**CentOS Firewall**

What is a Firewall?

A firewall is a security tool on computer systems that can control all incoming and outgoing traffic. The firewall is the first line of defense when it comes to cyber security. It is important to only allow necessary and secure ports when it comes to inbound traffic. Some network engineers utilize a DMZ (demilitarized zone) as another security measure. This redirects inbound traffic towards a less critical part of the corporate network.

**Web Server**

Includes all web/internet traffic between the server and its visitors. This helps control all the incoming traffic to the server on port 80 and redirect it to port 8080. Before we begin creating the rules for web server, we need to confirm that the firewall is enabled and ready to configure.

1. **Command – “sudo systemctl status firewalld” & “sudo**

A screenshot of a computer

Description automatically generated

* First, it is always good practice to check the firewall status and confirm it is running, confirm the firewall is enabled before making any changes. If it is not enabled, run the command **sudo systemctl enable firewalld**. Then move on to the next step. Note that firewalld is the default configuration method for the firewall on CentOS, but iptables can still be used to do some more advanced settings.

1. **Command – “sudo iptables -L | less” & “sudo firewall-cmd –list-all”.**

A screenshot of a computer program

Description automatically generated Firewalld rules.

A screenshot of a computer screen

Description automatically generated

IP Tables.

* Make note of the firewall rules / ip tables BEFORE we begin making changes. It is also a good idea to take a snapshot of the virtual machine in case we need to revert to the old settings. Run the command and redirect the output to a file named with today’s date (**ex:** **sudo iptables -L >> 13NOV24iptables.txt**) for admin records.

1. **Command – “sudo iptables -A INPUT -p tcp –dport 80/8080 -j ACCEPT” & “sudo iptables -A PREROUTING -p tcp –dport 80 -j REDIRECT –to-port 8080”**



* To redirect web traffic from port 80 to port 8080, we first need to open the ports for inbound traffic. **Iptables -A INPUT** states to append (**-A**) the inbound rule (**INPUT**) to the ip table. Then specify the TCP protocol (**-p TCP**) and the ports we wish to open (80 and 8080) and accept the inbound traffic to these ports (**-j ACCEPT**).

1. **Command – “sudo iptables -A PREROUTING -p tcp –dport 80 -j REDIRECT –to-port 8080” & “sudo iptables -t nat -L -n | less”.**



* The next command is how we redirect traffic to port 8080 from port 80. Port 80 is the default port for web (http) traffic and port 8080 is an alternate port for web traffic. To check if the rules are now applying, run command **sudo iptables -t nat -L -n | less** to get the results of all existing rules in the IP tables.

A black screen with white text

Description automatically generated

* This is a useful command to confirm port forwarding rules, like the one we just created. Option **-t nat** specifies the NAT Table, which is the resource for all Network Address Translation (NAT) rules. **-L** option lists all rules in the firewall, and **-n** is used to disable DNS because we do not need to see hostnames in this scenario.

A screenshot of a computer screen

Description automatically generated

Launch wireshark in the terminal (command **tshark**), and open a browser from your host computer. Enter the IP Address of the server, mine is 192.168.40.94, and confirm the web traffic shows in wireshark.

A screenshot of a computer

Description automatically generatedIf configured correctly, you will be able to view the default web page.

**MySQL**

MySQL is a database management system that is open source. To connect and make it functional on the CentOS server, port 3306 will need to be opened.

1. **Command – “sudo iptables -A INPUT -p tcp –dport 3306 -j ACCEPT”**



* This command is used to open port 3306, at this point we should save the IP Tables to ensure they hold after a server reboot. Run **sudo iptables-save** to save the IP Table settings. The output of this command will show the rules we just created at the end of the list (*see below*). This command could also be redirected to a text file for system admin records, the date and time is appended to the command when it is run.

A screenshot of a computer

Description automatically generated

Note the rules we just created are appended towards the end of the command output.

**SSH**

SSH, an acronym for Secure Shell, is a network tool that allows a more secure remote connection between two computers. SSH uses encryption as part of the ‘handshake’ between the two computers communicating with each other. We will need to configure the SSH service to allow both incoming and outgoing traffic.

1. **Command – “sudo systemctl status sshd”** A screenshot of a computer

   Description automatically generated

* First, confirm the SSH service is running and enabled.

1. **Command - “sudo iptables -A INPUT -p tcp –dport 22 -j ACCEPT” & “sudo iptables -A OUTPUT -p tcp --dport 22 -j ACCEPT”**

A black screen with white text

Description automatically generated

* This adds / appends (**-A**) the following rule to the IP Tables. Use the two commands to allow incoming traffic (**INPUT**) and outgoing traffic (**OUTPUT**) to port 22 for the SSH service. Note if we wanted to disable SSH access, we would change the **ACCEPT** option at the end of the command to **DROP**. Example - **“sudo iptables -A INPUT -p tcp –dport 22 -j DROP”**

1. **Test SSH**

A screenshot of a computer program

Description automatically generated

* Test SSH using PuTTY. Enter the IP Address of the CentOS server, using port 22, and click ‘Open’ at the bottom right of the window.

A screenshot of a computer

Description automatically generated

* If you are prompted to login on the terminal, the connection is successful.

1. **Command – “sudo iptables -I INPUT -p tcp --dport 22 -j DROP”**

A screen shot of a computer

Description automatically generated

* To disallow incoming SSH traffic, use this command to tell the system to drop all packets coming into port 22. This will restrict the server from any incoming SSH access, which is a security risk for unauthorized access. Reload the firewall (**sudo firewall-cmd –reload)** and attempt to login to the server with PuTTY.

A computer error message

Description automatically generated

Again, using the IP Address of the server and port 22, select ‘Open’. If the session fails to connect, it confirms the firewall rule is working as intended. The screenshot here shows the error received when attempting to SSH into the server. Now due to the firewall rule set in place, all remote connections are disallowed.

**Allow or Block Specific Hosts / MAC Addresses**

1. **Command – “sudo iptables -A INPUT -s *IPaddress* -j DROP”**



1. To block a specific IP Address, we need to utilize a command like what we just used to block SSH traffic. **Iptables -A** **INPUT** tells the system to append (**-A**) a new rule to the iptables incoming traffic (**INPUT**). The **-s** option specifies an address to block incoming traffic from (**-j DROP**). I used my Ubuntu server IP Address as an example. To allow traffic from the address, simply use ‘**ACCEPT**’ at the end of the command instead of ‘**DROP’**.
2. **Command – “sudo iptables -A INPUT -m mac –mac-source *MACaddress* -j DROP”**



1. Specifying a MAC Address to block is a little more complicated than blocking the IP Address. The start of the command is similar, but we use (**-m**) telling the server to match a MAC Address, and (**--mac-source *MACaddress***) to state the specific address to block. Just like before, **-j DROP** finishes the rule to drop all packets from this source. Alternatively, we can use **ACCEPT** at the end to allow the incoming traffic from the MAC Address.

The screenshot below shows the rules we have put into place with the firewall. To see this, run the command **sudo iptables -L | less** to view all the incoming and outgoing firewall rules. I usually pipe **| less** to have the option to arrow up and down through the output of this command.

A screen shot of a computer

Description automatically generated

**Block Telnet and Ping**

1. **Command – “sudo iptables -A INPUT -p –dport 23 -j DROP”**



* Telnet is a network tool that allows remote connections between computers on the same network. By default, Telnet uses port 23, so to block Telnet we will need to drop all incoming traffic to port 23. Just like before with SSH, use the command **Iptables -A** **INPUT** telling the system to append (**-A**) a new rule to the iptables incoming traffic (**INPUT**). Then specify TCP protocol and port 23 **(-p tcp --dport 23**) and use the option to drop the connections (**-j** DROP). To allow Telnet, simply use ‘**ACCEPT**’ at the end of the command instead of ‘**DROP’**.

A screenshot of a computer

Description automatically generated

* To test Telnet, open PuTTY again and connect to the server using telnet and port 23.

A screenshot of a computer error message

Description automatically generated

* The connection should be refused because of the firewall rule.

1. **Command – “sudo firewall-cmd –permanent –add-icmp-block=echo-request”**

A black screen with white text

Description automatically generated

* Ping is a network tool that is normally used to confirm connections to other computers. It works by using Internet Control Message Protocol (ICMP), which returns a reply if the device can be reached with ping. By default, ping is enabled on most devices; it can be a strategic security measure to have ping disabled in the event of a security breach. I had to use firewalld on CentOS rather than IP Tables to make this change. This command is telling firewalld to add a rule to block all ping requests.

A screen shot of a computer

Description automatically generated

* Reload the firewall and try to ping the server from the Ubuntu server. If the ping fails like this example, it means the rule is working as intended.

**Sources**

<https://www.cisco.com/site/us/en/learn/topics/security/what-is-a-firewall.html>

<https://www.aholdengouveia.name/LinuxAdmin/iptables.html>

<https://santoshk.dev/posts/2023/iptables-explained-an-introduction-to-linux-firewalling/>

<https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-using-firewalld-on-centos-7>

<https://serverfault.com/questions/616435/centos-7-firewall-configuration>

<https://upcloud.com/resources/tutorials/configure-iptables-centos>

<https://docs.rackspace.com/docs/use-iptables-with-centos-7>