Drop Or Accept

Exercise 15

What You'll Learn in this Lab:

- Basic firewall concepts
- iptables commands
- The ufw (Uncomplicated Firewall) host-based firewall

Important Note!

When working with firewalls, it is possible to lock yourself out from remote access over the network. Type carefully.

Details

A firewall is a set of rules to allow or deny passage of network traffic. A firewall may also perform more complex tasks, such as network address translation, bandwidth adjustment, provide encrypted tunnels and much more related to network traffic. For this lab, we will focus on host-based firewalling.

Iptables is an administration tool for IPv4/IPv6 packet filtering and NAT (Network Address Translation). Iptables and ip6tables are used to set up, maintain, and inspect the tables of IPv4 and IPv6 packet filter rules in the Linux kernel. Several different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

Each chain is a list of rules which can match a set of packets. Each rule specifies what to do with a packet that matches. This is called a `target', which may be a jump to a user-defined chain in the same table. A firewall rule specifies criteria for a packet and a target. If the packet does not match, the next rule in the chain is examined.

There are five different types of iptables tables: Filter, NAT, Mangle, Raw, and Security. We will be looking at filtering in this lab.

Developed to ease iptables firewall configuration, ufw provides a user friendly way to create an IPv4 or IPv6 host-based firewall. By default UFW is disabled. When you turn UFW on, it uses a default set of rules (profile) that should be fine for the average home user. However, if you add services or want any inbound access, you most likely will need to add rules.

While you will be working with a partner, you will perform all of the steps in this section on each of your Raspberry Pi's.

Setup:
Log into your Raspberry Pi. You are going to update the firmware and install the firewall software:

```
sudo apt-get update
sudo rpi-update
sudo apt-get remove iptables
sudo apt-get install ufw
sudo reboot
```

Log back into your RPi and, examine all of the rules in all of the chains:

```
sudo iptables –L -v
```

You should see something similar to:

```
Chain INPUT (policy ACCEPT 788 packets, 127K bytes)
 pkts bytes target     prot opt in     out     source          destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source          destination

Chain OUTPUT (policy ACCEPT 12 packets, 1552 bytes)
 pkts bytes target     prot opt in     out     source          destination
```

By default, iptables allows all input, output and forwarding traffic (i.e., nothing will be blocked).

iptables
Go to your web browser and enter your RPi's IP address to make sure that you can see the default page, and the page on port 8080. Once the page is up, we will block any input going to port 80 by typing:

```
sudo iptables -A INPUT -p tcp --dport 80 -j DROP
```

Now click on the link to your **hello world** program. This should time out. Examine the rules again, and you should see an entry that drops the packets from anywhere that is coming in on port 80 (http). You should also see how many packets and bytes were dropped.

Let's remove this rule and replace it with one that rejects the input attempt:

```
sudo iptables -D INPUT -p tcp --dport 80 -j DROP
sudo iptables -A INPUT -p tcp --dport 80 -j REJECT
```

Try to access your web page again. It should time out again. What is the difference between the DROP and the REJECT? When might you use one or the other?

Our rules are currently not persistent, they are only in memory at this point. The –F option tells iptables to flush all of the rules from memory. The iptables-save and iptables-restore commands allow up to backup our rules and to restore them. Try the following to see how this works:

```
sudo iptables –L –v
sudo iptables-save > /root/fwbackup.171116
sudo iptables –F
sudo iptables –L –v
sudo iptables-restore < /root/fwbackup.171116
sudo iptables –L –v
```

Flush the reject rule from iptables. What command id you issue?

ufw

ufw acts as a front-end to iptables. By default, ufw drops all input and forward requests. To see this, look at the file /etc/default/ufw and grep out the lines with "DEFAULT". If we enabled ufw at this point, we would not be able to PuTTy (ssh) in over the network. We therefore want to start by adding an allow rule for ssh. By default, ssh runs on port 22 and uses TCP. The command for this is:

```
sudo ufw allow 22/tcp
```

However, since ssh is a common, well known, type of connection we do not need to specify the port or protocol, but instead will could enter:

```
sudo ufw allow ssh
```

It is a good idea to document all of your firewall rules with comments. As your rule list grows, it will help you to troubleshoot connections. Type the following:

```
sudo ufw allow ssh comment 'Enable inbound ssh'
```

We are now ready to enable ufw. Type:

```
sudo ufw enable
```

You can display the current ufw rules and status by typing:

```
sudo ufw status verbose
```

You should see that the status is active, that the default for incoming traffic is deny, and outgoing is allow. We see that we are allowing port 22/tcp (ssh) in from anywhere. The first entry is for IP version 4, and the second is for IP version 6.

The status is a summary of what is in place. Once ufw was enabled it automatically added a number of rules to iptables. To see everything that was put into place, type:

```
sudo ufw show raw | less
```

Mid-way you should see an entry for "Chain ufw-user-input" that has the accept for port 22.

Try to access your web page again. Why is it blocked? What ufw command would you issue to enable http access to your web page? Enter that command and make sure that you can get to your default web page.

If you have a process running on your system listening on a specific port, you will need to open that port on the firewall to provide access. Take a look at the ports that your system is listening for connections by typing:

```
netstat -tupln
```

Assuming that you have completed lab 10 (and if you haven't, then this might be a good time to do so), there should be one port number that you are listening for, but we have not yet enabled inbound access. What is that port number and protocol? What ufw command would you issue to enable access? Do not yet enter that command.

The general format to allow access from a specific address or range of addresses, to a specific port would be (the following is all on one line):

```
ufw allow from <IP ADDRESS|NETWORK/CIDR> to <DESTINATION|"any">
port <PORT NUMBER> proto <"tcp" | "udp"> comment 'YOUR COMMENT'
```

Using the port number and protocol from the prior question and the above template enter the ufw command to allow access from the 10.0.0.0/24 network. Write down the command. How would you test to make sure it is working correctly?

In addition to allowing access, ufw also gives us the ability to deny traffic with the deny option.

Let's assume that you were being attacked by IP addresses in the 163.153.100.0 – 163.153.100.255 range. What ufw rule could you use to block all traffic from those addresses? Add the rule.

To remove a rule, you would type **ufw delete** followed by the exact rule. To simplify life, you can list and then manage rules by rule number. Type:

```
sudo ufw status numbered
```

You should see a list similar to:

```
     To          Action    From
     --          ------    ----
[ 1] 22/tcp      ALLOW IN  Anywhere      # Enable inbound ssh
[ 2] 80/tcp      ALLOW IN  Anywhere      # Enable inbouund http
[ 3] 8080/tcp    ALLOW IN  10.0.0.0/8    # Enable inbouund http on port 8080
[ 4] Anywhere    DENY IN   163.153.100.0/24 # Blocked network attack on
11/15/17
[ 5] 22/tcp (v6) ALLOW IN  Anywhere (v6)  # Enable inbound ssh
[ 6] 80/tcp (v6) ALLOW IN  Anywhere (v6)  # Enable inbouund http
```

Remove the deny rule using its rule number. What command did you issue?

Some applications need to use a loopback address. Create and implement a single ufw rule that will allow any traffic from both 127.0.0.1 and from 127.0.1.1.

Print out and attach the results of a verbose ufw status.

Reflection

Answers to these questions must not be written on the lab worksheet, but typed on separate sheets of paper attached to the end of your lab worksheet or shared as a Google Doc. Your answers must be typed (not handwritten), and you will be graded on all aspects of your answer (correctness, use of proper terminology, readability, use of complete sentences only, etc.). In general you are expected to write at least one or two paragraphs in answer to each question.

When you installed iptables, by default it allowed all input, output and forwarding traffic (i.e., nothing was blocked). Yet, the default installation for ufw drops all input and forwarding traffic. What are the benefits and drawbacks to each of these approaches? If you were to set up your own system, would you want to block everything except for the traffic that you explicitly allowed (i.e., white-list), or would you want to allow everything except for the traffic that you explicitly block (i.e., black-list)?

This lab drew material from UFW- Community Help Wiki at https://help.ubuntu.com/community/UFW and from Uncomplicated Firewall (ufw) – Debian wiki at https://wiki.debian.org/Uncomplicated%20Firewall%20%28ufw%29