

**Ryerson University**  
**CPS-633 Lab 2 Report**  
**Packet Sniffing and Spoofing Lab**  
**Group 2**

Nichalus: 500744672

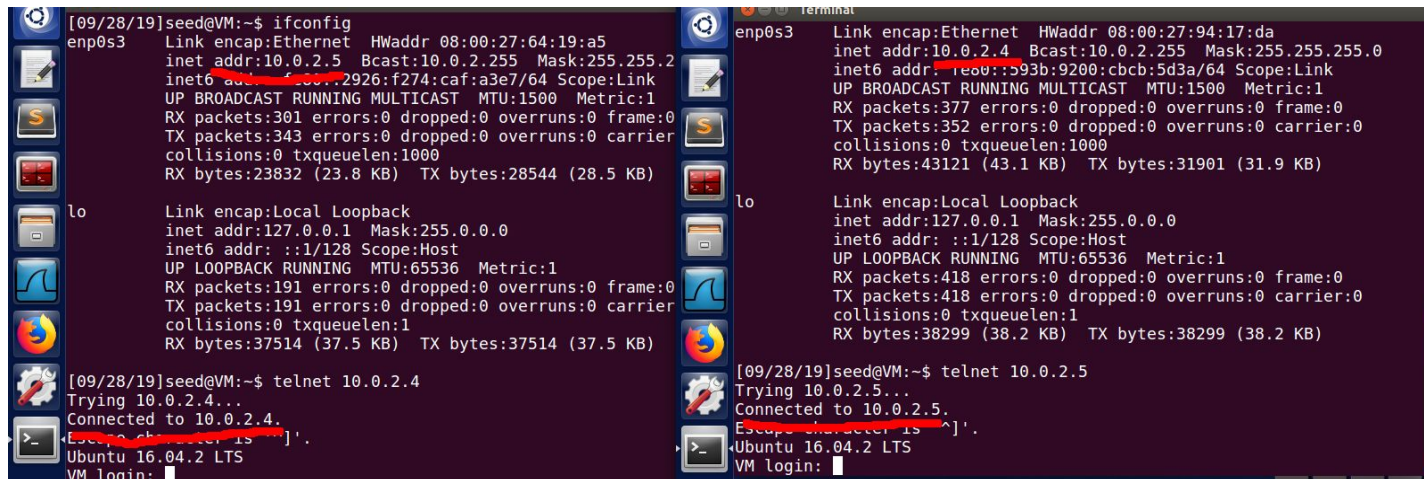
Matt: 500805168

Thomas: 500865018

Christopher: 500840595

## Task 1: Using a firewall

For this task we are using iptables through ufw to block communication between VMs. Here we can see the two VMs without ufw set up establishing telnet connections:



The image contains two side-by-side terminal screenshots from Ubuntu 16.04.2 LTS VMs. The left terminal shows the output of the 'ifconfig' command for the 'seed@VM' user, displaying details for the 'enp0s3' and 'lo' interfaces. The right terminal shows the output of 'ifconfig' for the 'enp0s3' and 'lo' interfaces. Below the 'ifconfig' output in both terminals, the user runs 'telnet 10.0.2.4' and 'telnet 10.0.2.5' respectively, showing successful connections to the other VM.

```
[09/28/19]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:64:19:a5
          inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255
          inet6 addr: fe80::2926:f274:caf:a3e7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:301 errors:0 dropped:0 overruns:0 frame:0
          TX packets:343 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:23832 (23.8 KB)  TX bytes:28544 (28.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:191 errors:0 dropped:0 overruns:0 frame:0
          TX packets:191 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:37514 (37.5 KB)  TX bytes:37514 (37.5 KB)

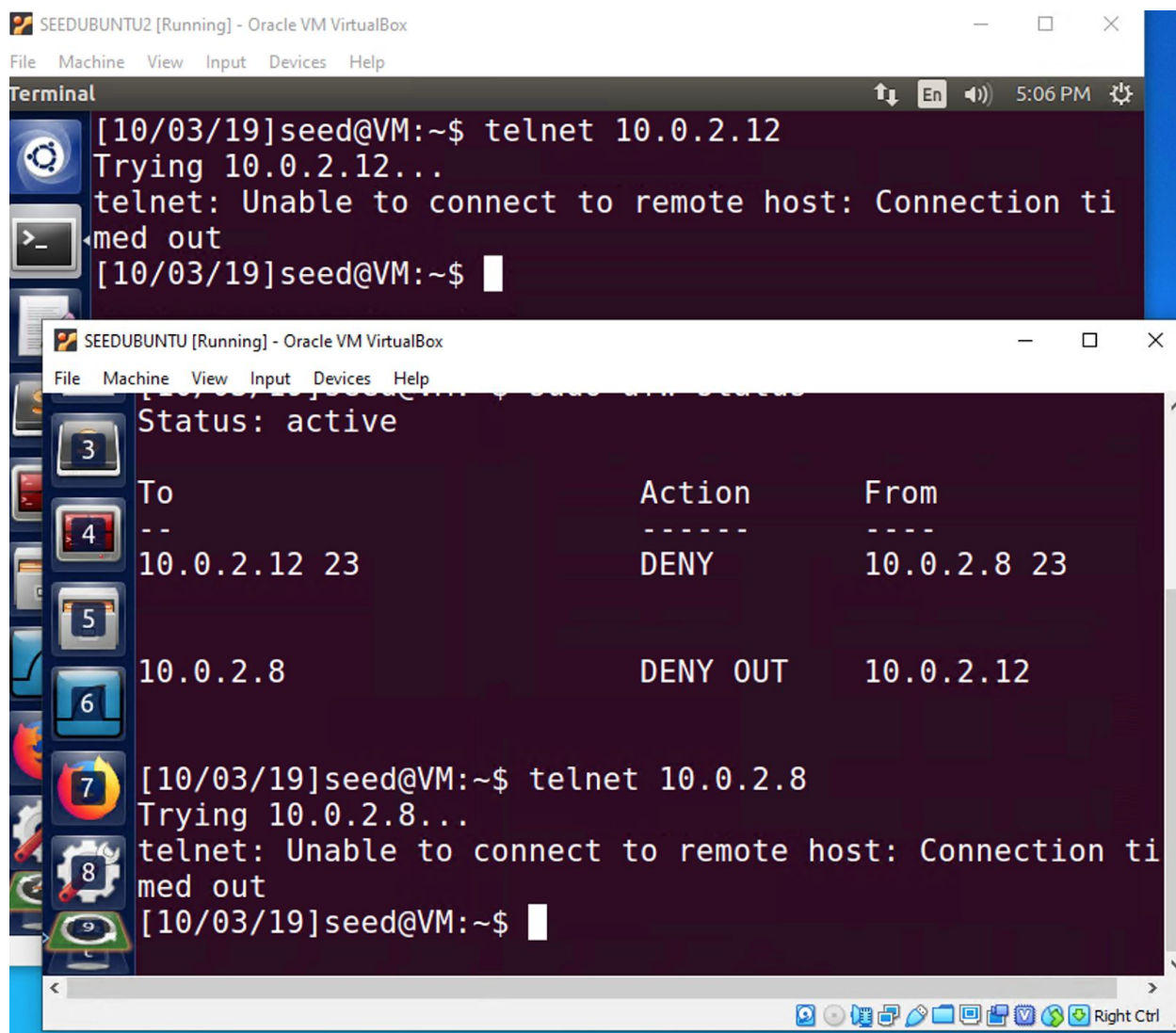
[09/28/19]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: 
```

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:94:17:da
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255
          inet6 addr: fe80::593b:9200:cbcb:5d3a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:377 errors:0 dropped:0 overruns:0 frame:0
          TX packets:352 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:43121 (43.1 KB)  TX bytes:31901 (31.9 KB)

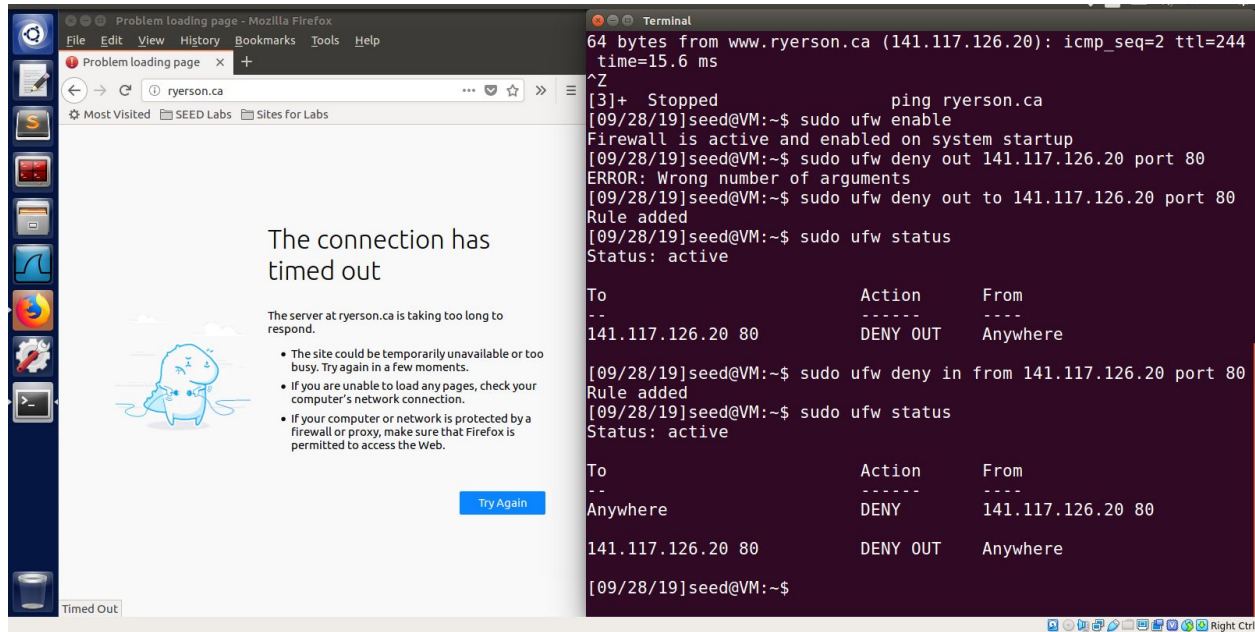
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:418 errors:0 dropped:0 overruns:0 frame:0
          TX packets:418 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:38299 (38.2 KB)  TX bytes:38299 (38.2 KB)

[09/28/19]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: 
```

Telnet connects to port 23. So adding the ufw command: “ufw deny from 10.0.2.4 to 10.0.2.5 port 23” and “sudo ufw deny out from 10.0.2.5 to 10.0.2.4” will block that connection. Note, we did this twice, so in the second screenshot, VM A has the IP 10.0.2.12 and VM B has the IP 10.0.2.8. Indeed we can see that here:



Next using ufw we must prevent A from visiting a website. I pinged ryerson.ca to get the ip address(141.117.126.20). Using “ufw deny out to 141.117.126.20 port 80” and “ufw deny in from 141.117.126.20” we can deny any and all access to ryerson.ca.



## Task 2: Implementing a firewall

In this task, you need to use LKM and Netfilter to implement the packet filtering module. This module will fetch the firewall policies from a data structure, and use the policies to decide whether packets should be blocked or not. To make your life easier, so you can focus on the filtering part, the core of firewalls, we allow you to hardcode your firewall policies in the program. You should support at least five different rules, including the ones specified in the previous task. Guidelines on how to use Netfilter can be found in Section 3.

Using the netfilter code in section 3 as a base we developed this code which implements all the rules from the previous task, as well as blocks example.com, ryerson.ca and facebook.com:

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

/*Here I register my functions*/

static struct nf_hook_ops nfho;
struct sk_buff *sock_buff;
struct udphdr *udp_header;
struct iphdr *ip_header;
struct tcphdr *tcp_header;
```

```

static unsigned char *A_ip_address = "\xA\x0\x2\x4";
static unsigned char *B_ip_address = "\xA\x0\x2\x5";
static unsigned char *example_ip_address = "\x5D\xB8\xD8\x22"; //93.184.216.34 for
example.com
static unsigned char *ryerson_ip_address = "\x8D\x75\x7E\x14"; //141.117.126.20 for
ryerson.ca
static unsigned char *facebook_ip_address = "\x1F\xD\x50\x24"; //31.13.80.36

unsigned int hook_func(void *priv, struct sk_buff *skb, const struct nf_hook_state *state)
{
    /* Here is where we inspect the packet given in the structure pointed by skb
       Here is then where we apply our filter, and decide whether or not to accept
       or drop a packet */

    sock_buff = skb;
    ip_header = ip_hdr(sock_buff);
    struct iphdr *ip_header = (struct iphdr *)skb_network_header(skb); //you can access to IP
    source and dest - ip_header->saddr, ip_header->daddr

    //Here we put like 5 different rules, including the ones in the task above.

    //1. stop 10.0.2.4 from receiving from 10.0.2.5
    if(ip_header->saddr == *(unsigned int*)B_ip_address)
    {
        printk("How dare VM B try to access VM A? Away with you, data packet.");
        return NF_DROP;
    }
    //2. stop 10.0.2.5 from receiving from 10.0.2.4
    if (ip_header->daddr == *(unsigned int*)B_ip_address)
    {
        printk("How dare VM A try to access VM B? Away with you, data packet.");
        return NF_DROP;
    }
    //3. stop 10.0.2.4 from receiving or transmitting to ryerson.ca: 141.117.126.20
    if (ip_header->saddr == *(unsigned int*)ryerson_ip_address)
    {
        printk("How dare you try to access ryerson.ca? Away with you, data packet.");
        return NF_DROP;
    }
    //4.
    if(ip_header->saddr == *(unsigned int*)example_ip_address)
    {
        printk("How dare you try to access example.com? Away with you, data packet.");
        return NF_DROP;
    }
    //5.
    if(ip_header->saddr == *(unsigned int*)facebook_ip_address)

```

```

{
    printk("How dare you try to access facebook.com? Away with you, data packet.");
    return NF_DROP;
}
return NF_ACCEPT; //this one accepts all packets
}

// Init routine, called when module loaded using 'insmod'
int init_module(){ //fill hook structure:
    nfho.hook = hook_func;
    nfho.hooknum = NF_INET_PRE_ROUTING; //called right after packet recieved, first hook in
Netfilter
    nfho.pf = PF_INET;                //IPV4 packets
    nfho.priority = NF_IP_PRI_FIRST;   //set to highest priority over all other hook functions
    nf_register_hook(&nfho);           //register hook

    return 0;
}

//Cleanup
void cleanup_module(){
    nf_unregister_hook(&nfho);
}

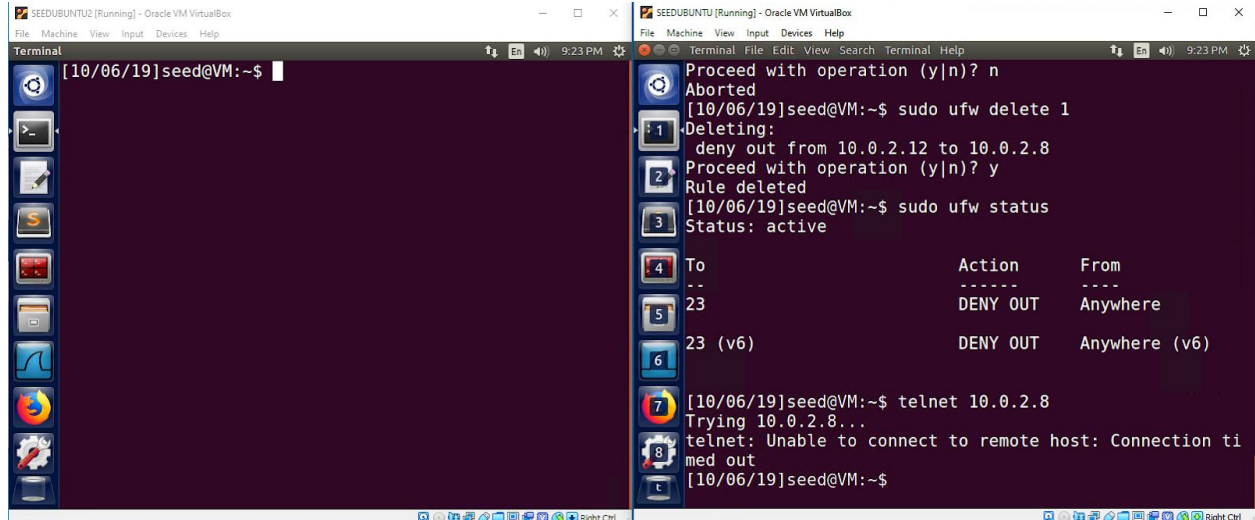
```

## Task 3 Evade Egress Filtering

1. Block outgoing traffic to telnet servers.

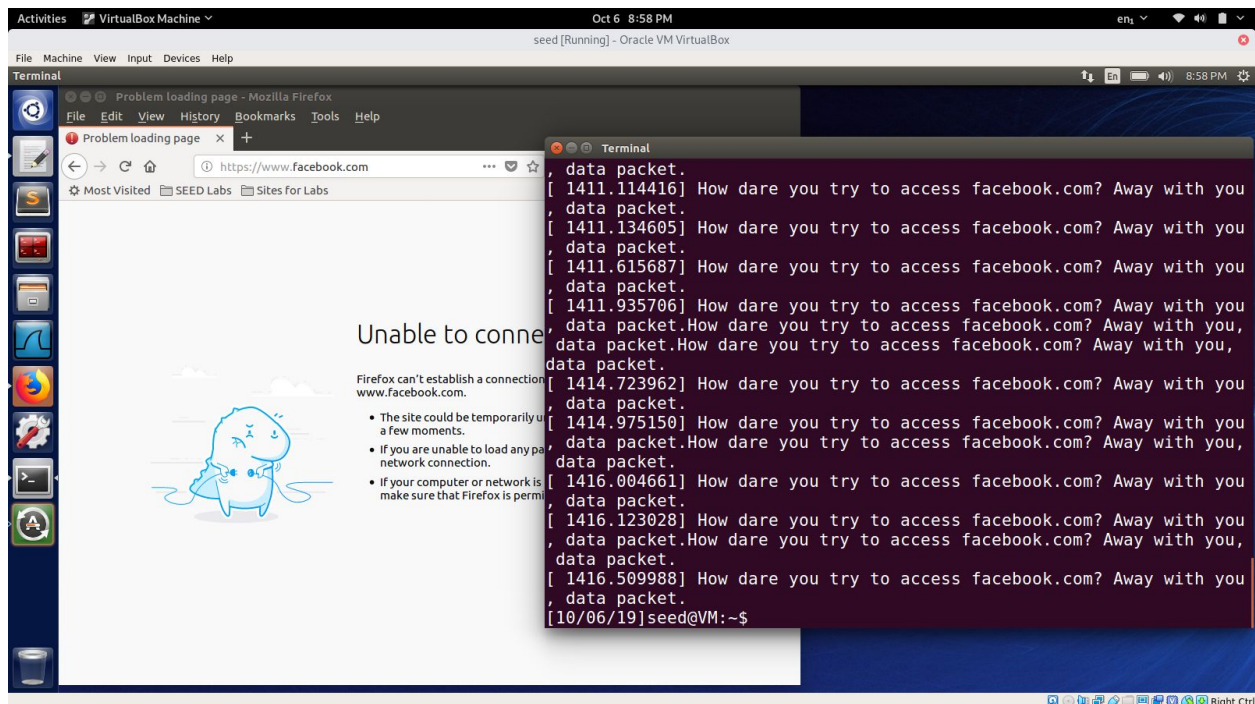
This can be done by running `ufw deny out 23` on machine A. We then run the telnet server on machine B. We cannot connect to B from A. As expected, the telnet request from VM A (right) to VM B (left) times out:





## 2. Blocking traffic to facebook.com

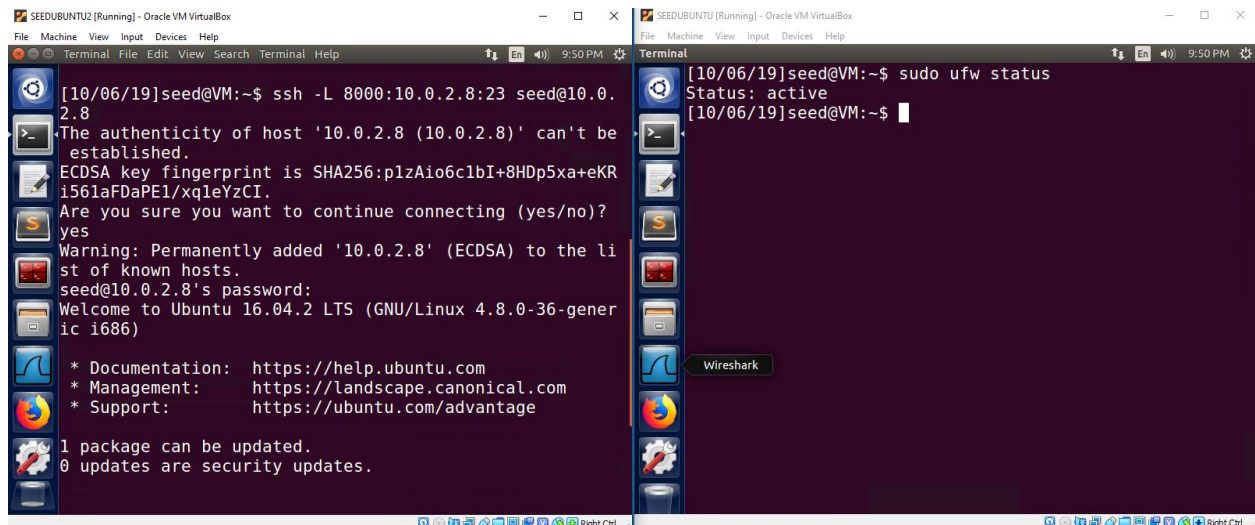
We have implemented this in the Netfilter firewall. Connecting to facebook.com will give us this:



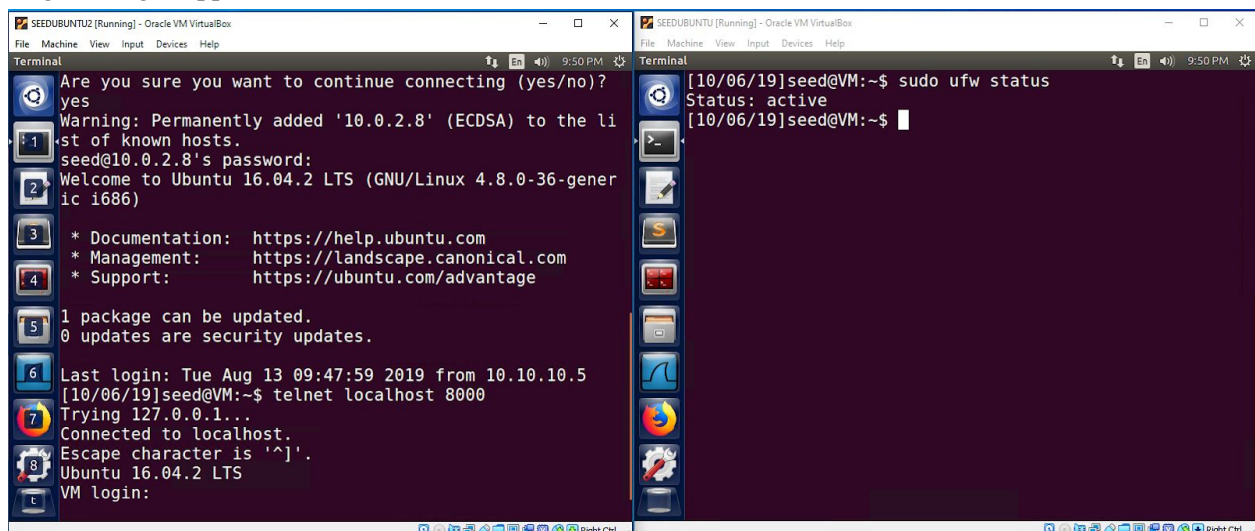
We cannot connect to facebook.com

## 3.a Telnet to B through firewall

We ended up using 2 VMs. From VM A (left), we are not able to telnet VM B (right) because outgoing packets to port 23 are dropped by the firewall. So we instead used an SSH tunnel because requests to port 22 are not being blocked by the firewall.



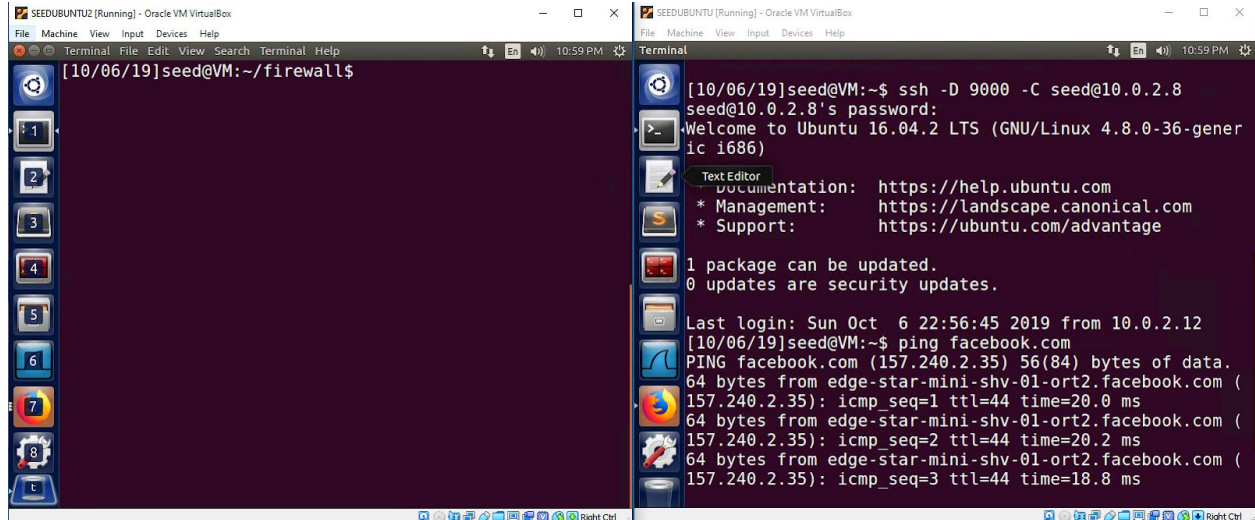
After we remotely accessed B from A using 'ssh -L 8000:10.0.2.8:23 10.0.2.8', our telnet requests are no longer being dropped, as seen here:



### 3.b SSH to facebook through firewall

1. Run Firefox and go visit the Facebook page. Can you see the Facebook page? Describe your observation.

- Now, we'll SSH into VM B and configure firefox proxy to go through port 9000
- Once we've remotely accessed B, we can connect to facebook:



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

- From before, we ran the firewall from before using 'sudo insmod hello-netfilter.ko'
- As expected, facebook.com will not load on this vm because the firewall is dropping the packets.
- We can see that the firewall is causing this by tailing the logfile: 'dmesg | tail -10':

```
[ 7907.671364] How dare you try to access facebook.com?
Away with you, data packet.
[ 7908.137443] How dare you try to access facebook.com?
Away with you, data packet.
[10/06/19]seed@VM:~$
```

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

- Once again, we remotely access VM B, configure firefox, and then access facebook.com
- And of course, there's no firewall on VM B, so we can connect to facebook just fine.

4. Explain what you have observed, especially on why the SSH tunnel can help bypass the egress filtering. You should use Wireshark to see what exactly is happening on the wire. Describe your observations and explain them using the packets that you have captured.

- VM A has a firewall that drops packets going to facebook.com.
- We got past this by remotely accessing another machine which does not have a firewall on it.
- We then configured firefox to connect to port 9000 as a proxy
- So we've successfully accessed facebook by requesting it indirectly through VM B.
- Wireshark confirms this. Without the tunnel, the requests to facebook.com do not get a response
- But the packets to VM B from A are not being dropped.



## Task 4: Evade Ingress Filtering

Machine A runs a web server behind a firewall; so only the machines in the internal network can access this web server. You are working from home and need to access this internal web server. You do not have VPN, but you have SSH, which is considered as a poor man's VPN. You do have an account on Machine A (or another internal machine behind the firewall), but the firewall also blocks incoming SSH connection, so you cannot SSH into any machine on the internal network. Otherwise, you can use the same technique from Task 3 to access the web server. The firewall, however, does not block outgoing SSH connection, i.e., if you want to connect to an outside SSH server, you can still do that. The objective of this task is to be able to access the web server on Machine A from outside. We will use two machines to emulate the setup. Machine A is the internal computer, running the protected web server; Machine B is the outside machine at home. On Machine A, we block Machine B from accessing its port 80 (web server) and 22 (SSH server). You need to set up a reverse SSH tunnel on Machine A, so once you get home, you can still access the protected web server on A from home.

On Machine A, Machine B (10.0.2.4 in this case) is blocked from accessing ports 22 and 80

```
[10/06/19]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
Anywhere DENY 10.0.2.4 22
Anywhere DENY 10.0.2.4 80

[10/06/19]seed@VM:~$
```

Setup the reverse ssh tunnel on Machine A (10.0.2.5) using `ssh -R 2222:localhost:22 seed@10.0.2.5`

```
[10/06/19]seed@VM:~$ ssh -R 2222:localhost:22 seed@10.0.2.5
The authenticity of host '10.0.2.5 (10.0.2.5)' can't be
established.
ECDSA key fingerprint is SHA256:plzAio6clbI+8HDp5xa+eKR
i561aFDaPE1/xqleYzCI.
Are you sure you want to continue connecting (yes/no)?
yes
Warning: Permanently added '10.0.2.5' (ECDSA) to the li
st of known hosts.
seed@10.0.2.5's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-gener
ic 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.
```

Then ssh from Machine A to Machine A using `ssh seed@localhost -p 2222`

```
* 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: Sun Oct  6 22:23:22 2019 from 127.0.0.1
[10/06/19]seed@VM:~$ ssh seed@localhost -p 2222
seed@localhost's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-gener
ic 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: Sun Oct  6 22:45:13 2019 from 127.0.0.1
[10/06/19]seed@VM:~$ █
```

Now Machine B can access Machine A using `ssh -p 2222 seed@localhost`:

```
[10/06/19]seed@VM:~$ ssh -p 2222 seed@localhost
seed@localhost's password:
Connection to localhost closed by remote host.
Connection to localhost closed.
[10/06/19]seed@VM:~$ ssh -p 2222 seed@localhost
seed@localhost'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: Sun Oct  6 22:13:48 2019 from 127.0.0.1
[10/06/19]seed@VM:~$ █
```