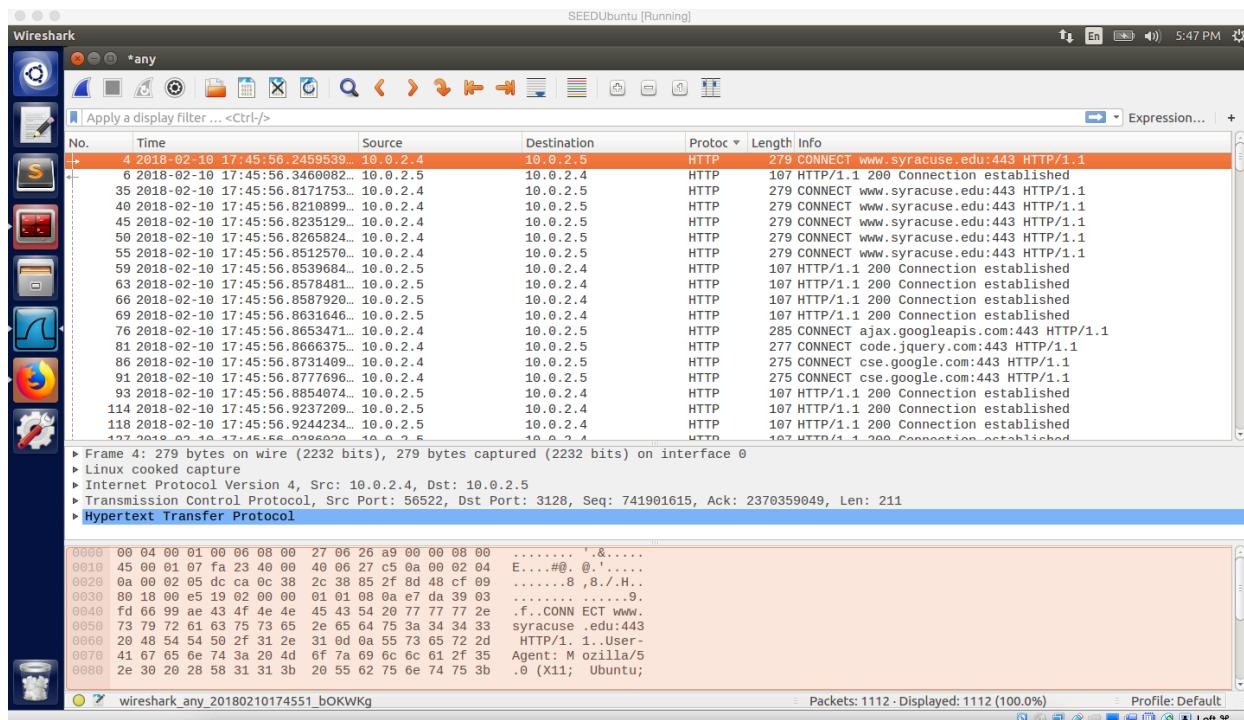


SSH tunnel cannot be established

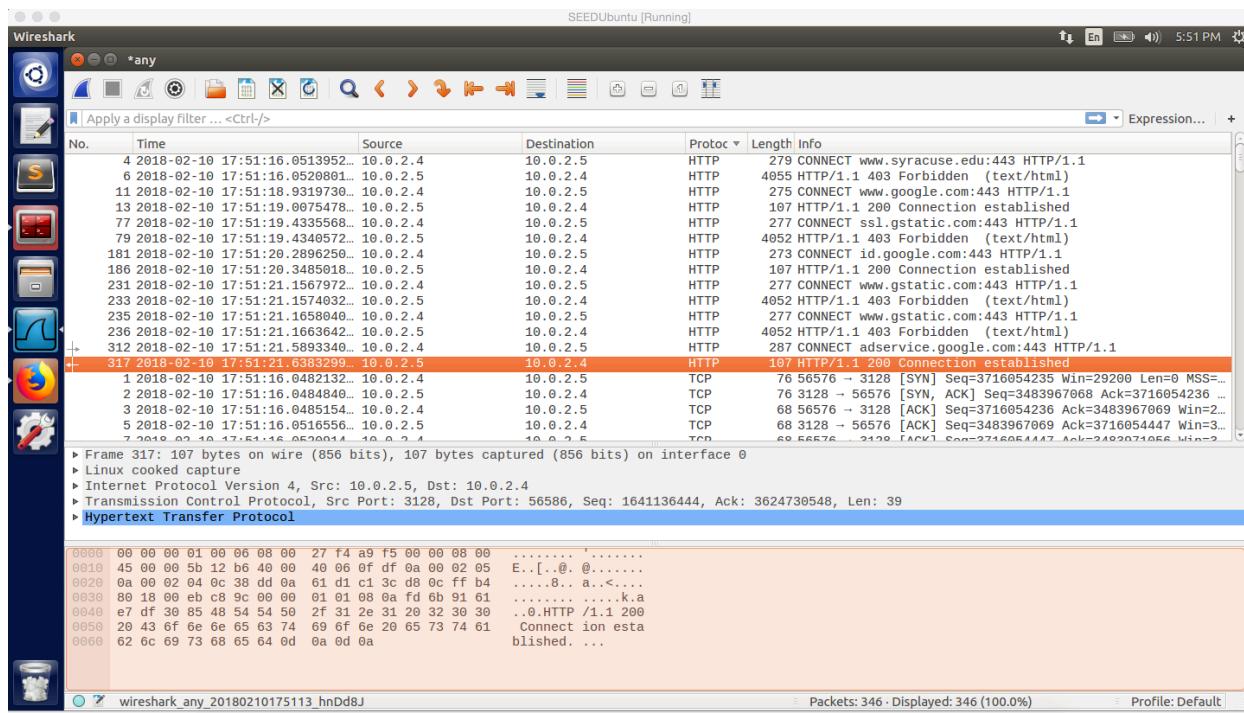
Task 4.a



Screenshot 1 on VM A. I try to access SU main website; the browser says “The proxy server is refusing connections”



Screenshot 2 on VM A. After I add rule of “http_access allow all” and start squid on VM B, I can access all website from VM A.



Screenshot 3 on VM A.

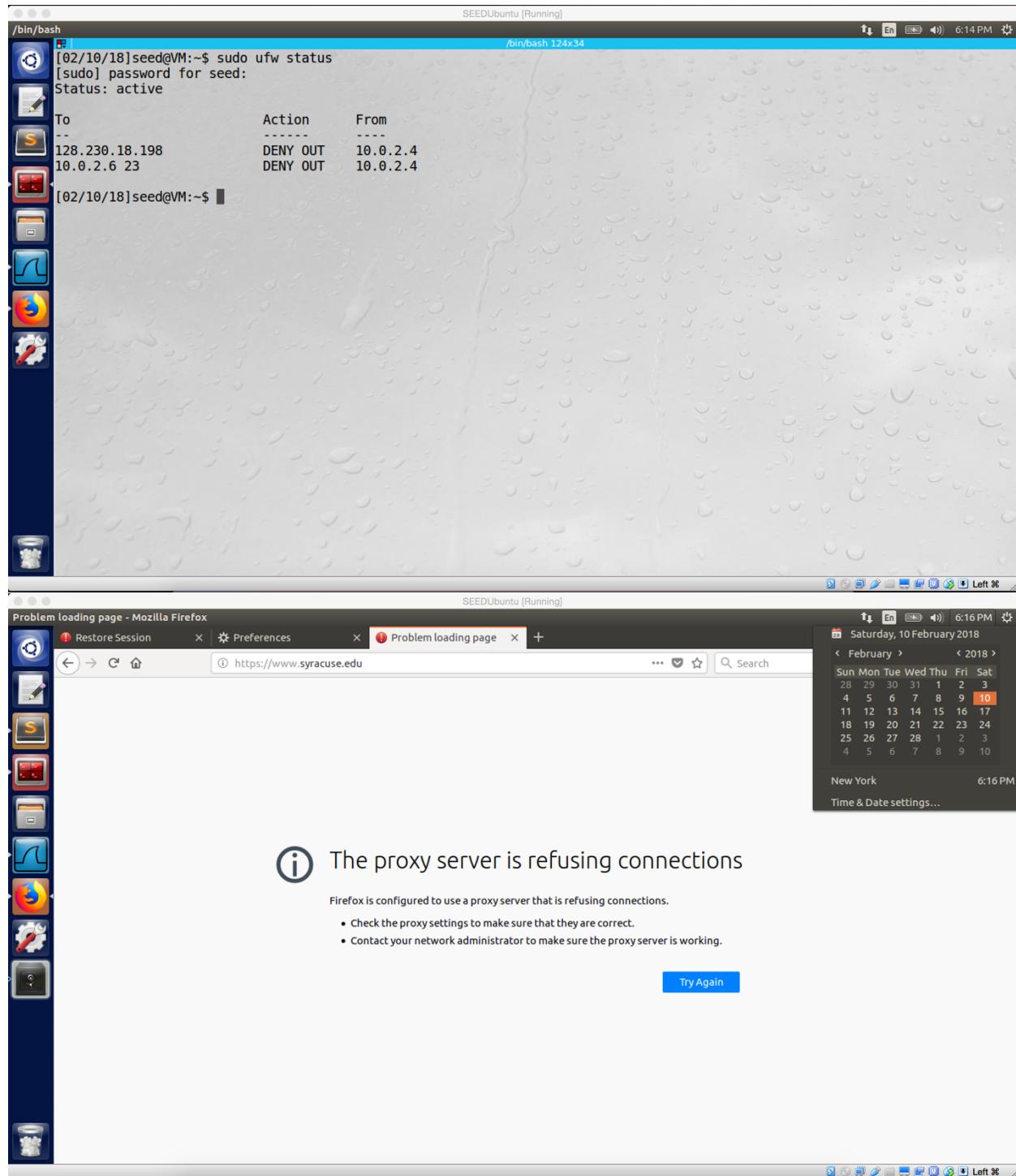
Observation:

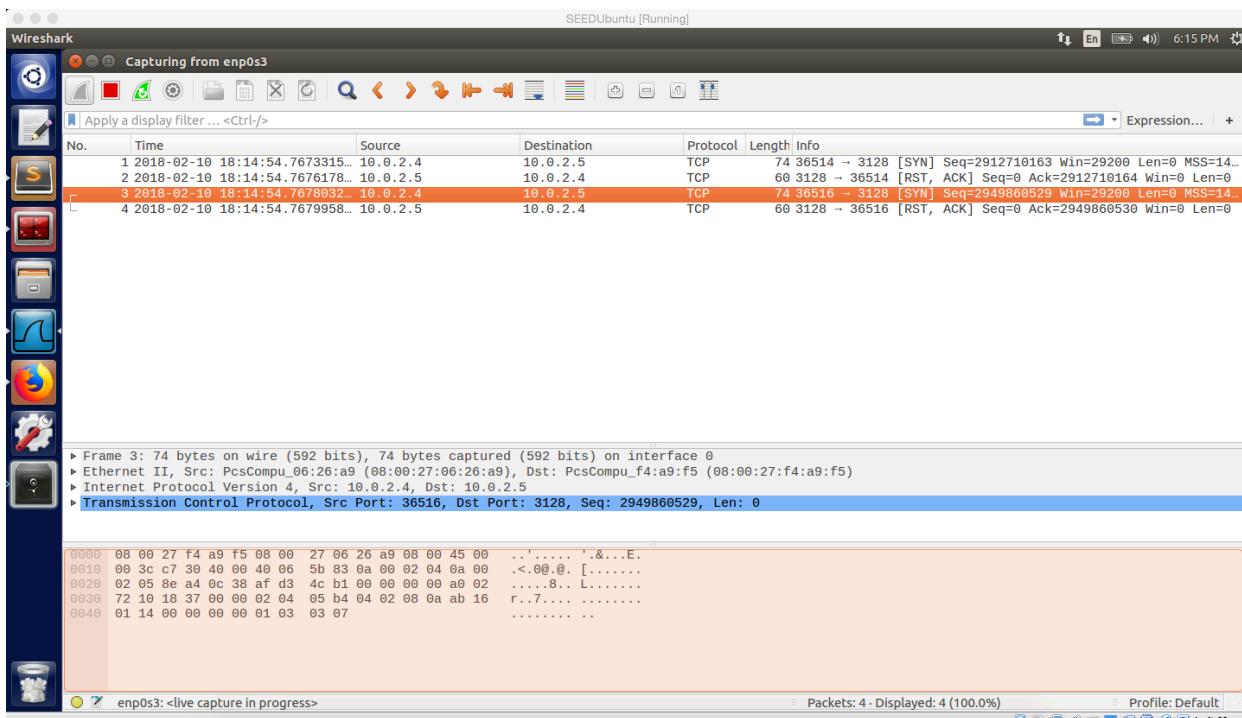
1. After I install squid on VM B and setup browser on VM A, I use the browser to connect SU website; but the connection is refused, the error message shows “The proxy server is refusing connections”. And then I check the Wireshark, it shows I connect to www.syracuse.com with HTTP protocol; but in the next two line, this packet is forbidden. Afterwards, I also try to connect to google.com, but I get same result. This means I cannot connect to this website by browser. (screenshot 1)
2. And then I go to read squid.conf. The http_access rule is used to allow or deny access based on defined access list, and it will allow or deny a message received on http, https, or FTP port. Because we have not yet add any rules in squid.conf, its default value is to deny request. Therefore, all external websites are blocked.
3. After I add “http_access allow all” to the squid.conf, I can use the browser to visit all website now. “http_access allow all” mean allow http, https and ftp port to receive message. As the Wireshark shows, when we connect to SU website, the connection is established (screenshot 2)
4. After I add “acl GOOGLE dstdomain .google.com; http_access allow GOOGLE” and delete “http_access allow all”, I can only access to google.com. all other websites are locked; the error message is “The proxy server is refusing connections”. The Wireshark shows only the connection to google.com is established, other connections are forbidden (screenshot 3).

Explanation:

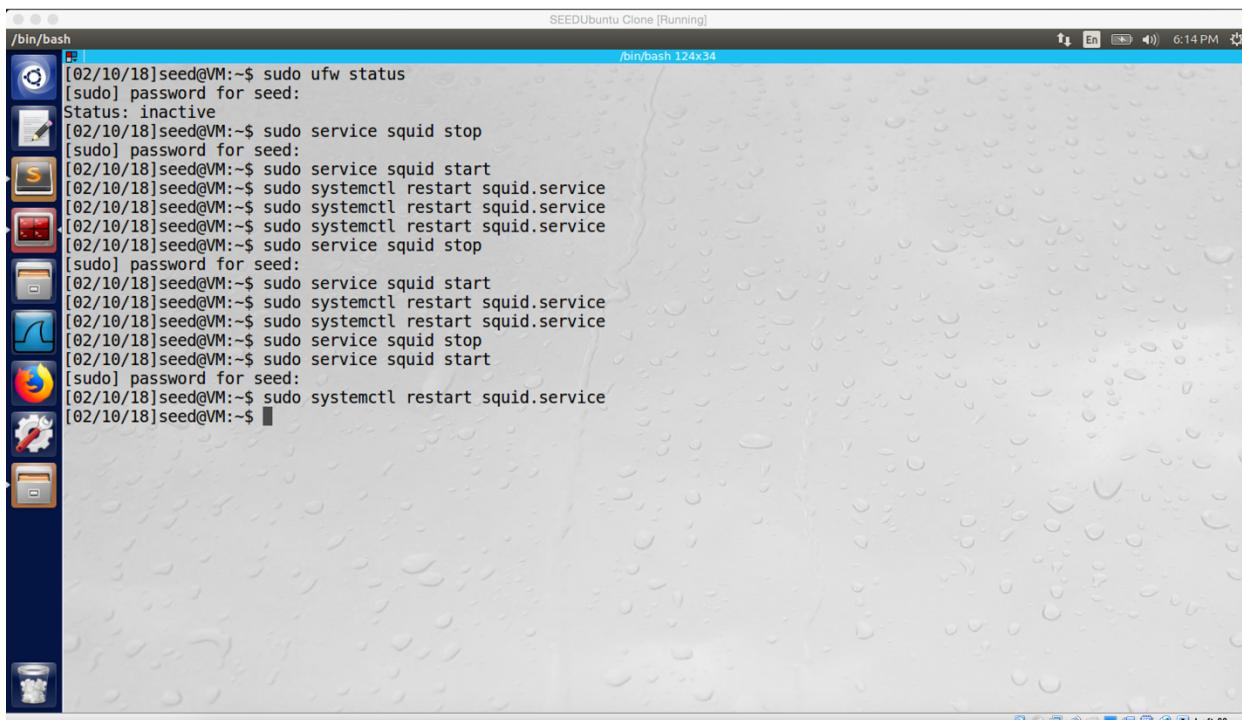
Web Proxy is an application firewall; it is used to control browser access. When we configure the browser to use http proxy with a particular IP address and port 3128, it actually builds a bridge between host machine and the machine of the particular IP address (in our case, they are VM A and VM B); and the Web Proxy will enforce all packets (transport between our browser and destination) go through the bridge. Meanwhile, Web Proxy can also add rule to the bridge, so it can control access of the bridge. In step 1, because squid deny any request by default, so we cannot use the browser to visit any website, and we can see all HTTP packets are forbidden (screenshot 1). In step 3, after I add rule to squid.config which allows all request, so I can use browser to visit any website; meanwhile we can see HTTP packets shows connection established (between 10.0.2.4 and 10.0.2.5) with port 3128 (screenshot 2). In step 4, after I change the rule which restrict access of any website, except google.com; the only website I can visit is google.com, and other website are blocked. As the screenshot 3 shows, only connection to google.com is established, and other connections are forbidden.

Task 4.b

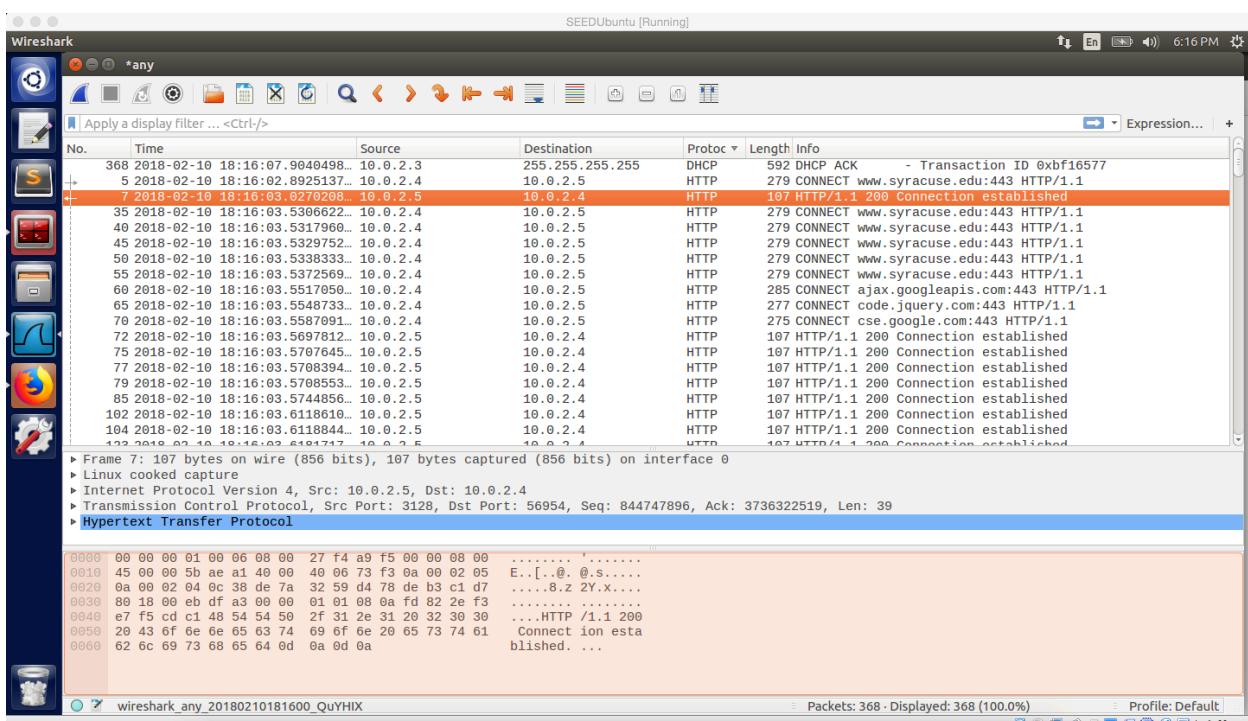
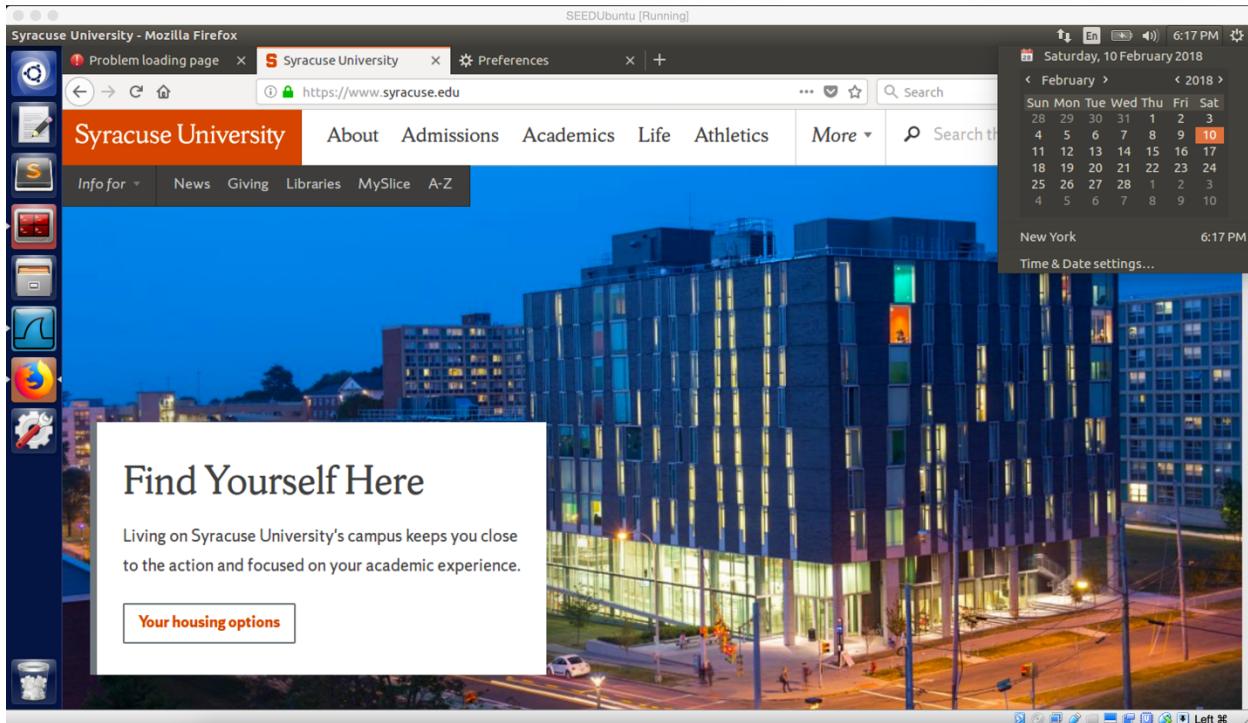




Screenshot 1 on VM A. Enable ufw on VM A and add rule to block SU main website. Cannot connect to SU website.



Screenshot 2 on VM B. Update http_access and restart squid on VM B



Screenshot 3 on VM A. Successfully bypassing the firewall and connect to SU main website

Observation:

First, I enable ufw firewall, and ufw already has rule to block SU website (screenshot 1). Afterwards, on VM B, I updated the squid.conf file, I set to allow all request. Finally, I use the

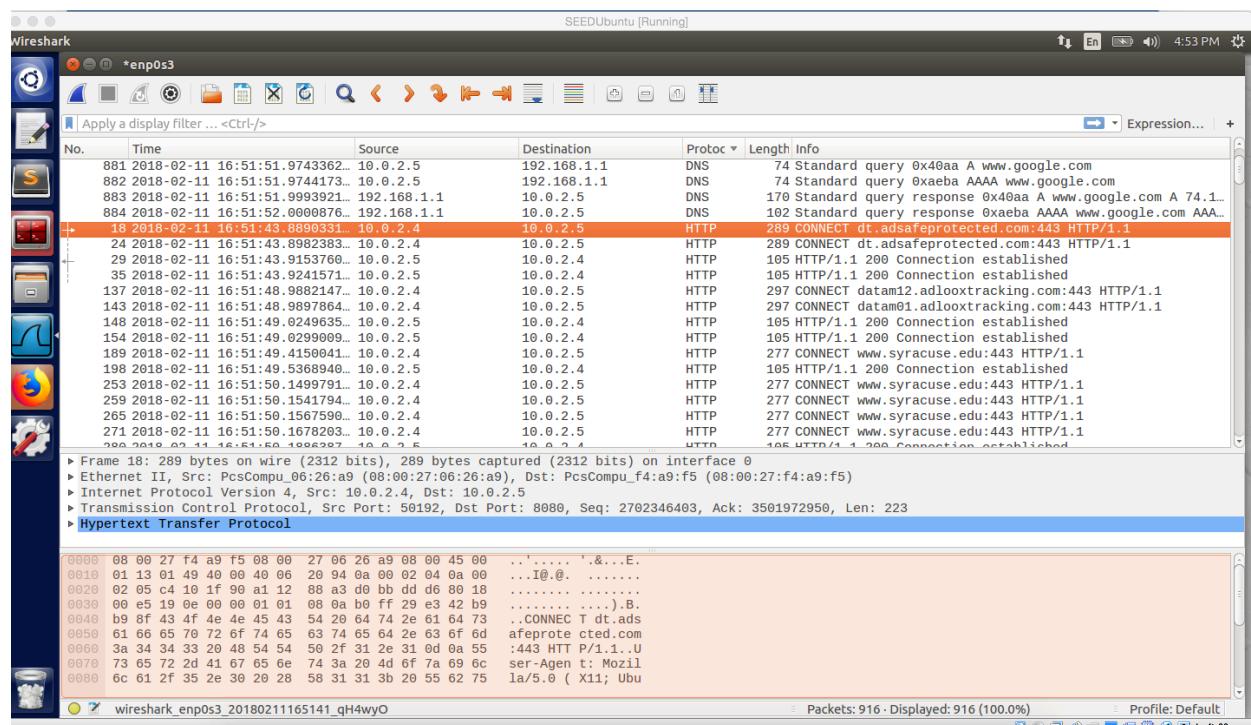
browser to visit SU website successfully (screenshot 3) shows (I didn't change the browser setting, so HTTP proxy still open).

Explanation:

In this task, we use Web Proxy to build a bridge between VM A and VM B. Because ufw does not block communication between VM A and VM B, we can send our packets to VM B by the bridge with port 3128; and then the VM B will forward these packets to SU website. Afterwards, SU website will send packet back in the same path, but reverse order. In the screenshot 3, we can see the connection to SU website is established.

Question 5:

Squid use port 3128 as web proxy server port in default. Therefore, after I added rule to ufw to block this port, I cannot use web proxy to evade the firewall. However, if we change port 3128 to the other ports (such as 8080, 6588) in the squid.conf file, then we can use web proxy again.



Web proxy works on port 8080

Task 4.c

Because squid is HTTP proxy, it does not work on HTTPS. As TA Kailiang's suggestion, I use www.bbc.com to do this task.

1. For the code, the only thing I changed which is using bbc.com instead of cis.syr.edu

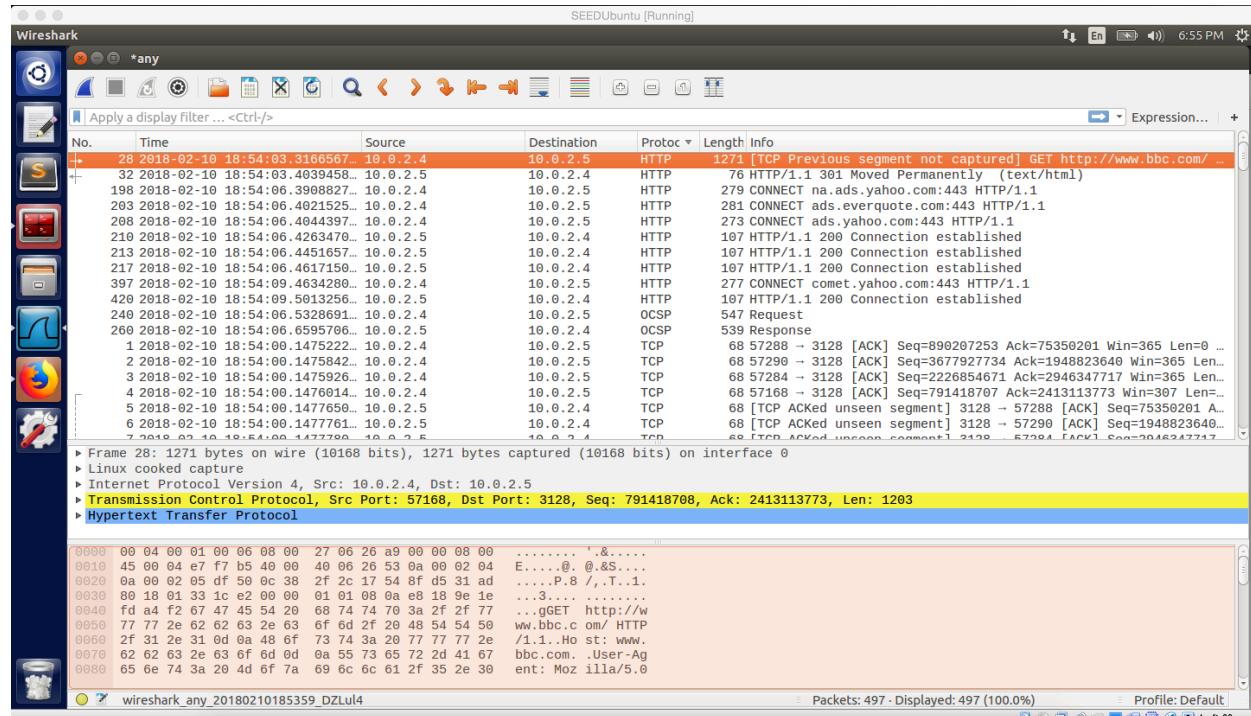
```
#!/usr/bin/perl -w
use strict;
use warnings;
```

```

# Forces a flush after every write or print on the STDOUT
select STDOUT; $|=1;
# Get the input line by line from the standard input.
# Each line contains an URL and some other information.
while (<>)
{
    my @parts = split;
    my $url = $parts[0];
    #check url is the url we want
    #if yes, redirect the user to the "STOP" sign page
    if ($url =~ /www\.bbc\.com/) {
        # URL Rewriting
        print "http://www.yahoo.com\n";
    }
    else {
        # No Rewriting.
        print "\n";
    }
}

```

The above code is used to written URL. In the if statement, when the target URL is www.bbc.com, it will be rewritten to www.yahoo.com. As a result, every time when we visit www.bbc.com, we will be redirected to www.yahoo.com. If we change [bbc.com](http://www.bbc.com) in the if statement to other website, then the new URL will be written by [yahoo.com](http://www.yahoo.com) as well.

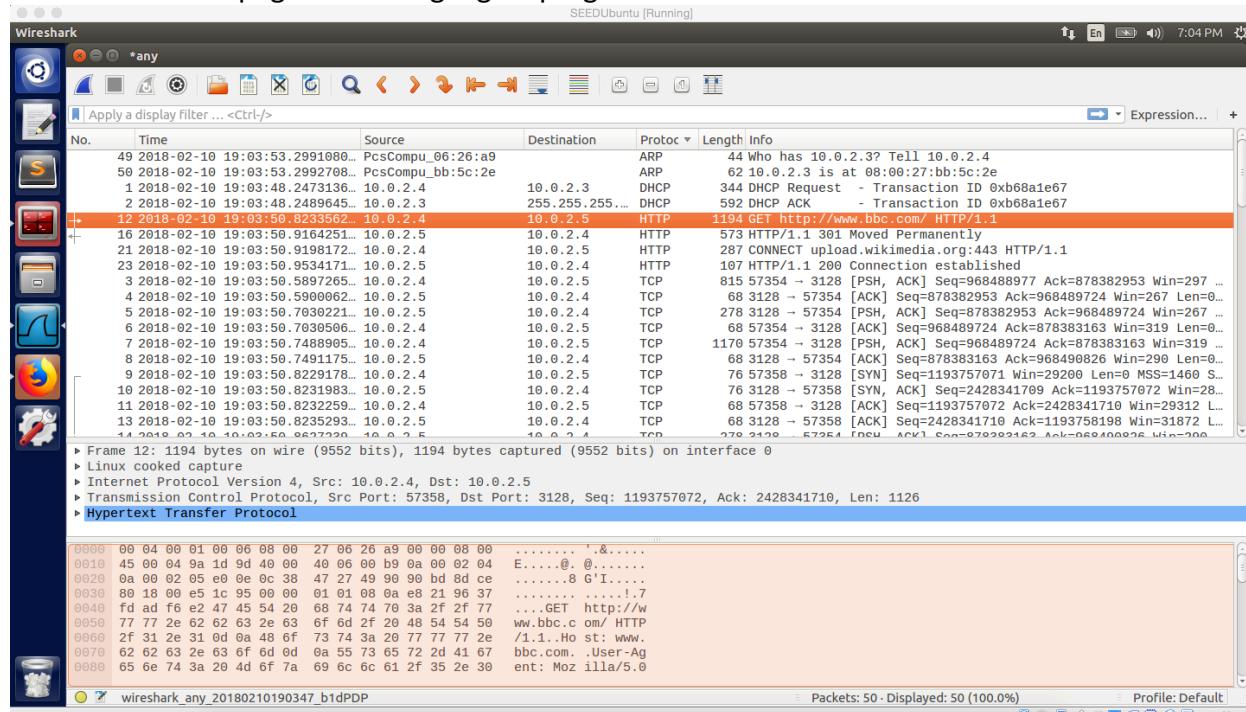


screenshot 1. According to the screenshot of Wireshark, the first HTTP packet contained information of [bbc.com](http://www.bbc.com), and then it is written to be [yahoo.com](http://www.yahoo.com). Finally, the connection to [yahoo.com](http://www.yahoo.com) is established.

2. For the code, I leave all things unchanged, excepting the code inside the if statement.

And I changed it to

```
print "http://upload.wikimedia.org/wikipedia/commons/b/bd/France_road_sign_AB4.svg\n";  
which is a page containing big stop sign.
```



screenshot 2. We can see bbc.com is rewritten again by the new URL, and the new URL redirect us to the following page:

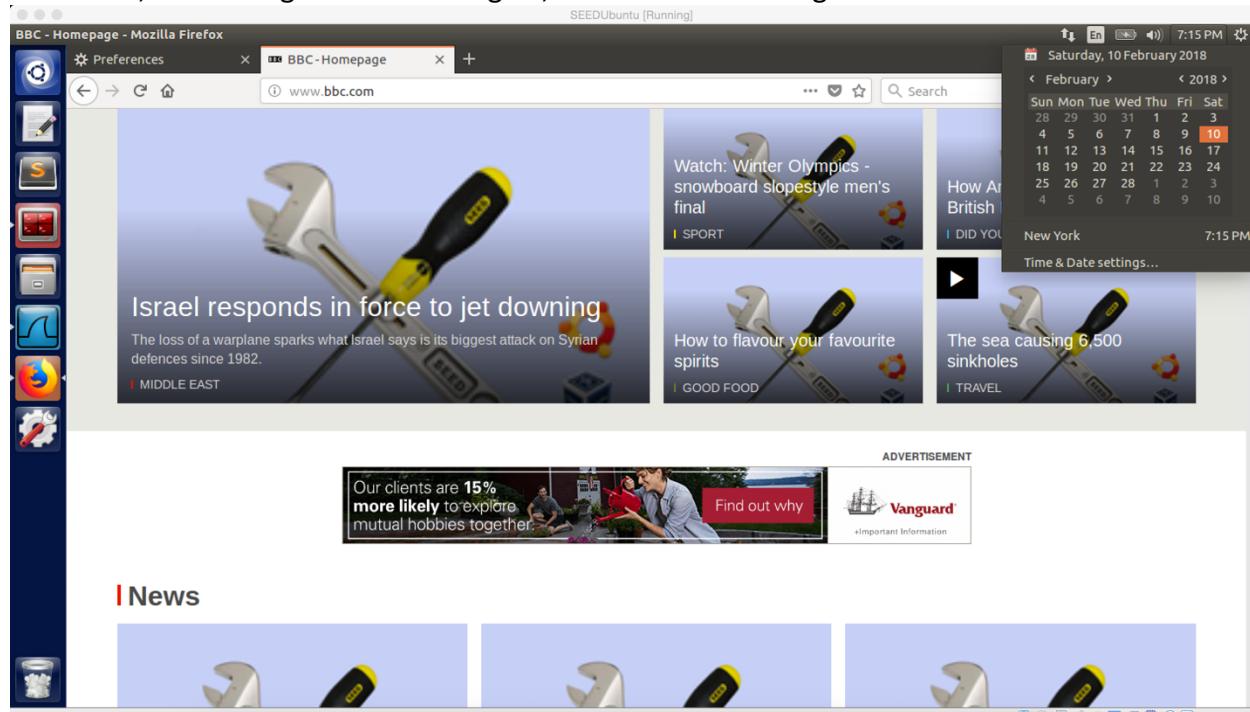


3. For this task, I changed the whole if statement, and my code is following:

```
#!/usr/bin/perl -w
use strict;
use warnings;
# Forces a flush after every write or print on the STDOUT
select STDOUT; $|=1;
# Get the input line by line from the standard input.
# Each line contains an URL and some other information.
while (<>)
{
    my @parts = split;
    my $url = $parts[0];

    #if the url with extension of .gif, .jpg, or .png, this means it's an image
    #so we just swap this url with my url, and then the image will also be changed to my image
    if (($url =~ /(\.*\gif\)/i) || ($url =~ /(\.*\jpg\)/i) || ($url =~ /(\.*\png\)/i))
    {
        print "http://www.cis.syr.edu/~wedu/seed/img/setup_img.png\n";
    }
    else {
        # No Rewriting.
        print "\n";
    }
}
```

As above code shows, whenever URL with extension of .gif, .jpg, or .png, such link will be rewritten to http://www.cis.syr.edu/~wedu/seed/img/setup_img.png which is a link of my image. Therefore, when we go to [bbc.com](http://www.bbc.com) again, it becomes following:



screenshot 3. All images on www.bbc.com are changed to be my image.

Observation and Explanation:

1. I first try to understand and use code from myprog.pl. The most important part of the code is if statement part. The if part will compare the current URL with the URL we set, if they are same, then the current URL will be rewritten. In our case for step one, when we visit bbc.com, we will be redirect to yahoo.com (screenshot1).
2. In this step, I changed statement inside if. So every time when we visit bbc.com, we will be redirected to a page will stop sign (screenshot 2).
3. In this step, I changed the whole if statement part. For the if statement, the program, will catch any URL link with extension of .jpg, .png, or gif. These are image file extension. Therefore, the program will catch any image on the website. And then it will rewrite the image to my image. As a result, when we visit bbc.com again, all images are replaced by my image (screenshot 3).

Question 6:

Yes. We need to build a tunnel between host machine and another machine, then we can add date into an ICMP packet and send it (by the tunnel) to the other machine by ICMP request. After that machine received packet, it can forward the ICMP packet to a server. When the server received the packet, it can send reply back in the same path with reverse order. This seems not very hard.