Linux Firewall Exploration Lab

1 Overview

The learning objective of this lab is for students to gain the insights on how firewalls work by playing with firewall software and implement a simplified packet filtering firewall. Firewalls have several types. In this lab, we focus on packet filters. Packet filters inspect packets, and decide whether to drop or forward a packet based on firewall rules. Packet filters are usually stateless; they filter each packet based only on the information contained in that packet, without paying attention to whether a packet is part of an existing stream of traffic. Packet filters often use a combination of a packet's source and destination address, its protocol, and, for TCP and UDP traffic, port numbers. In this lab, students will play with this type of firewall, and also through the implementation of some of the key functionalities, they can understand how firewalls work. Moreover, students will learn how to use SSH tunnels to bypass firewalls. This lab covers the following topics:

- Firewall
- Netfilter
- Loadable kernel module
- SSH tunnel

2 Lab Environment

This lab has been tested on the Ubuntu 16.04 VM that you should have installed on your computer during the first lab of this course.

3 Submission

Submit a PDF document with your answers to the tasks and questions in this lab. Include the task numbers and the questions. Place your answers immediately after the task number and question. Note that some answers require you to include a screen shot. Paste any code that you write or use next to the corresponding question. Some answers ask you to attach the source code to the Blackboard assignment. Use meaningful and easy to read names for your source code files.

4 Lab Tasks

4.1 Task 1: Using a Firewall

Linux has a tool called `iptables`, which is essentially a firewall. It has a nice front end program called `ufw`. In this task, the objective is to use `ufw` to set up some firewall policies, and observe the behaviors of your system after the policies become effective. You need to set up at least two VMs, one called Machine A, and other called Machine B. You run the firewall on your Machine A. Basically, we use `ufw` as a personal firewall. After you set up the two VMs, you should perform the following tasks:

- Prevent A from using `telnet` to connect Machine B.
- Prevent B from doing `telnet` to Machine A.
- Prevent A from visiting an external web site.

You can choose any web site that you like to block, but keep in mind, some web servers have multiple IP addresses. You can find the manual of `ufw` by typing "`man ufw` "or search it online. It is pretty straightforward to use. Please remember that the firewall is not enabled by default, so you should run a command to specifically enable it. We list some commonly used commands in at the end of this document.

Before starting the task, go to the default policy file `/etc/default/ufw`. Find the following entry, and change the rule from `DROP` to `ACCEPT`; otherwise, all the incoming traffic will be dropped by default.

```
# Set the default input policy to ACCEPT, DROP, or REJECT. Please note that if
# you change this you will most likely want to adjust your rules.
DEFAULT_INPUT_POLICY="DROP"
```

In addition to set up the firewall rules, you also need the following commands to manage the firewall:

```
$ sudo ufw enable          // this will enable the firewall.
$ sudo ufw disable         // this will disable the firewall.
$ sudo ufw status numbered // this will display the firewall rules.
$ sudo ufw delete 3        // this will delete the 3rd rule.
```

Deliverable. A screen shot of the output of the command `sudo ufw status` showing the correct rules are active. Make sure that the rules achieved the desired result. The rules we be implemented and tested for grading.

4.2 Task 2: Evading Egress Filtering

Many companies and schools enforce egress filtering, which blocks users inside of their networks from reaching out to certain web sites or Internet services. They do allow users to access other web sites. In many cases, this type of firewalls inspect the destination IP address and port number in the outgoing packets. If a packet matches the restrictions, it will be dropped. They usually do not conduct deep packet inspections (i.e., looking into the data part of packets) due to the performance reason. In this task, we show how such egress filtering can be bypassed using the tunnel mechanism. There are many ways to establish tunnels; in this task, we only focus on SSH tunnels.

You need two VMs A and B for this task. Machine A serves as both the user machine and the firewall. Machine B is outside of the firewall. Typically, there is a dedicated machine that runs the firewall, but in this task, for the sake of convenience, you run the firewall on Machine A. You can use use `ufw` as in the previous task. You need to set up the following two firewall rules:

1. Block all the outgoing traffic to external telnet servers. This is different from the previous tasks where you only blocked the telnet service to Machine B. In reality, the servers that are blocked are usually game servers or other types of servers that may affect the productivity of employees. In this task,we use the telnet server for demonstration purposes.

2. Block all the outgoing traffic to `www.facebook.com`, so employees (or school kids) are not distracted during their work/school hours. Social network sites are commonly blocked by companies and schools. After you set up the firewall rule, launch your Firefox browser, and try to connect to Facebook, and report what happens. If you have already visited Facebook before using this browser, you need to clear all the caches using Firefox's menu: `Edit -> Preferences -> Privacy& Security (left pane) -> Clear History (Button on right)` ; otherwise, the cached pages may be displayed. If everything is set up properly, you should not be able to see Facebook pages. It should be noted that Facebook has many IP addresses, it can change over the time. Remember to check whether the address is still the same by using ping or dig command.

Task 2.a: Telnet to Machine B through the firewall. To bypass the firewall, we can establish an SSH tunnel between Machine A and B, so all the telnet traffic will go through this tunnel (encrypted), evading the inspection. The following command establishes an SSH tunnel between the `localhost` (port 8000) and machine B (using the default port 22); when packets arrive to Machine B on port 22 they will be forwarded to Machine B's port 23 (telnet port). In practice, the machine running the telnet server would be different from Machine B, and Machine B would act as a proxy forwarding the telnet traffic to the third machine.

```
$ ssh -L 8000:Machine_B_IP:23  seed@Machine_B_IP
// We can now telnet to Machine B via the tunnel:
$ telnet localhost 8000
```

When we `telnet` to `localhost`'s port 8000, SSH will transfer all our TCP packets from one end of the tunnel on `localhost:8000` to the other end of the tunnel on Machine B; from there, the packets will be forwarded to port 23 on Machine B. Replies from port 23 will take a reverse path, and eventually reach our telnet client. Essentially, we are able to telnet to port 23 on Machine B.

Deliverable Explain what the "-L" option does in the ssh command. Explain how the tunneling approach allows us to bypass the egress filtering. A screen shot of the packet traffic provided by Wireshark on Machine A showing the packet traffic when connecting to the telnet server on machine B.

Task 2.b: Connect to Facebook using SSH Tunnel To achieve this goal, we can use the approach similar to that in Task 3.a, i.e., establishing a tunnel between your `localhost:port` and Machine B, and ask Machine B to forward packets to Facebook. To do that, you can use the following command to set up the tunnel:"`ssh -L 8000:FacebookIP:80 seed@MachineB_IP`".

However, we will not use this approach, and instead, we use a more generic approach, called dynamic port forwarding, instead of a static one like that in Task 3.a. To do that, we only specify the local port number, not the final destination. When Machine B receives a packet from the tunnel, it will dynamically decide where it should forward the packet to based on the destination information of the packet.

```
$ ssh -D 9000 -C seed@machine_B
```

Similar to the telnet program, which connects `localhost:9000`, we need to ask Firefox to connect to `localhost:9000` every time it needs to connect to a web server, so the traffic can go through our SSH tunnel. To achieve that, we can tell Firefox to use `localhost:9000` as its proxy. To support dynamic port forwarding, we need a special type of proxy called SOCKS proxy, which is supported by most browsers. To set up the proxy in Firefox, go to the menu bar, click Edit -¿ Preferences, scroll down to Network Proxy, and

click the Settings button. Choose Manual Proxy configuration, in SOCKS host add 127.0.0.1, Port 9000, and add "localhost, 127.0.0.1" to the field "No proxy for:". After the setup is done, do the following

- Run Firefox and go visit the Facebook page. Can you see the Facebook page? Please describe your observation.
- After you get the facebook page, break the SSH tunnel, clear the Firefox cache, and try the connection again. Please describe your observation.
- Establish the SSH tunnel again and connect to Facebook. Describe your observation.

Deliverable. Explain what the -D and -C options do in the ssh command. Explain how this approach helps you bypass the firewall rules. A screen shot of the packet traffic from Wireshark when there is a tunnel and when there is no tunnel.

4.3 Task 3: Evading 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 could 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.

Deliverable. A description that includes all the command line instructions that your run on both MAchine A and B to establish the reverse tunnel.