Lab #3

Class: CECS 303 – Networks and Network Security
Instructor: Chris Samayoa
Due Date: February 21, 2022 by 9pm PST

Objective: Understand and implement basic iptables commands in order to understand the functionality of basic firewall rules.

Legend:

- *Server*: refers to the two Ubuntu Server VMs that were created in Lab 1
    - *Apache Server*: refers to the server VM with Apache installed on it
- *Workstation*: refers to the Ubuntu Desktop VM that was created in Lab 2

References:

- iptables manual: https://ipset.netfilter.org/iptables.man.html
    - Current version can be checked on linux terminal using the 'man iptables' command
- Basic iptables commands: https://help.ubuntu.com/community/IptablesHowTo

Video:

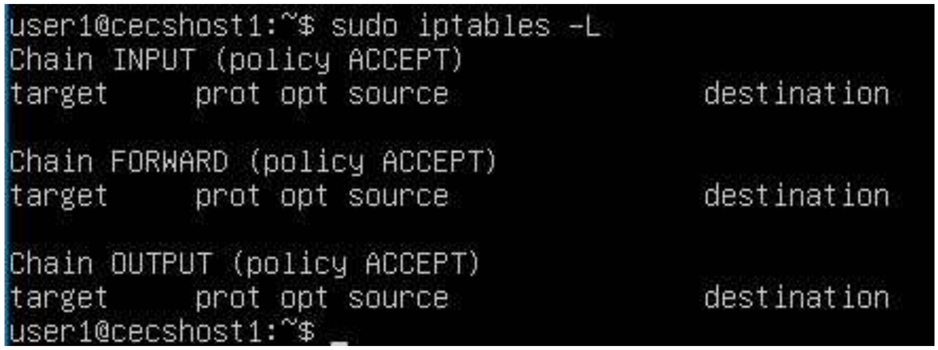
https://www.youtube.com/watch?v=6Ra17Qpj68c

Watch this video in order to view a more in depth explanation of the iptables filters that you will be working with in this lab.

Basic iptables setup

In order to have a functioning baseline for your two server VMs, run the following commands:

1. View current iptables filter rules (should be empty to begin)
    a. 'sudo iptables -L': basic command for listing all current rules in iptables

i. 

ii. Note that the default for each chain type is to ACCEPT all traffic. This means that no traffic is being blocked.

b. ‘sudo iptables –L --line-numbers’: lists all current rules along with their assigned priority number

c. ‘sudo iptables –L –v’: shows all current rules in a verbose manner

   i. Shows how many network packets have matched the rule and interface(s) the rules applies to

d. ‘sudo iptables -L -n –v’: all of 1c, but the source and destination IP addresses are shown numerically and the port rules are shown as numbers instead of service names

2. Make iptables rules persistent on reboot – by default all commands entered will be forgotten upon system reboot

   a. ‘sudo apt-get install iptables-persistent’: command installs utility to make iptables commands persistent on reboot

      i. Accept default options during installation

   b. ‘sudo netfilter-persistent save’: IMPORTANT – use this command any time that you need to save the rules you have added. Without this command, your modifications to iptables will disappear upon a reboot

3. Enter baseline firewall rules

   a. Accept all traffic on your loopback interface. This ensures that you can connect to other services running on the same VM and that you can test connections locally using the localhost

      i. ‘sudo iptables -A INPUT -i lo -j ACCEPT’: allows all incoming traffic to loopback interface

      ii. ‘sudo iptables -A OUTPUT -o lo -j ACCEPT’: allows all outbound traffic to loopback interface

b. Allow all established and related incoming connections. Network traffic generally needs two-way traffic to function. This rule ensures that the server allows return traffic for outgoing connections initiated within the VM
    i. 'sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT'
c. Allow established outgoing connections. This rule ensures that the server allows return traffic for established incoming connections
    i. 'sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT'
d. At this point it is safe to change the default to DROP for the INPUT and OUTPUT chains. This means that all traffic will be blocked unless it is specifically allowed by an iptables rule. Note that after these rules are applied, you will only be able to access the VM from the console (through VirtualBox)
    i. 'sudo iptables -P INPUT DROP'
    ii. 'sudo iptables -P OUTPUT DROP'
    iii. Verify that the defaults have now been changed to DROP for the INPUT and OUTPUT chains using the 'sudo iptables –L -v' command

1. 

    a. *Take screenshot for deliverable (need one from each of your server VMs)*

At this point, all traffic not coming to/from the loopback network interface or that was not already established before the rules were put in place will be blocked off from this VM. I would recommend saving the configuration at this point if you have not already ('sudo netfilter-persistent save').

Add iptables commands to allow ICMP and DNS queries:

With the default now set to DROP all traffic not specifically allowed, we need to allow ICMP traffic in and out of each VM for ping commands to work.

Additionally, you will not be able to ping an external domain (e.g. www.google.com) until DNS queries are specifically allowed:

1. Allow inbound and outbound ICMP connections (e.g. ping commands)
    a. First, try to ping each server VM from the other and an external IP address such as '4.2.2.2'. Since ICMP traffic has not been allowed, the commands will fail
        i.
```
user2@cecshost2:~$ ping -c 4 -4 192.168.4.46
PING 192.168.4.46 (192.168.4.46) 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
^C
--- 192.168.4.46 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3068ms

user2@cecshost2:~$ ping -c 4 -4 4.2.2.2
PING 4.2.2.2 (4.2.2.2) 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
^C
--- 4.2.2.2 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3068ms
```
    b. Run the following commands on both server VMs to allow ICMP traffic:
        i. 'sudo iptables -A OUTPUT -p icmp -m conntrack --ctstate NEW,RELATED -j ACCEPT'
        ii. 'sudo iptables -A INPUT -p icmp -m conntrack --ctstate NEW -j ACCEPT'
    c. Now try the ping commands listed above again
        i.
```
user2@cecshost2:~$ ping -c 4 -4 192.168.4.46
PING 192.168.4.46 (192.168.4.46) 56(84) bytes of data.
64 bytes from 192.168.4.46: icmp_seq=1 ttl=64 time=0.353 ms
64 bytes from 192.168.4.46: icmp_seq=2 ttl=64 time=0.962 ms
64 bytes from 192.168.4.46: icmp_seq=3 ttl=64 time=0.911 ms
64 bytes from 192.168.4.46: icmp_seq=4 ttl=64 time=0.893 ms

--- 192.168.4.46 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3020ms
rtt min/avg/max/mdev = 0.353/0.779/0.962/0.247 ms
user2@cecshost2:~$ ping -c 4 -4 4.2.2.2
PING 4.2.2.2 (4.2.2.2) 56(84) bytes of data.
64 bytes from 4.2.2.2: icmp_seq=1 ttl=56 time=10.7 ms
64 bytes from 4.2.2.2: icmp_seq=2 ttl=56 time=11.8 ms
64 bytes from 4.2.2.2: icmp_seq=3 ttl=56 time=11.6 ms
64 bytes from 4.2.2.2: icmp_seq=4 ttl=56 time=11.2 ms

--- 4.2.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 10.698/11.337/11.824/0.428 ms
```

1. They should now be successful

d. How about pinging www.google.com?

i.
```
user2@cecshost2:~$ ping -c 4 -4 www.google.com
ping: www.google.com: Temporary failure in name resolution
user2@cecshost2:~$
```

ii. It fails because the name cannot be resolved

e. Run the following commands on both server VMs to allow for DNS queries to be made:

i. 'sudo iptables -A OUTPUT -p udp --dport 53 -m conntrack --ctstate NEW -j ACCEPT'

ii. 'sudo iptables -A OUTPUT -p tcp --dport 53 -m conntrack --ctstate NEW -j ACCEPT'

iii. Now test pinging a domain name again

1.
```
user2@cecshost2:~$ ping -c 4 -4 www.google.com
PING www.google.com (142.250.68.36) 56(84) bytes of data.
64 bytes from lax17s46-in-f4.1e100.net (142.250.68.36): icmp_seq=1 ttl=116 time=8.98 ms
64 bytes from lax17s46-in-f4.1e100.net (142.250.68.36): icmp_seq=2 ttl=116 time=8.25 ms
64 bytes from lax17s46-in-f4.1e100.net (142.250.68.36): icmp_seq=3 ttl=116 time=8.99 ms
64 bytes from lax17s46-in-f4.1e100.net (142.250.68.36): icmp_seq=4 ttl=116 time=9.32 ms

--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 8.250/8.885/9.322/0.391 ms
```

2. Similarly, the 'dig' commands previously learned should now work again from both servers

You can now test connectivity to other machines and to the internet using ICMP again and can resolve domain names to their respective IP addresses.

Turn on logging

The ability to audit failed connections is essential for both troubleshooting network connectivity issues and detecting attempted attacks

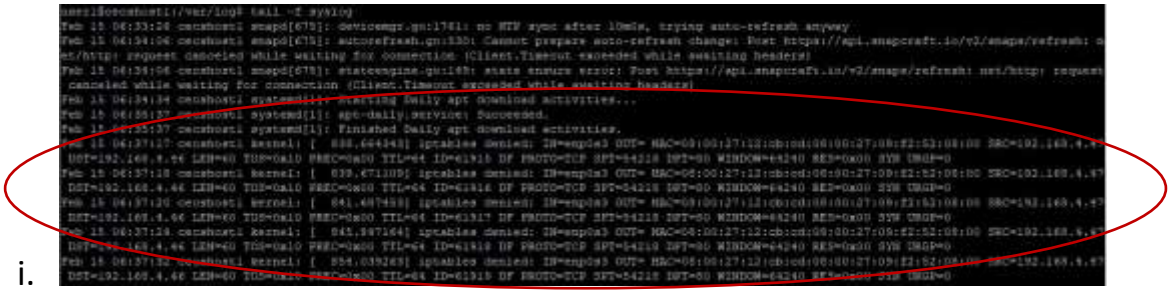
1. Complete Deliverable 1a below. STOP THERE, DO NOT COMPLETE 1b UNTIL AFTER THIS SECTION IS COMPLETE.
2. Log blocked traffic
   a. 'sudo iptables -A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables denied: " --log-level 7'
      i. Run this command on both server VMs
3. Test that denied traffic is now being logged
   a. On a terminal for the Apache Server (web server), run the following command to follow the syslog (where iptables logs are stored by default):

i. ‘tail –f /var/log/syslog’

ii. 

1. By default, this command shows the previous 10 entries in the syslog file and then monitors for new entries
2. Leave this running and move on to the next step

b. From a machine that is NOT the Apache Server, attempt to telnet to the Apache Server on port 80 (‘telnet <ip address> 80’)
   i. The telnet attempt should fail
c. Switch back to your Apache Server running the ‘tail –f’ command and you should see the failed connection attempt:

i. 

ii. *Take screenshot for deliverable*

Deliverables (submit via BeachBoard)

1. Using what you’ve learned from this lab along with the references and video above, create additional iptables rules for the following:
   a. On both server VMs, create a rule to allow outbound HTTP and HTTPS traffic
      i. The goal here is to allow both servers to browse the internet using these protocols
      ii. You can test for the success of your command by using telnet to test connectivity to a known domain (e.g. ‘telnet google.com 80’). If the command is successful, then a connection should be established.

   b. On the Apache Server VM, create a rule(s) to allow inbound HTTP and HTTPS traffic from anywhere inbound to the server
      i. Once this rule is created, then you should be able to test a telnet connection to your Apache Server successfully (e.g. 'telnet <Apache Server IP Address> 80')
   c. After confirming that your above two modifications work properly, run the 'sudo iptables –L –v' command on both server VMs and take a screenshot of each output for your submittal. The screenshots should include the modifications you made.
2. Suppose you only wanted to allow HTTP connectivity to your Apache Server from one other machine on your network (e.g. your laptop or one of the other VMs in Virtual Box). What iptables command would you use?
3. Compile the three screenshots requested throughout the document in a single .doc, .docx, or .pdf file along with the answer to question #2.

Note: Command "shutdown now" will cleanly shut down virtual machines when you are done working with them