Linux Academy

Live! Lab

Red Hat Security with FirewallD

Contents

Lab Connection Information

- Labs may take up to five minutes to build
- The IP address of your server is located on the Live! Lab page
- Username: linuxacademy
- Password: 123456
- Root Password: 123456

Related Courses

Linux Academy Red Hat Certified Systems Administrator Prep Course

Related Videos

Configure Firewall Settings

Need Help?

Linux Academy Community

... and you can always send in a support ticket on our website to talk to an instructor!

FirewallD is a dynamic firewall provided with Red Hat 7. FirewallD is used to interact with the package-filtering framework *Netfilter*, located inside the Linux kernel. Netfilter allows the kernel to inspect every packet throughout the system, allowing the system to read, stop, drop, modify, or reject packages. FirewallD is not installed on our lab system by default.

FirewallD works as 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 from the iptables configuration interface. FirewallD allows for the setting of dynamic rules, which can be implemented in the current runtime environment without reloading the FirewallD system. Unlike iptables, FirewallD runs as a service and interacts with `systemctl`.

Throughout this guide, and for future use, please remember **not** to use `systemctl` commands to restart FirewallD. Always use `firewall-cmd`'s `--reload` option to implement permanent rule changes.

Network Zones

Zones are used to create different levels of trust. We may 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 precisely, rules for ports. We can also assign rules to a zone. For example, we might assign port 80 (used for the Apache web server) 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:** Drop all incoming traffic with no reply, but allow outgoing traffic.
- **Block:** Only network connections initiated from within the system are possible. Drop all incoming traffic with the reply message of *icmp-host-prohibited*.
- **Public:** Allow traffic from any location and only accept specific connections based on firewall rules.
- **External:** For use on external networks with masquerading.
- **DMZ:** For servers/computers in the demilitarized zone that are publicly-accessible.
- **Work:** Use for work areas; this implies trust between the server and the machine on the network that the machines cannot do damage to the server.
- **Home:** Use for home areas; this implies trust between the server and the machine on the network that the machines cannot do damage to the server. Home rejects all incoming traffic unless it is in response or related to outgoing traffic.
- **Internal:** Use on internal networks; this implies trust between the server and the machine on the network that the machines cannot do damage to the server.

- **Trusted:** All network connections are accepted.

Packages, Resources and Prequisites

On Live! Lab page, two servers with public IP addresses are listed for connectivity. 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*.

```
su -
```

Each server has a hostname, *linuxacademy1* and *linuxacademy2*. These names are used throughout the lab to identify to which server we should be performing our command line actions.

On both *linuxacademy1* and *linuxacademy2*, change to the root user and install FirewallD and Apache. We use the Apache web server to demonstrate firewall rules.

```
yum install -y firewalld httpd
systemctl enable httpd; systemctl start httpd echo 'hello world' >> /
var/www/html/index.html
```

Getting Started

Before using FirewallD, we need to mask iptables and ip6tables. From the *linuxacademy1* server, run:

```
systemctl mask iptables systemctl mask ip6tables
```

Should you receive a *Failed to execute operation: Access denied* error, skip this step.

Ensure the FirewallD service is running:

```
systemctl status firewalld
```

If not, start the service:

```
systemctl start firewalld
```

Remember: When applying permanent changes to FirewallD, do not apply changes with `systemctl reload` or `restart`. This does not ensure that the rules have changed.

firewall-cmd

The command to manage FirewallD is `firewall-cmd`. Take a few minutes to explore these commands:

- `firewall-cmd --reload`
- `firewall-cmd --get-zones`
- `firewall-cmd --list-all-zones`
- `firewall-cmd --list-all`
- `firewall-cmd --get-default-zone`

Creating httpd Runtime Firewall Rules

In this scenario, we are going to use *linuxacademy1* server to act as the httpd server.

Get the **Public IP** for Server 1 back on the Linux Academy Live! Lab page. This is the same server as *linuxacademy1*. Open a browser, place the IP in the address bar and hit enter. This attempts to load the page; however, since we have FirewallD installed and started, the page does not load.

Since we are trying 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 the FirewallD configuration to allow incoming and outgoing traffic to the httpd server.

Query the firewall service to see if any rules exist for httpd:

```
firewall-cmd --list-services
```

No httpd service is listed. Add a rule to the public zone by specifying the protocol and port:

```
firewall-cmd --zone=public --add-port=80/tcp
```

If no zone is specified with `--zone`, then the default zone from `firewall-cmd --get-default-zone` is used. Run:

```
firewall-cmd --list-all
```

The `--list-all` parameter lists all sources/services/interfaces/etc. for the default zone. To view information about another zone, you must pass the `--zone` parameter. For example:

```
firewall-cmd --zone=internal --list-all
```

Now attempt to open the IP address again in the browser. You should see the hello world text we created

during the lab setup process.

However, we did not specify the `--permanent` parameter when we created the rule. This means we only made a *runtime* change. A runtime change allows us to modify firewall rules without having to reload the firewall; the changes we make are not persistent. If the server is rebooted, service restarted, or `firewall-cmd --reload` has been executed, our rule goes away.

Run:

```
firewall-cmd --reload
```

Now attempt to refresh the browser window. The connection fails again 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 Persistent httpd 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, etc. So how do we make this change? By adding the `--permanent` parameter to the command when creating the firewall rule.

Run:

```
firewall-cmd --permanent --zone=public --add-port=80/tcp
```

Verify the rule exists with `firewall-cmd --list-all`. The rule, in fact, does NOT exist, because we have not reloaded the firewall. Reload the firewall:

```
firewall-cmd --reload firewall-cmd --list-all
```

Port 80 is now added and accessible.

Remove a Rule

To remove a rule, run:

```
firewall-cmd --remove-port=80/tcp
```

For this, the same rules apply when adding the rule; we did not use `--permanent`, so it is applied only to runtime. When the service is reloaded, the rule will be back. Let's make the change permanent:

```
firewall-cmd --permanent --remove-port=80/tcp firewall-cmd --reload
```

```
firewall-cmd --list-all
```

Modifying Firewalld Rules To Allow Only Internal Traffic

Our zones still need to be configured to allow incoming traffic only from certain areas. For example, our public zone is already set 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 which rules apply?

Take a look at the Live! 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.

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. Your private IP is part of a private network. You cannot use those addresses to access the servers externally (from your computer to the server). However, they can be used between the two lab servers, since the two servers are on the same private network.

Add Source and Create httpd Rule

On *linuxacademy1* run:

```
firewall-cmd --permanent --zone=internal --add-source=10.0.0.0/24
```

Open the *linuxacademy2* server and attempt to open the webpage located on *linuxacademy1*'s 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. To do this, run:

curl http://private.ip.here

You will notice the command does not complete or load a web page. Port 80 is still 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.

On *linuxacademy1* run:

```
firewall-cmd --permanent --zone=internal --add-port=80/tcp
firewall-cmd --reload
```

On *linuxacademy2*, attempt to access the *linuxacademy1* httpd server using the private IP address provided on the LinuxAcademy.com lab page for Server 1:

```
curl http://private.ip.here
```

The hello world page should return.

Panic Mode

Panic mode stops all incoming and outgoing packets. Panic mode should be used when you're actively under attack or think that your system is being compromised. In this example, when we enable panic mode, we are going to be booted from the system. That is because 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.

On *linuxacademy1*, run:

```
firewall-cmd --panic-on
```

On a physical machine, you could get the status of panic with `firewall-cmd --query-panic`, and also turn off panic mode with `firewall-cmd --panic-off`. In this instance, you are instantly booted from the system – your command does not even complete.