2015

Hands On Labs

Protecting Your Servers With Firewalld (CentOS 7)

Table of Contents

Introduction

Firewalld is a new dynamic firewall provided with CentOS 7 On our lab systems is it not installed by default and will be part of the process. Firewalld is used to interact with the powerful tool inside the Linux kernel called Netfilter. Netfilter allows the kernel to inspect every packet throughout the system. Because of Netfilter the system has the ability to read, stop, drop, modify, or reject the packages and in this lab the tool we are going to use to do that is Firewalld. Firewalld interface is a replacement for iptables and provides support for IPv4 and IPv6. Firewalld uses the iptables tool to talk to the kernel packet filter (netfilter). However, Firewalld is different than the iptables configuration interface. Firewalld allows for the setting of dynamic rules, which can be implemented in the current runtime without reloading the Firewalld system. Iptables never acted as a true Linux service. Firewalld does and interacts perfectly with systemctl. However, it's important to note, now and a few times later in this document, **never use systemctl to restart Firewalld to implement permanent rule changes;** always use the --reload option provided with firewall-cmd command.

Network Zones

Zones are used to create different levels of trust. We might trust traffic coming from the internal or private zone more than we trust traffic coming from the public zone. We can use Firewalld to configure high level rules or, more specifically, rules for ports. We can assign the rules to a zone. For example, we might assign port 80 for the apache webserver to the public zone if it's a web server. That way the web server can serve http data to any packet requesting it.

Zones -- Zone XML settings are located in the /etc/firewalld/zones directory

- Drop -- All incoming traffic is dropped with no reply, but outgoing traffic is allowed
- Block -- Only network connections initiated from within the system are possible. All incoming traffic is dropped with a reply message of "icmp-host-prohibited message"
- Public -- Allow traffic from any location and only specific connections are accepted based off of firewall rules
- External -- Use on external networks with masquerading
- Dmz -- For servers/computer in the your demilitarized zone that are publicly accessible
- Work -- use for work areas, there is a trust for other computers on this network not to inflict damage on the machine
- Home -- Use for home areas and trust other machines on the network not to harm or damage the machine; home rejects all incoming traffic unless it is in response to or related to outgoing traffic

- Internal -- use on internal networks and trust other machines on the network not to harm your machine
- Trusted -- all network connections are accepted

Packages, Resources and Prerequisites

On your lab page you will see two severs and public ip addresses listed for connectivity. Follow the steps below in order to setup the conventions to follow this lab guide.

1. Connect to both servers using the linuxacademy username and 123456 password. The linuxacademy user does have sudo privileges but you can also su -- into the root user with a password of 123456. Even thought this is just a lab ,it is suggested you change your passwords for good practice, however, it is not a requirement. You will need to have two terminals open or be using tabs to navigate between the two terminal windows for each of our servers.
2. Throughout this lab we will refer to “server1” and “server2” as they are labeled on the Linux Academy lab page. You can choose to set the hostname on each server to make it easier to follow the lab. Simple execute hostnamectl set-hostname server1 and hostnamectl set-hostname server2 on each server respectively.
3. On both Server1 and Server2, change to the root user and install Firewalld.
    a. **sudo su --**
    b. **yum install --y firewalld**
    c. **yum install --y httpd**
    d. **systemctl enable httpd; systemctl start httpd**
    e. **echo “hello world’ >> /var/www/html/index.html**
        i. We will be using the apache server to demonstrate firewall rules

Getting Started

If we are using Firewalld, we need to make sure that we mask iptables and ip6tables.

On the Server1 lab server:

1. **systemctl mask iptables**
2. **systemctl mask ip6tables**
3. Ensure the Firewalld service is running **systemctl status firewalld**
4. If the service is not running, then start it **systemctl start firewalld**

Remember: when applying permanent changes to Firewalld, DO NOT apply changes with systemctl reload or restart of Firewalld. This does not always ensure the rules are changed to the configuration.

Now lets take a few minutes to practice some administrative and navigation functions with firewall-cmd

On Server1

1. **firewall-cmd --reload**
2. **firewall-cmd --get-zones**
3. **firewall-cmd --list-all-zones**
4. **firewall-cmd --list-all**
5. **firewall-cmd --get-default-zone**

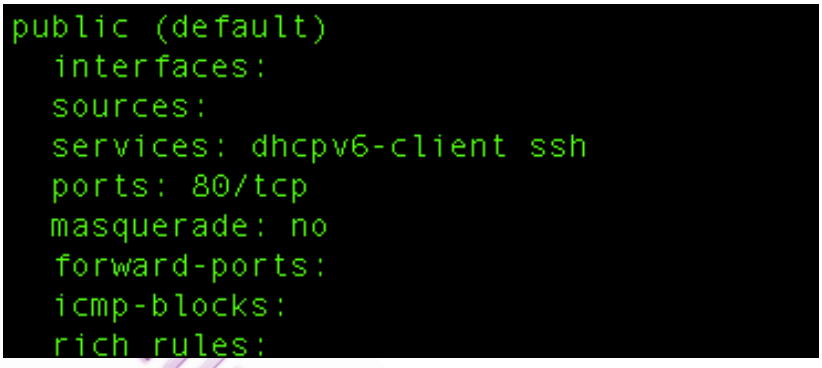
Take a few moments and explore the results of these commands; just be familiar with the results and common navigation commands with the firewall-cmd command.

Creating Httpd Runtime Firewalld Rules

In this scenario, we are going to use Server1 server to act as the httpd server. We will go through several exercises for allowing and denying access. We will be using Server2 lab server for this lab as well as your desktop machine to test the results.

1. Get the [Public IP] for server1 back on the Linux Academy lab page. This is the same server with the hostname of server1. With the IP address, open a browser and place the IP in the address bar and hit go/enter/return. This will attempt to load the page. However, since we have Firewalld installed, the page will not load.
2. Since we are attempting to access the server from a public computer (your computer is not part of the lab network so it is considered public) we need to add a rule to Firewalld configuration to allow incoming and outgoing traffic to the server httpd.
3. Let's query our firewall services and see if one exists already for httpd.
    a. **Firewall-cmd --list-services**
4. No httpd service is listed so lets add a rule to the public zone by specifying the protocol and port
    a. **Firewall-cmd --zone=public --add-port=80/tcp**
    b. If no zone is specified with --zone then the default zone from firewall-cmd --get-default-zone will be used. Even though our default was public we are going to specify it for learning purposes and practice.
    c. **Firewall-cmd --list-all**

i. You will not see ports: 80/tcp as part of this result. This lists all the current rules for the firewall.
ii. **Important:** the --list-all parameter will list all sources/services/interfaces etc for the default zone. In order to view information about another zone you must pass the --zone parameter.
    1. **firewall-cmd --zone=internal --list-all**
iii.



5. Now attempt to open the ip address again in the browser. You should see our Hello World text we created during the lab setup process.

Important: We did not specify the --permanent parameter when we created the rule. This means we only made a "run time" change. A run time change allows us to modify firewall rules WITHOUT having to reload the firewall. However, the changes we make are not persistent. If the server is rebooted, service restarted, or firewall—cmd --reload has been executed then our rule goes away. Lets demonstrate this in the next exercise.

1. **Firewall-cmd --reload**
2. Now attempt to refresh the browser window with linuxacdemy1 (server1)'s public ip address. It will again fail because the change was only made for the current runtime and was not persistent. You can also execute **firewall-cmd --list-all** and verify that 80/tcp is no longer listed.

Creating Httpd Persistent Firewalld Rules

In most cases when we want to make a change to the firewall we want that change to be persistent. It needs to exist after --reload commands, reboots, shut downs, service restarts, so

on and so forth. So how do we make this change? We make the change by adding --permanent parameter to the command creating the firewall rule.

Important: When issuing runtime changes we do not have to reload the firewall rules using firewall-cmd --reload. However, after making changes with the --permanent parameter to create persistent rules in order for the rule to take affect we MUST reload the firewall rules using firewall-cmd --reload. Again, it's important to use this command and not systemctl to reload the firewall rules.

1. **firewall-cmd --permanent --zone=public --add-port=80/tcp**
2. Verify the rule exists **firewall-cmd --list-all**
    a. The rule in fact DOES NOT exist because we have not reloaded the firewall
3. **firewall-cmd --reload**
4. **firewall-cmd --list-all** and you will see port 80 added and accessible.
5. Lets learn how to remove a rule once added
    a. **firewall-cmd --remove-port=80/tcp**
    b. Same rules apply when adding the rule; we did not use --permanent so it applied only to runtime. After a reload of the service, the rule would be back. Lets make the change permanent.
    c. **firewall-cmd --permanent --remove-port=80/tcp**
    d. **firewall-cmd --reload**
    e. **firewall-cmd --list-all**

Modifying Firewalld Rules To Allow Only Internal Traffic

Our zones still need to be configured to allow incoming traffic from certain areas. For example, our public zone is already configured to let traffic come in from the outside world. This did not take any additional configuration. However, how does our internal zone know what ip address or CIDR range to apply the rules too?

Take a look at the LinuxAcademy.com lab page. You'll see two private ip addresses. In this exercise we are going to specify what servers/computers apply to our "internal zone" and then create a rule to allow those computers to access the httpd server on port 80.

Please note and verify your private ip addresses on the lab page as they may be different than the ones used in this document.

The private CIDR range looks to be inside of our lab network of 10.0.0.0/24. We are going to add a source rule to our zone stating that all traffic from this CIDR range is considered "internal". This way when we add rules to the internal zone we know it is going to apply for all servers on this network. 10.x.x.x is a PRIVATE network. You cannot use those address to access the servers externally (from your computer to the server). However, they can be used between the two since the two servers are on the same private network.

Exercise: Add the source and create the rule for port 80

1. **firewall-cmd --permanent --zone=internal --add-source=10.0.0.0/24**
2. Open server2 server and attempt to open the webpage located on server1 servers PRIVATE ip address. The private ip address needs to be used because we are communicating on the internal network. If we did not specify the private ip but the public, then our traffic would be routed over a different network and would be considered external.
3. **curl http://x.x.x.x**
    a. You will notice the command does not complete or load a webpage. Port 80 is still being blocked for this server but we did specify the server's internal ip address as part of the internal network. Now we need to add a rule for port 80.
4. On server1 **firewall-cmd --permanent --zone=internal --add-port=80/tcp**
5. **firewall-cmd --reload**
6. On Server2 attempt to access server1 httpd server using the PRIVATE ip address provided on the linuxacademy.com lab page for server1.
7. **curl 10.0.0.x**
8. The curl command will return Hello World.

Panic Mode

For our last step, you're going to learn how to enter panic mode. Panic mode essentially stops all incoming and outgoing packets. Panic mode should be used when you're actively under attack or think that your system is actively being compromised. In this example, when we enable panic mode we are going to be booted from the system. That is because in order to communicate with the lab servers we have to use network ssh. Panic mode is generally for when you're physically in front of the machine.

1. **firewall-cmd --panic-on**
2. On a machine you were physically in front of, you could get the status of panic with **firewall-cmd --query-panic** and also turn off panic mode with **firewall-cmd --panic-off**

You are instantly booted from the system, in fact your command will not even complete. Again, this is for machines that you're generally physically in front of.

Conclusion

In this lab you were introduced to Firewalld. We worked with runtime rule changes and persistent rule changes. We also learned how to use zones to specify rules for different computers. We can specify certain ports for computers on our local network and different ports for computers on public network. Again, there are more zones than this but zones help us classify and organize our sources and rules for those sources.