

Lab 17: Linux Firewall Exploration Lab

By Michael Minogue

4/28/20

—

CSP 544

—

A20388449

For the following tasks, Machine A has an IP of 10.0.2.15, and Machine B has an IP of 10.0.2.5. This was verified using Wireshark onto their telnet communications.

Task 1: Using Firewall

As discussed in the lab document, we will be configuring ufw to control our firewall. This file can be found in the /etc/default/ufw file on our Machine A. This is important so that our default policy is to accept, rather than drop.

```
ufw
1 # /etc/default/ufw
2 #
3
4 # Set to yes to apply rules to support IPv6 (no m
5 # accepted). You will need to 'disable' and then
6 # the changes to take affect.
7 IPV6=yes
8
9 # Set the default input policy to ACCEPT, DROP, o
10 # you change this you will most likely want to ad
11 DEFAULT_INPUT_POLICY="ACCEPT"
12
13 # Set the default output policy to ACCEPT, DROP,
14 # you change this you will most likely want to ad
15 DEFAULT_OUTPUT_POLICY="ACCEPT"
16
17 # Set the default forward policy to ACCEPT, DROP
18 # if you change this you will most likely want to
19 DEFAULT_FORWARD_POLICY="DROP"
20
21 # Set the default application policy to ACCEPT, D
22 # note that setting this to ACCEPT may be a secur
23 # details
```

Next, I will configure ufw so as to prevent certain connections from our lab assessment.

a. Prevent A from doing telnet to Machine B

First we will be preventing A from telnetting into machine B. This will be done by performing filtering on outgoing traffic from 10.0.2.15, stopping it from sending in its tracks.

```
[04/29/20]seed@VM:~$ sudo ufw deny from 10.0.2.15 to 10.0.2.5
Rule added
[04/29/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
Anywhere DENY 12.0.2.5
Anywhere DENY 10.0.2.5
10.0.2.5 DENY 10.0.2.15

[04/29/20]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
```

This code is pretty self-explanatory. We are denying traffic with a source destination in 10.0.2.15, aimed at 10.0.2.5. As you can see, telnetting to 10.0.2.5 now fails; a success!

b. Prevent B from doing telnet to Machine A.

This is essentially the same process, just in reverse. We now filter out traffic that is incoming to 10.0.2.15 and outgoing from 10.0.2.5.

The image shows two terminal windows. The left window shows the configuration of UFW rules to deny traffic from 10.0.2.5. The right window shows a telnet attempt from 10.0.2.5 to 10.0.2.15, which is blocked by the firewall.

```
[04/29/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[04/29/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
Anywhere DENY 12.0.2.5

[04/29/20]seed@VM:~$ sudo ufw deny from 10.0.2.5
Rule added
[04/29/20]seed@VM:~$ sudo ufw status
No command 'sdo' found, did you mean:
Command 'sdoc' from package 'ruby-sdoc' (universe)
Command 'sudo' from package 'sudo-ldap' (universe)
Command 'sudo' from package 'sudo' (main)
Command 'cdo' from package 'cdo' (universe)
Command 'sdb' from package 'sdb' (universe)
Command 'sdop' from package 'sdop' (universe)
Command 'udo' from package 'udo' (universe)
Command 'sd' from package 'sd' (universe)
Command 'sds' from package 'simh' (universe)
Command 'sdf' from package 'sdf' (universe)
Command 'sdc' from package 'hpsockd' (universe)
sdo: command not found
[04/29/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
Anywhere DENY 12.0.2.5
Anywhere DENY 10.0.2.5

[04/29/20]seed@VM:~$
```

```
[04/29/20]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: seed
Password:
Last login: Wed Apr 29 19:06:57 EDT 2020 from 10.0.2.5 on pts/19
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.

[04/29/20]seed@VM:~$ exit
logout
Connection closed by foreign host.
[04/29/20]seed@VM:~$ telnet 10.0.2.15
Trying 10.0.2.15...
```

As you can see I altered things slightly in that I banned all traffic from machine B, but this ultimately has the same effect: another success that stops telnet from connecting.

c. Prevent A from visiting an external website.

I decided to do the website Wired for this task, and my research is shown below.

Find IP Address

Test results : - wired.com

| |
|------------------------------------|
| Result URL: wired.com |
| Load Time: 597 ms |
| Tested on: 29 Apr 2020 06:59:37 PM |

Check Website

Find IP Results: 29 Apr 2020 06:59:37 PM

| S. No. | Domain Name | IP Address |
|--------|-------------|---------------------------|
| 1 | wired.com | ired.com./151.101.130.194 |
| 2 | wired.com | ired.com./151.101.66.194 |
| 3 | wired.com | ired.com./151.101.2.194 |
| 4 | wired.com | ired.com./151.101.194.194 |

Related System Administrator and Network Tools

The terminal window shows the configuration of UFW rules to deny outgoing traffic to the IP addresses of wired.com. It also shows a ping attempt to wired.com, which fails due to firewall rules.

```
[04/29/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[04/29/20]seed@VM:~$ sudo ufw status
Status: active
[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.15 to 151.101.194.194
Rule added
[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.15 to 151.101.2.194
Rule added
[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.15 to 151.101.66.194
Rule added
[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.15 to 151.101.130.194
Rule added
[04/29/20]seed@VM:~$ sudo ufw status
Status: active

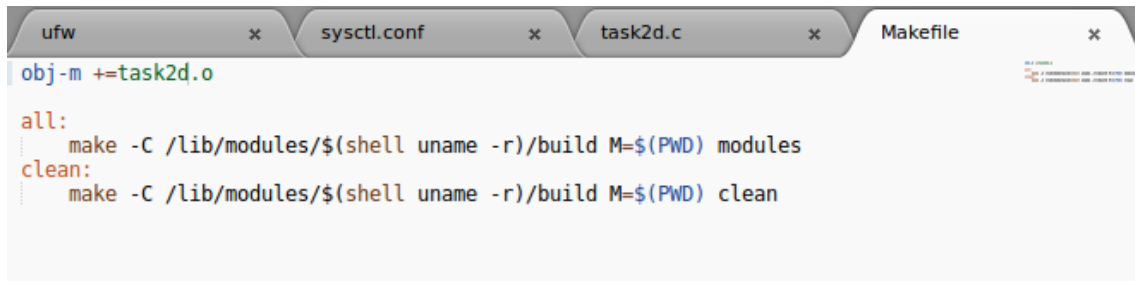
To Action From
--
151.101.194.194 DENY OUT 10.0.2.15
151.101.2.194 DENY OUT 10.0.2.15
151.101.66.194 DENY OUT 10.0.2.15
151.101.130.194 DENY OUT 10.0.2.15

[04/29/20]seed@VM:~$ ping wired.com
PING wired.com (151.101.194.194) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
```


So what exactly is happening here? Well as you can see, wired.com has 4 IP addresses. I went ahead and blocked all traffic outgoing from machine A towards ANY of those 4 IP addresses. As a result, we receive the method “operation not permitted” when we attempt to ping wired.com, demonstrating that such behavior is no longer allowed for machine A.

Task 2: Implementing a Simple Firewall

All of the following tasks follow a general structure for both the task2.c file and the make file, whose structure was gotten directly from our lab document, and I’ve pasted below. Code alterations are posted for each task as well to demonstrate progress over time.



```
obj-m +=task2d.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
```

Task 2a: Preventing A from telnetting to B

Ultimately our code has two major tasks: to implement the operation of the filter itself, and the parameters that filter will follow. If the packet follows drop criteria, the drop command should be triggered on that packet. If otherwise, we will accept that packet.



```
if(iph->protocol == IPPROTO_TCP && tcph->dest == htons(23) && iph->saddr
{
    return NF_DROP;
}
else
{
    return NF_ACCEPT;
}
}

20
21
22
23 s(23) && iph->saddr == in_aton("10.0.2.15") && iph->daddr == in_aton("10.0.2.5"))
24
25
26
```

Above you have our first part. It sorts Netfilter packets into ACCEPT or DROP categories. It is currently set to drop TCP packets from 10.0.2.15 to 10.0.2.5 to prevent telnetting from A to B.



```
int createFilter(void)
{
    printk(KERN_INFO "Generating filter\n");
    telnetFilterHook.hook = telnetFilter;
    telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
    telnetFilterHook.priority = NF_IP_PRI_FIRST;
    telnetFilterHook.pf = PF_INET;
    nf_register_net_hook(&init_net,&telnetFilterHook);
    return 0;
}

void destroyFilter(void)
{
    printk(KERN_INFO "Boom: Filter Destroyed!\n");
    nf_unregister_net_hook(&init_net,&telnetFilterHook);
}

module_init(createFilter);
module_exit(destroyFilter);
```

There we have our functions that create and destroy the filter respectively, following the format listed in the lab document and linked Netfilter documentation. The system notifies the kernel when the filter is created or destroyed, and sets the telnetFilter above as the filter to be implemented.

We make it using the Makefile discussed previously, and our attack is a success!

```
[04/29/20]seed@VM:~$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
  CC [M] /home/seed/task2a.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC /home/seed/task2a.mod.o
  LD [M] /home/seed/task2a.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/29/20]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
^C
[04/29/20]seed@VM:~$ sudo ufw status
Status: inactive
[04/29/20]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: seed
Password: Connection closed by foreign host.
[04/29/20]seed@VM:~$ sudo insmod task2a.ko
[04/29/20]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
```

Task 2b: Preventing B from telnetting to A

The only part we really need to adjust in the following examples (other than pointing the Makefile to the updated C code) is the telnetFilter code. The other code, for creating the structures and filters, will basically remain the same.

We now do the opposite of before, filtering 10.0.2.5 to 10.0.2.15.

```
ufw x sysctl.conf x task2a.c x task2b.c x Makefile
7 #include <linux/ip.h>
3 #include <linux/tcp.h>
9
9 #include <linux/inet.h>
1
2 static struct nf_hook_ops telnetFilterHook;
3
4
5 unsigned int telnetFilter(void *priv, struct sk_buff *skb, const struct nf_hook_sta
6 {
7     struct iphdr *iph;
8     struct tcphdr *tcph;
9     iph = ip_hdr(skb);
10    tcph = (void *)iph+iph->ihl*4;
11
12    if(iph->saddr == in_aton("10.0.2.5") && iph->daddr == in_aton("10.0.2.15"))
13    {
14        return NF_DROP;
15    }
16    else
17    {
18        return NF_ACCEPT;
19    }
20 }
```

```
[04/29/20]seed@VM:~$ telnet 10.0.2.6
Trying 10.0.2.6...
```

This should prevent B from telnetting to A. After some initial difficulty, as the filter really didn't want to implement, I finally managed to get it!

Task 2c: Preventing A from connecting to a website

Again we only need to do a simple change, this time in the destination address. I decided to use the address of the Neopets website, as I had no idea it still existed and it initially looked to have only 1 IP address.

```
iph = ip_hdr(skb);
tcph = (void *)iph+iph->ihl*4;

if(iph->saddr == in_aton("10.0.2.15") && iph->daddr == in_aton("23.96.35.235"))
{
    return NF_DROP;
}
else
{
    return NF_ACCEPT;
}
```

It looks as if the end server is rejecting our ping in any case, even before the firewall, but you can see it was implemented correctly, as attempting to use it after installation causes an error message to occur.

```
[04/29/20]seed@VM:~$ make
make -C /lib/modules/4.8.0-36-generic/build M=/home/seed modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-36-generic'
CC [M] /home/seed/task2c.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/seed/task2c.mod.o
LD [M] /home/seed/task2c.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-36-generic'
[04/29/20]seed@VM:~$ ping neopets.com
PING neopets.com (23.96.35.235) 56(84) bytes of data.
^C
--- neopets.com ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4087ms

[04/29/20]seed@VM:~$ sudo insmod task2c.ko
[04/29/20]seed@VM:~$ ping neopets.com
PING neopets.com (23.96.35.235) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- neopets.com ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5114ms

[04/29/20]seed@VM:~$
```

Task 2d: Preventing A from connecting via SSH

This task is slightly more complicated as it involves port addresses, but only ever so slightly. Again, this is just an adjustment of the filtering rules, this time so that any outgoing packets from machine A aimed at port 22 (which is used for SSH) are automatically blocked.

First we have the code:

```
if(iph->saddr == in_aton("10.0.2.15") && tcph->dest == htons(22))
{
    return NF_DROP;
}
else
{
    return NF_ACCEPT;
}
```

Below you can see the effects of ssh before and after the module is installed. Before we can ssh to machine B without any problem, but now ssh is a complete failure, as the firewall prevents transmission.

```
[04/29/20]seed@VM:~$ ssh 10.0.2.5
seed@10.0.2.5'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: Wed Apr 29 21:40:19 2020 from 10.0.2.15
[04/29/20]seed@VM:~$ exit
logout
Connection to 10.0.2.5 closed.
[04/29/20]seed@VM:~$ sudo insmod task2d.ko
[04/29/20]seed@VM:~$ ssh 10.0.2.5
```

Task 2e: Preventing A from sending ICMP packets

This one involves a protocol filtering change, as we wish to search for any ICMP packets from A and reject them. First we have the code adjusted to reflect this.

```
if(iph->saddr == in_aton("10.0.2.15") && iph->protocol == IPPROTO_ICMP)
{
    return NF_DROP;
}
else
{
    return NF_ACCEPT;
}
```

As you can see, we set the source address as 10.0.2.15, and any packet with this source and ICMP status is rejected immediately. As ping utilizes ICMP it is a perfect avenue for testing, and I've done this below.

[illegible]

As you can see, the machine can ping 10.0.2.5 without issue before the module is installed. After its installation, it blocks the ICMP message coming out of A, leading to another “operation not permitted” message.

Task 3: Evading Egress Filtering

Each firewall implementation is listed in its respective area. We will begin by creating an SSH tunnel to bypass the firewall to machine B, then to Wired.com as in previous tasks.

3.a: Telnet to Machine B through the firewall

Ufw is used to deny any outgoing TCP connections through port 23, blocking machine A from using telnet to connect. As you can see, normal connection to 10.0.2.5 will now fail.

```
[04/29/20]seed@VM:~$ sudo ufw deny out 23/tcp
Rules updated
Rules updated (v6)
[04/29/20]seed@VM:~$ sudo ufw status
Status: inactive
[04/29/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[04/29/20]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
```

We need to create a tunnel that will allow us to still telnet to localhost even with this block in place. To do so, I follow the formula outlined on the lab document, using 10.0.2.6 as my machine C.


```

[04/29/20]seed@VM:~$ ssh -L 8000:10.0.2.6:23 10.0.2.5
seed@10.0.2.5'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: Wed Apr 29 21:42:55 2020 from 10.0.2.15
[04/29/20]seed@VM:~$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Apr 29 22:05:04 EDT 2020 from 10.0.2.15 on pts/0
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.

[04/29/20]seed@VM:~$

```

As you can see, telnetting to localhost is a success! When we attempt to connect to local host at 127.0.0.1 we are able to, despite our block normally preventing telnet access here.

| Source | Destination | Protocol | Length | Info |
|-----------|-------------|----------|--------|---------------------------|
| 10.0.2.5 | 10.0.2.15 | SSHv2 | 104 | Server: Encrypted packet |
| 10.0.2.15 | 10.0.2.5 | TCP | 68 | 39506 → 22 [ACK] Seq=2833 |
| 10.0.2.5 | 10.0.2.15 | SSHv2 | 136 | Server: Encrypted packet |
| 10.0.2.15 | 10.0.2.5 | TCP | 68 | 39506 → 22 [ACK] Seq=2833 |
| 10.0.2.5 | 10.0.2.15 | SSHv2 | 104 | Server: Encrypted packet |
| 10.0.2.5 | 10.0.2.15 | SSHv2 | 104 | Server: Encrypted packet |
| 10.0.2.15 | 10.0.2.5 | TCP | 68 | 39506 → 22 [ACK] Seq=2833 |
| 10.0.2.5 | 10.0.2.15 | SSHv2 | 128 | Server: Encrypted packet |

▶ Frame 174: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface
 ▶ Linux cooked capture
 ▶ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.5
 ▶ Transmission Control Protocol, Src Port: 39506, Dst Port: 22, Seq: 2833061697,

| | | | |
|------|-------------------------|-------------------------|------------------|
| 0000 | 00 04 00 01 00 06 08 00 | 27 ed ba 12 00 00 08 00 |'. |
| 0010 | 45 10 00 34 83 55 40 00 | 40 06 9f 4b 0a 00 02 0f | E..4.U@. @..K... |
| 0020 | 0a 00 02 05 9a 52 00 16 | a8 dd 17 41 b5 d2 63 e7 |R... ..A..c. |
| 0030 | 80 10 01 22 18 3a 00 00 | 01 01 08 0a 00 2c e5 08 | ..."..... |
| 0040 | 00 2c b4 33 | | ..3 |

| | | | | |
|-----------|-----------|-------|------|----------------------------|
| 10.0.2.15 | 10.0.2.5 | TCP | 76 | 39506 → 22 [SYN] Seq=28330 |
| 10.0.2.5 | 10.0.2.15 | TCP | 76 | 22 → 39506 [SYN, ACK] Seq= |
| 10.0.2.15 | 10.0.2.5 | TCP | 68 | 39506 → 22 [ACK] Seq=28330 |
| 10.0.2.15 | 10.0.2.5 | SSHv2 | 109 | Client: Protocol (SSH-2.0- |
| 10.0.2.5 | 10.0.2.15 | TCP | 68 | 22 → 39506 [ACK] Seq=30504 |
| 10.0.2.5 | 10.0.2.15 | SSHv2 | 109 | Server: Protocol (SSH-2.0- |
| 10.0.2.15 | 10.0.2.5 | TCP | 68 | 39506 → 22 [ACK] Seq=28330 |
| 10.0.2.5 | 10.0.2.15 | SSHv2 | 1044 | Server: Key Exchange Init |

Our Wireshark capture shows that our TCP traffic is being transmitted encrypted over port 22. Not only does this circumvent the port 23 normally accessed by telnet, but SSH encryption also means our packets are harder to read by the filter, so it cannot see our final destination is localhost.

3.b: Connect to Facebook using SSH Tunnel

Ultimately I decided to use Wired.com instead of Facebook, as it still has multiple IPs and I've already researched its structure.

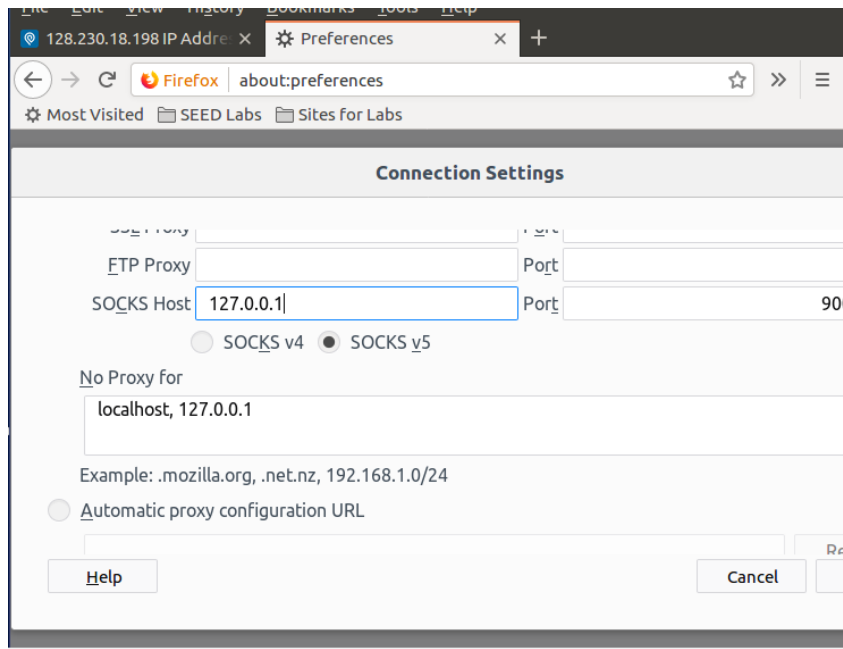
```
[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.15 to 151.101.194.194
Rules updated
[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.15 to 151.101.2.194
Rules updated
[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.15 to 151.101.66.194
Rules updated
[04/29/20]seed@VM:~$ sudo ufw deny out from 10.0.2.15 to 151.101.130.194
Rules updated
[04/29/20]seed@VM:~$ ssh -D 9000 -C 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:plzAio6c1bI+8Hdp5xa+eKRi561aFDaPE1/xqleYzCI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.5' (ECDSA) to the list of known hosts.
seed@10.0.2.5'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

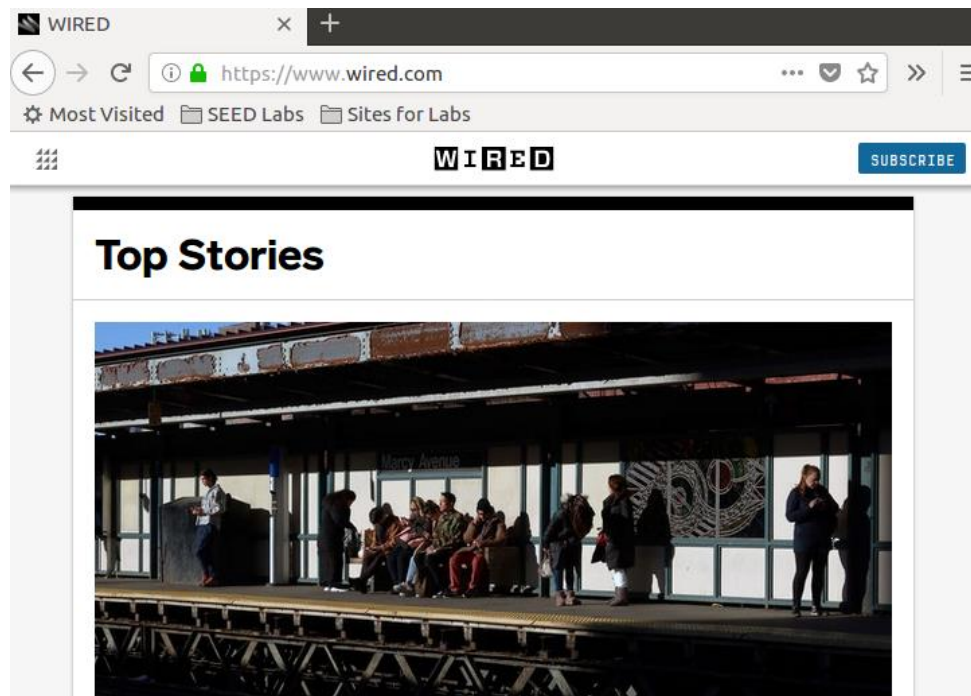
1 package can be updated.
0 updates are security updates.

Last login: Wed Apr 29 22:15:09 2020 from localhost
[04/29/20]seed@VM:~$
```

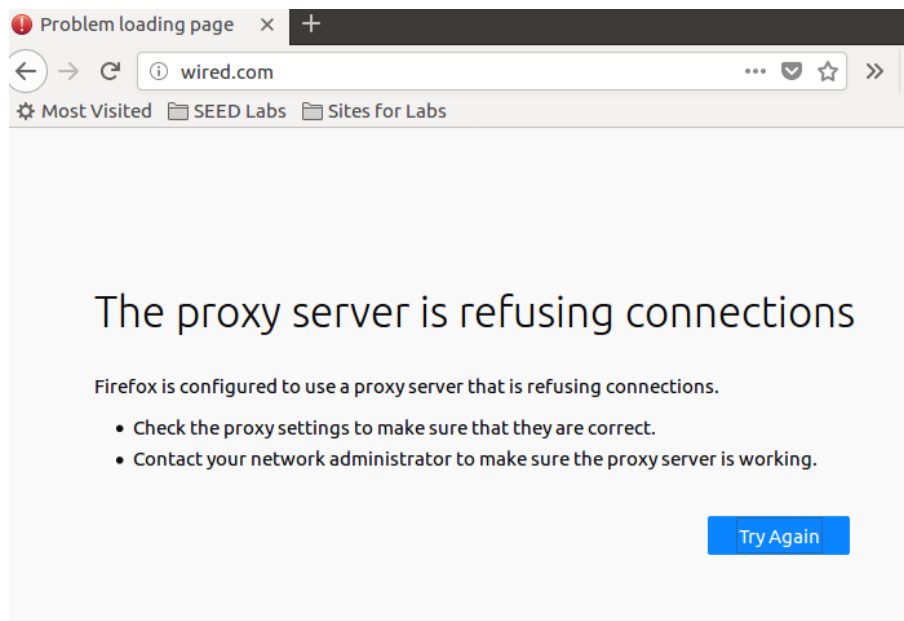
As in task 1 I set rules to deny all outgoing access from machine A to Wired, preventing its access. Firefox is routed as described on the lab document, so as to utilize a SOCKS proxy.



With the SSH tunnel in place, we can still access Wired without issue.



Once this connection is broken, however, things change. We are now no longer able to access Wired, and see a message notifying us it is blocked instead.



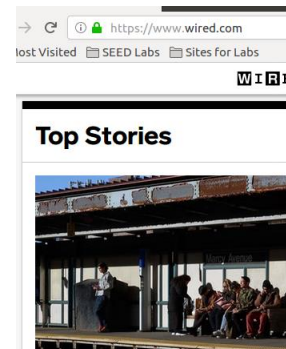
We now reestablish the SSH tunnel, to see if we can reconnect to Wired. As expected, we are now able to see the page again!

```
[04/29/20]seed@VM:~$ ssh -D 9000 -C seed@10.0.2.5
seed@10.0.2.5'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: Wed Apr 29 22:38:56 2020 from localhost
[04/29/20]seed@VM:~$ sudo ufw status
Status: inactive
```



What exactly is happening here? Well looking at Wireshark conveniently shows us the answer. We are routing packets through Machine B to the web server, causing traffic to appear as if its coming from 10.0.2.5 when its actually coming from 10.0.2.15. This causes the filter to not see the correct source, so our packets move on undisturbed to their final destination.

| Apply a display filter ... <Ctrl-/> | | | | | Expression... |
|-------------------------------------|--------------------------------|-----------------|-----------------|---|---------------|
| No. | Time | Source | Destination | | |
| 5315 | 2020-04-29 22:44:17.2313089... | 10.0.2.5 | 10.0.2.15 | S | |
| 5316 | 2020-04-29 22:44:17.2316749... | 10.0.2.15 | 10.0.2.5 | T | |
| 5317 | 2020-04-29 22:44:18.1699832... | 10.0.2.5 | 151.101.130.194 | I | |
| 5318 | 2020-04-29 22:44:18.2861776... | 151.101.130.194 | 10.0.2.5 | I | |
| 5319 | 2020-04-29 22:44:18.2862953... | 10.0.2.5 | 10.0.2.5 | S | |
| 5320 | 2020-04-29 22:44:18.2863015... | 10.0.2.5 | 10.0.2.5 | T | |
| 5321 | 2020-04-29 22:44:18.2864110... | 127.0.0.1 | 127.0.0.1 | T | |
| 5322 | 2020-04-29 22:44:18.2864150... | 127.0.0.1 | 127.0.0.1 | T | |
| 5323 | 2020-04-29 22:44:18.2864700... | 10.0.2.5 | 10.0.2.15 | S | |
| 5324 | 2020-04-29 22:44:18.2868009... | 10.0.2.15 | 10.0.2.5 | T | |
| 5325 | 2020-04-29 22:44:19.1716171... | 10.0.2.5 | 151.101.130.194 | I | |
| 5326 | 2020-04-29 22:44:19.1855708... | 151.101.130.194 | 10.0.2.5 | I | |
| 5327 | 2020-04-29 22:44:19.1856991... | 10.0.2.5 | 10.0.2.5 | S | |

Not once does 10.0.2.15 connect directly to Wired.com (151.101.130.194). Every time it is 10.0.2.5 that connects, and eventually passes the information back to 10.0.2.15 for further instruction.

Task 4: Evading Ingress Filtering

We can begin by figuring out the ground rules that our victim machine, A, follows. Specifically, it is to block all incoming SSH and web server traffic, which goes over ports 80 and 22.

The first thing I will do is set new ufw traffic rules on Machine A, denying this incoming traffic.

```
[04/29/20]seed@VM:~$ sudo ufw deny in ssh
Rules updated
Rules updated (v6)
[04/29/20]seed@VM:~$ sudo ufw deny in http
Rules updated
Rules updated (v6)
[04/29/20]seed@VM:~$
```

With these new rules in place, outside systems cannot SSH into our VM. This is shown in action below.


```
Terminal File Edit View Search Terminal Help
[04/30/20]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[04/30/20]seed@VM:~$ sudo ufw status
Status: active
[04/30/20]seed@VM:~$ sudo ufw deny in http
Rule added
Rule added (v6)
[04/30/20]seed@VM:~$ sudo ufw deny in ssh
Rule added
Rule added (v6)
[04/30/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
80 DENY Anywhere
22 DENY Anywhere
80 (v6) DENY Anywhere (v6)
22 (v6) DENY Anywhere (v6)

[04/30/20]seed@VM:~$
```

```
CSP544 Clone [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal File Edit View Search Terminal Help
[04/30/20]seed@VM:~$ ssh 10.0.2.15
```

Despite the block, we are still able to open up our SSH tunnel out of the machine into machine B. This tunnel has a few parts:

1. We set the connection method through port SSH.
2. We create a channel through port 5000 of our local host that information can be routed through.
3. We direct the user [seed@10.0.2.5](#) as the operator of this reverse tunnel.

```
[04/30/20]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
22 DENY Anywhere
80 DENY Anywhere
22 (v6) DENY Anywhere (v6)
80 (v6) DENY Anywhere (v6)

[04/30/20]seed@VM:~$ ssh -p 22 -qngfNTR 5000:localhost:22 seed@10.0.2.5
seed@10.0.2.5's password:
[04/30/20]seed@VM:~$
```

We then connect via the other end to local host, listening in on port 5000.

```
[04/30/20]seed@VM:~$ sudo ssh -p 5000 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: Thu Apr 30 00:51:23 2020 from 127.0.0.1
[04/30/20]seed@VM:~$
```

Ultimately we have success, as with some more troubleshooting I am successfully able to get access to my main file repository via my clone/B VM!

```
[04/30/20]seed@VM:~$ ls
android          Lab2             procombo         spoof_icmp.c     task2c.o
bin              Lab3             procombo.c       spoof_tcp.c      task2d.c
checksum.c       Lab4             prosniff         spoof_udp.c      task2d.ko
Customization    Lab5             prosniff.c       task2a.c         task2d.mod.c
Desktop          Lab9             psniff           task2a.ko        task2d.mod.o
Documents        lib              psniff.c         task2a.mod.c     task2d.o
Downloads        Makefile         Public           task2a.mod.o     task2e.c
elf-hijack       Meltdown_Attack sniffer2.py      task2a.o         task2e.ko
examples.desktop modules.order     sniffer3.py     task2b.c         task2e.mod.c
get-pip.py       Module.symvers   sniffer.py      task2b.ko        task2e.mod.o
icmp_spoof       Music            snifferspoofer  task2b.mod.c     task2e.o
icmp_spoof       myheader.h       snifferspoofer.c task2b.mod.o     Templates
icmp_spoof       passsniff        sniffspoof.py   task2b.o         udp_client.c
icmp_spoof       passsniff.c      source          task2c.c         udp_spoof
icmp_spoof       passsniffer      spoof.c         task2c.ko        update-binaryold
Lab11            passsniffer.c    spoofer.py      task2c.mod.c     useful
Lab1Files        Pictures         spoofertace.py  task2c.mod.o     Videos
[04/30/20]seed@VM:~$
```

Judging from Wireshark, we are essentially retransmitting packets over SSH to Machine B via our execution on Machine A, causing us to be able to access our data on a second machin.