linode.com

Introduction to FirewallD on CentOS

Updated Monday, October 12th, 2015 by Linode Contributed by Florent Houbart

Use promo code **DOCS10** for $10 credit on a new account.

This is a Linode Community guide. Write for us and earn $250 per published guide.

FirewallD is frontend controller for iptables used to implement persistent network traffic rules. It provides command line and graphical interfaces and is available in the repositories of most Linux distributions. Working with FirewallD has two main differences compared to directly controlling iptables:

1. FirewallD uses *zones* and *services* instead of chain and rules.
2. It manages rulesets dynamically, allowing updates without breaking existing sessions and connections.

> FirewallD is a wrapper for iptables to allow easier management of iptables rules–it is **not** an iptables replacement. While iptables commands are still available to FirewallD, it's recommended to use only FirewallD commands with FirewallD.

This guide will introduce you to FirewallD, its notions of zones and services, and show you some basic configuration steps.

Installing and Managing FirewallD

FirewallD is included by default with CentOS 7 but it's inactive. Controlling it is the same as with other systemd units.

1. To start the service and enable FirewallD on boot:

```
1   sudo systemctl start firewalld
2   sudo systemctl enable firewalld
```

To stop and disable it:

```
sudo systemctl stop firewalld
sudo systemctl disable firewalld
```

2. Check the firewall status. The output should say either `running` or `not running` .

```
sudo firewall-cmd --state
```

3. To view the status of the FirewallD daemon:

```
sudo systemctl status firewalld
```

Example output:

```
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system
/firewalld.service; disabled)
   Active: active (running) since Wed 2015-09-02 18:03:22
UTC; 1min 12s ago
 Main PID: 11954 (firewalld)
   CGroup: /system.slice/firewalld.service
           └─11954 /usr/bin/python -Es /usr/sbin/firewalld
--nofork --nopid
```

4. To reload a FirewallD configuration:

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

Configuring FirewallD

Firewalld is configured with XML files. Except for very specific configurations, you won't have

to deal with them and **firewall-cmd** should be used instead.

Configuration files are located in two directories:

- `/usr/lib/FirewallD` holds default configurations like default zones and common services. Avoid updating them because those files will be overwritten by each firewalld package update.

- `/etc/firewalld` holds system configuration files. These files will overwrite a default configuration.

Configuration Sets

Firewalld uses two *configuration sets*: Runtime and Permanent. Runtime configuration changes are not retained on reboot or upon restarting FirewallD whereas permanent changes are not applied to a running system.

By default, `firewall-cmd` commands apply to runtime configuration but using the `--permanent` flag will establish a persistent configuration. To add and activate a permanent rule, you can use one of two methods.

1. Add the rule to both the permanent and runtime sets.

```
sudo firewall-cmd --zone=public --add-service=http
--permanent
sudo firewall-cmd --zone=public --add-service=http
```

2. Add the rule to the permanent set and reload FirewallD.

```
sudo firewall-cmd --zone=public --add-service=http
--permanent
sudo firewall-cmd --reload
```

> The reload command drops all runtime configurations and applies a permanent configuration. Because firewalld manages the ruleset dynamically, it won't break an existing connection and session.

Firewall Zones

Zones are pre-constructed rulesets for various trust levels you would likely have for a given location or scenario (e.g. home, public, trusted, etc.). Different zones allow different network services and incoming traffic types while denying everything else. After enabling FirewallD for the first time, *Public* will be the default zone.

Zones can also be applied to different network interfaces. For example, with separate interfaces for both an internal network and the Internet, you can allow DHCP on an internal zone but only HTTP and SSH on external zone. Any interface not explicitly set to a specific zone will be attached to the default zone.

To view the default zone:

```
1   sudo firewall-cmd --get-default-zone
```

To change the default zone:

```
1   sudo firewall-cmd --set-default-zone=internal
```

To see the zones used by your network interface(s):

```
1   sudo firewall-cmd --get-active-zones
```

Example output:

```
1   public
2     interfaces: eth0
```

To get all configurations for a specific zone:

```
1   sudo firewall-cmd --zone=public --list-all
```

Example output:

```
1   public (default, active)
2     interfaces: ens160
```

```
3     sources:
4     services: dhcpv6-client http ssh
5     ports: 12345/tcp
6     masquerade: no
7     forward-ports:
8     icmp-blocks:
9     rich rules:
```

To get all configurations for all zones:

```
1   sudo firewall-cmd --list-all-zones
```

Example output:

```
1    block
2      interfaces:
3      sources:
4      services:
5      ports:
6      masquerade: no
7      forward-ports:
8      icmp-blocks:
9      rich rules:
10
11     ...
12
13   work
14     interfaces:
15     sources:
16     services: dhcpv6-client ipp-client ssh
17     ports:
18     masquerade: no
19     forward-ports:
20     icmp-blocks:
21     rich rules:
```

Working with Services

FirewallD can allow traffic based on predefined rules for specific network services. You can create your own custom serivce rules and add them to any zone. The configuration files for the

default supported services are located at `/usr/lib/firewalld/services` and user-created service files would be in `/etc/firewalld/services`.

To view the default available services:

```
sudo firewall-cmd --get-services
```

As an example, to enable or disable the HTTP service:

```
sudo firewall-cmd --zone=public --add-service=http --permanent
sudo firewall-cmd --zone=public --remove-service=http
--permanent
```

Allowing or Denying an Arbitrary Port/Protocol

As an example: Allow or disable TCP traffic on port 12345.

```
sudo firewall-cmd --zone=public --add-port=12345/tcp --permanent
sudo firewall-cmd --zone=public --remove-port=12345/tcp
--permanent
```

Port Forwarding

The example rule below forwards traffic from port 80 to port 12345 on **the same server**.

```
sudo firewall-cmd --zone="public" --add-forward-
port=port=80:proto=tcp:toport=12345
```

To forward a port to **a different server**:

1. Activate masquerade in the desired zone.

   ```
   sudo firewall-cmd --zone=public --add-masquerade
   ```

2. Add the forward rule. This example forwards traffic from local port 80 to port 8080

on *a remote server* located at the IP address: 123.456.78.9.

```
1   sudo firewall-cmd --zone="public" --add-forward-
    port=port=80:proto=tcp:toport=8080:toaddr=123.456.78.9
```

To remove the rules, substitute `--add` with `--remove`. For example:

```
1   sudo firewall-cmd --zone=public --remove-masquerade
```

Constructing a Ruleset with FirewallD

As an example, here is how you would use FirewallD to assign basic rules to your Linode if you were running a web server.

1. Assign the *dmz* zone as the default zone to eth0. Of the default zones offered, dmz (demilitarized zone) is the most desirable to start with for this application because it allows only SSH and ICMP.

```
1   sudo firewall-cmd --set-default-zone=dmz
2   sudo firewall-cmd --zone=dmz --add-interface=eth0
```

2. Add permanent service rules for HTTP and HTTPS to the dmz zone:

```
    sudo firewall-cmd --zone=dmz --add-service=http
1   --permanent
2   sudo firewall-cmd --zone=dmz --add-service=https
    --permanent
```

3. Reload FirewallD so the rules take effect immediately:

```
1   sudo firewall-cmd --reload
```

If you now run `firewall-cmd --zone=dmz --list-all`, this should be the

output:

```
dmz (default)
  interfaces: eth0
  sources:
  services: http https ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

This tells us that the **dmz** zone is our **default** which applies to the **etho interface**, all network **sources** and **ports**. Incoming HTTP (port 80), HTTPS (port 443) and SSH (port 22) traffic is allowed and since there are no restrictions on IP versioning, this will apply to both IPv4 and IPv6. **Masquerading** and **port forwarding** are not allowed. We have no **ICMP blocks**, so ICMP traffic is fully allowed, and no **rich rules**. All outgoing traffic is allowed.

Advanced Configuration

Services and ports are fine for basic configuration but may be too limiting for advanced scenarios. Rich Rules and Direct Interface allow you to add fully custom firewall rules to any zone for any port, protocol, address and action.

Rich Rules

Rich rules syntax is extensive but fully documented in the firewalld.richlanguage(5) man page (or see `man firewalld.richlanguage` in your terminal). Use `--add-rich-rule`, `--list-rich-rules` and `--remove-rich-rule` with firewall-cmd command to manage them.

Here are some common examples:

Allow all IPv4 traffic from host 192.168.0.14.

```
sudo firewall-cmd --zone=public --add-rich-rule 'rule
family="ipv4" source address=192.168.0.14 accept'
```

Deny IPv4 traffic over TCP from host 192.168.1.10 to port 22.

```
1 sudo firewall-cmd --zone=public --add-rich-rule 'rule
  family="ipv4" source address="192.168.1.10" port port=22
  protocol=tcp reject'
```

Allow IPv4 traffic over TCP from host 10.1.0.3 to port 80, and forward it locally to port 6532.

```
1 sudo firewall-cmd --zone=public --add-rich-rule 'rule
  family=ipv4 source address=10.1.0.3 forward-port port=80
  protocol=tcp to-port=6532'
```

Forward all IPv4 traffic on port 80 to port 8080 on host 172.31.4.2 (masquerade should be active on the zone).

```
1 sudo firewall-cmd --zone=public --add-rich-rule 'rule
  family=ipv4 forward-port port=80 protocol=tcp to-port=8080
  to-addr=172.31.4.2'
```

To list your current Rich Rules:

```
1 sudo firewall-cmd --list-rich-rules
```

iptables Direct Interface

For the most advanced usage, or for iptables experts, FirewallD provides a direct interface that allows you to pass raw iptables commands to it. Direct Interface rules are not persistent unless the `--permanent` is used.

To see all custom chains or rules added to FirewallD:

```
1 firewall-cmd --direct --get-all-chains
2 firewall-cmd --direct --get-all-rules
```

Discussing iptables syntax details goes beyond the scope of this guide. If you want to learn more, you can review our iptables guide.

More Information

You may wish to consult the following resources for additional information on this topic. While these are provided in the hope that they will be useful, please note that we cannot vouch for the accuracy or timeliness of externally hosted materials.

- FirewallD Official Site
- RHEL 7 Security Guide: Introduction to FirewallD
- Fedora Wiki: FirewallD

This guide is published under a CC BY-ND 4.0 license.