Firewall Lab using IPTABLES

PURPOSE
In this lab you will explore the Linux firewall iptables. The first part of the lab will describe iptables, then basic commands, followed by some exercises.

SOFTWARE REQUIREMENTS: Use of a Linux VM (KALI or Amazon)

REFERENCES:
1. https://help.ubuntu.com/community/IptablesHowTo

Part 1 – Background on iptables

Originally, the most popular firewall/NAT package running on Linux was ipchains, but it had a number of shortcomings. To rectify this, the Netfilter organization decided to create a new product called iptables, giving it such improvements as:

- Better integration with the Linux kernel with the capability of loading iptables-specific kernel modules designed for improved speed and reliability.
- Stateful packet inspection. This means that the firewall keeps track of each connection passing through it and in certain cases will view the contents of data flows in an attempt to anticipate the next action of certain protocols. This is an important feature in the support of active FTP and DNS, as well as many other network services.
• Filtering packets based on a MAC address and the values of the flags in the TCP header. This is helpful in preventing attacks using malformed packets and in restricting access from locally attached servers to other networks in spite of their IP addresses.
• System logging that provides the option of adjusting the level of detail of the reporting.
• Better network address translation.
• Support for transparent integration with such Web proxy programs as Squid.
• A rate limiting feature that helps iptables block some types of denial of service (DoS) attacks.

Considered a faster and more secure alternative to ipchains, iptables has become the default firewall package installed under Ubuntu, Debian, RedHat and Fedora Linux.

Part 2 – Basic Commands & Use

The executable binary for the iptables application is the /sbin/iptables file. So when execute iptables this is the file that is executed. RedHat also has an executable script in the /etc/init.d/iptables file so that it can start its default firewall at startup. An easy way to reset the firewall rules is to use this script as
/etc/rc.d/init.d/iptables stop

This flushes all the rules in memory and gives you a clean start.

1. You can flush individual rules with

iptables –F    flushes all chains
iptables –X <chain name> deletes a chain
iptables –X    deletes all chains
2. To check if there are any current rules, type

`iptables -L`

It'll show you a couple of chains, but there won’t be any rules in them.

You might have to do `/etc/init.d/iptables restart` to apply these changes (note: this is a different command than the one you used above).

3. You can start, stop, and restart iptables after booting by using the commands:

   > service iptables start
   > service iptables stop
   > service iptables restart

4. To get iptables configured to start at boot, use the `chkconfig` command:

   `chkconfig iptables on`

5. To get iptables configured to start at boot, use the `chkconfig` command:

   `service iptables status`
Part 3 – Exploring iptables

1. Start the VMware Linux guest O/S and log in.
2. Determine the IP address by using ifconfig.
3. Use an ssh client to connect to your guest O/S from your Host OS.
4. Enter the following command **ping** `www.cs.gmu.edu`
5. What do you see as output?
6. Now let’s block access to this site, enter the following command to list all the firewall rules: **iptables –L**
7. What do you see as output?
8. Let’s try to block the ping requests using the following command:
   ```bash
   iptables -A OUTPUT -s 0/0 -d 129.174.125.139 -j DENY
   ```
10. Use **iptables –L** again, what is the output?
11. Use **iptables –F** to flush (delete all the rules)
12. Use **iptables –L** to verify that you deleted the rules
13. Let’s try to block the ping requests using the following command:
   ```bash
   iptables -A INPUT -s 129.174.125.139 -j DROP
   ```
15. Use **iptables –L** again, what is the output?
16. What is the difference between the command in step 8 and the command in step 13? Do they produce the same output? Do they have the same effect?
Part 4 – Questions

1) What is the command to block the following subnet: 192.168.2.x?

2) Describe what the following rules do:
   a. `iptables -A INPUT -i lo -p all -j ACCEPT`
   b. `iptables -A INPUT -p all -s localhost -i eth0 -j DROP`
   c. `iptables -A INPUT -s 0/0 -i eth0 -d 192.168.1.1 -p TCP -j ACCEPT`
   d. `iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP --sport 1024:65535 --dport 80 -j ACCEPT`

3) Create a rule that will restrict SSH to allow only specific IP addresses
4) Create rules that will block any HTTP and HTTPS browsing from the virtual machine
5) Create a rules to rate-limit ICMP(Ping) traffic to 5 packets per second
6) Allow only incoming SSH connections
7) Block all incoming TCP traffic
8) Block all incoming UDP traffic

9) Assume you are running a web server that you would like to secure with iptables. Discuss in detail how you would go about doing that.
**Extra Credit**

a) A SYN flood is a form of denial-of-service attack in which an attacker sends a succession of SYN requests to a target's system. This is a well known type of attack and is generally not effective against modern networks. It works if a server allocates resources after receiving a SYN, but before it has received the ACK. If Half-open connections bind resources on the server, it may be possible to take up all these resources by flooding the server with SYN messages. Syn flood is common attack and it can be blocked with iptable rules.

Can you craft iptable rules that can block SYN flooding attacks? Explain your work and rationale.

b) Port knocking is a stealth method to externally open ports that, by default, the firewall keeps closed. It works by requiring connection attempts to a series of predefined closed ports. When the correct sequence of port "knocks" (connection attempts) is received, the firewall opens certain port(s) to allow a connection.

The benefit is that, for a regular port scan, it may appear as the service of the port is just not available. This article shows how to use port knocking with either a daemon or with iptables only.

Can you create a port knocking system only with iptable commands that will open up port 22 (SSH) when the remote host “knocks” on ports 2222, 3333, 4444 within 30 seconds?

c) Can an attacker defeat the port knocking system? How?