Matt Favazza

Firewalls CentOS

There are more indepth explanations in the ‘Firewalls Ubunutu’ document. This document glosses over finer details in favor of efficiency.

To see the IPTABLE of the machine type **sudo iptables -L**

This is a sample of some lines of the IPTABLE for the CentOS Server. See iptablesCentOS.txt

Chain INPUT (policy ACCEPT)

target prot opt source destination

ACCEPT all -- anywhere anywhere ctstate RELATED,ESTABLISHED

ACCEPT all -- anywhere anywhere

INPUT\_direct all -- anywhere anywhere

INPUT\_ZONES\_SOURCE all -- anywhere anywhere

INPUT\_ZONES all -- anywhere anywhere

DROP all -- anywhere anywhere ctstate INVALID

REJECT all -- anywhere anywhere reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)

target prot opt source destination

ACCEPT all -- anywhere anywhere ctstate RELATED,ESTABLISHED

ACCEPT all -- anywhere anywhere

FORWARD\_direct all -- anywhere anywhere

FORWARD\_IN\_ZONES\_SOURCE all -- anywhere anywhere

FORWARD\_IN\_ZONES all -- anywhere anywhere

FORWARD\_OUT\_ZONES\_SOURCE all -- anywhere anywhere

FORWARD\_OUT\_ZONES all -- anywhere anywhere

DROP all -- anywhere anywhere ctstate INVALID

REJECT all -- anywhere anywhere reject-with icmp-host-prohibited

Unlike Ubuntu, which has ufw, we can only use IPTABLE commands. So for this, I’m writing one script to do everything necessary. First we **touch FIREWALL** to create the file for the script and then we can use an editor like nano to modify it.

Graphical user interface, text, application

Description automatically generated

1. Deal with webserver ports

For webserver ports we need port 80 and 443, but we also need to port forward port 80 to 8080, so 8080 is added to the list of ports we need to open.

Unlike ufw, we need to specify what protocol is being used too, so for web ports we need the tcp and udp protocols enabled, which doubles the lines we need to write.

**Sudo iptables -A INPUT -p tcp [portnum] -j ACCEPT** and **Sudo iptables -A INPUT -p udp [portnum] -j ACCEPT** needs to be done for each of the three ports to be opened.

Graphical user interface, text

Description automatically generated with medium confidence

For the port forwarding part we need to know our machines IP address, which can be found using **ifconfig** then we can add these lines to the script

**iptables -t nat -A PREROUTING -p tcp --dport 22 -j DNAT --to-destination 192.168.139.137:8080**

**iptables -t nat -A PREROUTING -p udp --dport 22 -j DNAT --to-destination 192.168.139.137:8080**

Text

Description automatically generated

1. Deal with MySQL

For this we just need to open ports 3306 and 33060 again with tcp and udp using the **Sudo iptables -A INPUT -p** commands.

Text

Description automatically generated with medium confidence

1. Deal with SSH

For this machine I want to always allow ssh, but I will add an **#comment line** in the close ports if I decide I want it closed. For SSH we need port 22 with tcp and udp. Again we use the the **Sudo iptables -A INPUT -p** commands.

Text

Description automatically generated

In a second script we will run the same commands but with **DENY** instead of **ACCEPT**

Text

Description automatically generated

1. Deal with Email services

For this there are the three protocols, SMTP, POP3, and IMAP using ports 587, 995, and 993. Ive chosen these because these are secured version of the protocol, meaning data that is sent to and from the ports are encrypted. Again we need to have both tcp and udp, so in total 6 lines will be added to the script, and a second script will be made to deny these too.

Text

Description automatically generated

Text

Description automatically generated

1. Blocking IPs/MACs

Like on the Ubuntu server, I am choosing to block the address 192.168.139.250 and the MAC 00:00:00:00:00:AA. The MAC is just a placeholder unless I actually need to block a MAC address.

To block Ips we need to add **sudo iptables -I INPUT -s xxx.xxx.xxx.xxx -j DROP**

and to block MACs we need to add **iptables -A INPUT -m mac --mac-source 00:00:00:00:00:AA -j DROP**

Text

Description automatically generated

and in a second script, I will have the same commands but with **ALLOW** instead of **DROP** so we can toggle them on and off.

Text

Description automatically generated

1. Block telnet and pings

telnet runs off of port 23, and we need to block both tcp and udp. Pings (ICMP) runs off of port 5813 and again used tcp and udp.

So I will use **Sudo iptables -A INPUT** commands to block them in this script, then in a second script allow them

Text

Description automatically generated

Text

Description automatically generated

Finally we need to make sure our changes are permanent, we can add **sudo iptables-save** to the script.

Text

Description automatically generated

This script needs to be ran as ROOT. To elevate, type **sudo -i** then **whoami** to check if you are root.

Text

Description automatically generated

Again for testing purposes, the everything will be denied except webserver and MySQL ports. The way I wrote the scripts, Im going to run the main script first, then all the secondary scripts to block everything. The testing is done in ‘Firewalls Testing’ document. There is also the **iptables -L** output in the iptablesCentOSAfter.txt which was generated after the scripts where ran.

Resources that helped me

<https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?search=>

<https://wiki.centos.org/HowTos/Network/IPTables>

<https://www.adamintech.com/how-to-use-iptables-for-port-forwarding/>

<https://docs.rackspace.com/support/how-to/block-an-ip-address-on-a-Linux-server/>

<https://www.cyberciti.biz/tips/iptables-mac-address-filtering.html>

<https://www.cyberciti.biz/faq/iptables-block-port/>

<https://www.thomas-krenn.com/en/wiki/Saving_Iptables_Firewall_Rules_Permanently>