Linux Firewall Exploration Lab

Table of Contents:

Task 1: Using Firewall

Task 2: How Firewall Works

Task 3: Evading Egress Filtering

- Task 3.a: Telnet to Machine B through the firewall
- Task 3.b: Connecting to Google using SSH tunnel

Task 4: Evade Ingress Filtering

Environment:

This lab is based on the SEED Ubuntu 16.04 VM. This lab requires multiple VMs to be setup. In the rest of the lab, we will be using 3 VMs with the following IPs:

VM 1: 10.0.2.9

VM 2: 10.0.2.10

(For later tasks) VM 3: 10.0.2.11

Task 1: Using Firewall

In this task, we will use two VMs (VM1 and VM2). We first show that VM1 can telnet to VM2.

Command:

```
telnet 10.0.2.10
```

Provide a screenshot of your observations.

We need to first prevent VM1 from being able to telnet to VM2. For this we will configure the firewall using ***ufw***. We will first enable the firewall on VM1.
Commands:

```
sudo ufw enable
```

```
sudo ufw status verbose
```

Provide screenshots of your observations.

We will next configure the firewall on VM1 to deny telnet (port 23) to VM2.
Command:

```
sudo ufw deny out from 10.0.2.9 to 10.0.2.10 port 23
```

```
sudo ufw status verbose
```

Provide a screenshot of your observations.

We now try to telnet from VM1 to VM2. Due to the firewall rule, the telnet is denied.
Command:

```
telnet 10.0.2.10
```

Provide a screenshot of your observations.

We next need to deny telnet from VM2 to VM1. We will first test whether telnet works from VM2 to VM1.
Command:

```
telnet 10.0.2.9
```

Provide a screenshot of your observations.

We next delete the firewall rule in VM1. We add a rule to prevent VM2 from being able to do telnet to VM1.
Commands:

```
sudo ufw delete 1
```

```
sudo ufw deny in from 10.0.2.10 to 10.0.2.9 port 23
```

Provide screenshots of your observations.

We can try to telnet from VM2 to VM1 again. The telnet is blocked and does not work.
Command:

```
telnet 10.0.2.9
```

Provide a screenshot of your observations.

We next need to block VM1 from visiting a website. We use the www.pes.edu website. We first find its IP address.
Command:

```
ping www.pes.edu
```

Provide a screenshot of your observations.

We also test whether the website is accessible to the browser.
Provide a screenshot of the page in the browser.

Before proceeding further, clear the browser cache.
We next add the firewall rule to prevent VM1 from accessing the IP address for www.pes.edu (216.10.247.185)
Commands:

```
sudo ufw delete 1
```

```
sudo ufw deny out to 216.10.247.185
```

Provide screenshots of your observations.

With the firewall rule in place, we next try to ping www.pes.edu. We see the message "***Operation not permitted***" because the firewall has blocked it.
Provide screenshot of your observations.

This can be verified by revisiting www.pes.edu in the Firefox browser. The page is not loaded.
Provide a screenshot of the browser and the terminal.

Task 2: How Firewall Works

In this task, we will develop a firewall using netfilter and LKM. We implement five rules in this firewall:

- Block telnet from VM1 to VM2
- Block telnet from VM2 to VM1
- Block external website access from VM1
- Block ssh from VM1 to VM2
- Block ssh from VM2 to VM1

Below is the code for the firewall:
lkm.c

Makefile:
The program can be compiled using the ***make*** command. (Note: put lkm.c & Makefile into a folder. Open a terminal by right clicking inside that folder. Type 'make').

The compiled kernel module (lkm.ko) can be inserted using insmod:
Commands:

```
sudo dmesg --clear
```

```
sudo insmod lkm.ko
```

```
lsmod | grep lkm
```

Provide a screenshot of your observations.

We first test whether ***telnet*** from VM1 to VM2 is blocked.
Command:

```
telnet 10.0.2.10
```

Provide a screenshot of your observations.

We next test whether ***telnet*** from VM2 to VM1 is blocked.
Commands:

```
telnet 10.0.2.9
```

```
dmesg | tail -10 (This command will show u dropping packets)
```

Provide screenshots of your observations.

We next test whether external website access (port 80) is allowed from VM1. We try to load the page www.pes.edu in the Firefox browser and the page does not load.
Provide a screenshot of the browser.
Command:

```
dmesg | tail -10
```

Provide a screenshot of your observations.

We next test whether ***ssh*** is blocked from VM1 to VM2.
Lastly, we test whether ssh is blocked from VM2 to VM1.
Command:

```
telnet 10.0.2.9
```

Provide a screenshot of your observations.

Task 3: Evading Egress Filtering

In this task, we will be using a ***ssh*** tunnel to evade egress filtering. Please delete all the firewall rules from the previous task using the ***ufw delete <rule_number>*** command.

Task 3.a: Telnet to Machine B through the firewall

In this lab, we will use three VMs. VM1 will be blocked from being able to telnet to VM2. We will utilize a ***ssh*** tunnel to allow VM1 to telnet to VM3 via VM2. The diagram below depicts the tunnel (in the diagram, the ***home*** machine is VM1, the ***apollo*** machine is VM2 and the ***work*** machine is VM3).

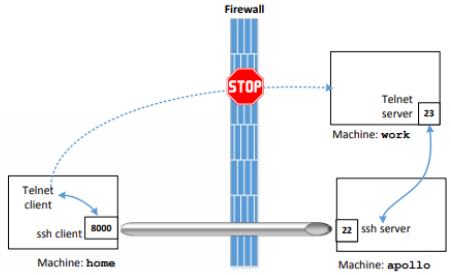


Figure 1: SSH Tunnel Example

We will first block VM1 from being able to ***telnet*** to any other machine.
Command:

```
sudo ufw enable
sudo ufw status verbose
sudo ufw deny out from 10.0.2.9 to any port 23
sudo ufw status verbose
```

Provide a screenshot of your observations.

The effect of the previous firewall rule can be observed below. The telnet is blocked.
Command:

```
telnet 10.0.2.11
```

Provide a screenshot of your observations.

We next setup a ***ssh*** tunnel between VM1 and VM2 to allow VM1 to telnet via VM2, evading the firewall on VM1. The ***ssh*** command below allows VM1 to use its local port 8000 to ***telnet*** to VM3 via VM2. (**before doing this task please keep wireshark open**)

Command:

```
ssh -L 8000:10.0.2.10:23 seed@10.0.2.11
```

Provide a screenshot of your observations.

With the ***ssh*** tunnel setup, we can now ***telnet*** from VM1 to VM3 even though the firewall policy on VM1 denies outgoing ***telnet***.
Open wireshark on all 3 VMs.

Command:

```
telnet localhost 8000
```

Provide a screenshot of your observations.

The wireshark traffic for the telnet from VM1 to VM3 via VM2 is to be shown.
Provide screenshot of VM1 wireshark capture.
Provide screenshot of VM2 Wireshark capture.
Provide screenshot of VM3 Wireshark capture.

Task 3.b: Connecting to Google using SSH tunnel

In this task, we will setup a firewall rule to block VM1 from visiting www.google.com but then leverage dynamic forwarding via ssh tunnel to visit www.google.com from VM1 via VM2.

Note: please delete other firewall rules

We first find the IP address of www.google.com a shown below:
Commands:

```
sudo ufw status verbose
```

```
ping www.google.com
```

Provide a screenshot of your observations.

Given the IP address, we can setup a firewall rule to block traffic to that IP.

Command:

```
sudo ufw deny out to 172.217.166.106
```

```
sudo ufw status verbose
```

Provide a screenshot of your observations.

With the firewall rule in place, we can try to ping www.google.com. The operation is not permitted because it is being blocked by the firewall.
Command:

```
ping www.google.com
```

Provide a screenshot of your observations.

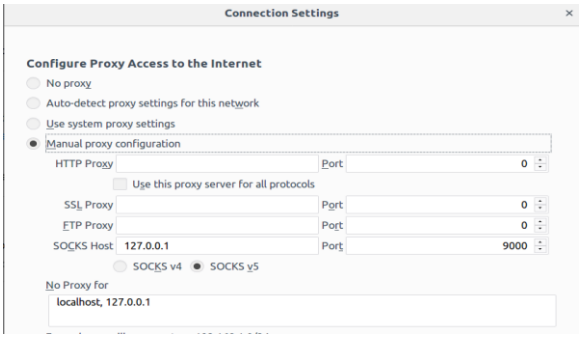
We can also try to visit www.google.com through the browser, where we are unable to load the page.
Provide screenshot of browser.

We now setup a ssh tunnel with dynamic port forwarding between VM1 and VM2. With this tunnel setup, VM1 will be able to use its local port 9000 to send a request to www.google.com via VM2.
Command:

```
ssh -D 9000 seed@10.0.2.11
```

Provide a screenshot of your observations.

To use the established tunnel, we need to set the proxy settings in the firefox browser as shown below.



With the proxy settings in place, we are able to visit www.google.com from VM1 even though the firewall has a policy to block it.
Provide screenshot of the browser.

Provide screenshot of VM1 and VM2 Wireshark capture.

We next disable the tunnel to see its effect.
Command:

```
ssh -D 9000 seed@10.0.2.11
```

Provide a screenshot of your observations.

We also clear the browser cache to make sure firefox does not show the webpage it just loaded.

If we try to visit www.google.com again, we observe that the browser informs the proxy is refusing connections. The browser is still configured to use proxy, but with the tunnel not running any more, the browser cannot use local port 9000 to get the result.

Provide a screenshot of the browser.

If we re-enable the proxy, the browser will get the desired result.

Command:

```
ssh -D 9000 seed@10.0.2.11
```

Provide a screenshot of your observations.

Provide a screenshot of the browser page.

Task 4: Evade Ingress Filtering

In this task, we will block incoming port 80 and port 22 on VM1, but still access a web page on the web server in VM1 from VM2 by using a reverse ***ssh*** tunnel.

Note: please delete the firewall rules from the previous tasks.

Our goal is to access a secret page on VM1 (test.html) from VM2. The content of the page is shown below:

```
seed@10.0.2.9:~$cat /var/www/html/test.html
<html>

<body>
This is a page on VM1 (10.0.2.9)
</body>
</html>
seed@10.0.2.9:~$
```

With no firewall rules setup on VM1, we try to access the page from VM2.
Command:

```
sudo ufw status verbose
```

Provide a screenshot of your observations.

Provide a screenshot of the browser.

We next block incoming requests on port 80 and port 22 on VM1.
Commands:

```
sudo ufw deny in from any to 10.0.2.9 port 80
sudo ufw deny in from any to 10.0.2.9 port 22
sudo ufw status verbose
```

Provide a screenshot of your observations.

Given the firewall rules in place on VM1, we next check if the page is still accessible. We clear the browser cache.
If we try to access the page again from VM2, the page is no longer accessible.
Provide a screenshot of the browser.

Also, because port 22 is blocked, VM2 cannot ***ssh*** to connect to VM1.
Command:

```
ssh seed@10.0.2.9
```

Provide a screenshot of your observations.

We next set up a reverse tunnel. Using this VM2 can use its local port 9000 to access port 80 on VM1.

Command:

```
ssh -R 9000:10.0.2.9: 80 10.0.2.10
```

Provide a screenshot of your observations

With the tunnel setup, we test whether we can access the webpage using port 9000 on VM2.

Provide a screenshot of the browser on VM2.

We next break the tunnel to see its effect (on VM1)

Command:

```
exit
```

Provide a screenshot of your observations.

With the tunnel broken, we can no longer access the page on VM1 from VM2.
Provide a screenshot of the browser on VM2.

Submission:

Students need to submit a detailed lab report to describe what they have done, what they have observed, and explanation. Reports should include the evidence to support the observations. Evidence include packet traces, screen shots, etc. Students also need to answer all the questions in the lab description. For the programming tasks, students should list the important code snippets followed by explanation. Simply attaching code without any explanation is not enough.