

Firewalld:
A new interface to your netfilter stack

Pat Riehecky
Scientific Linux, Fermilab
October 2014

Firewalld

First off:
Why do we need this?

Configuring Default State

- How do you set the default state for the firewall?
 - Manually edit `/etc/sysconfig/iptables`
 - `iptables-save > /etc/sysconfig/iptables`
 - Depending on `/etc/sysconfig/iptables-config`
 - `service iptables save`
 - Write a custom script?
- How do you manipulate the default state with a shell script?
- What if the current firewall state is not what you want for the default?

Virtual Networks/Routes

- Virtual Hosts can have all sorts of additional rules for the Virtual Guests currently running.
 - br0 is on vlan 2 if guest1 is running
 - Set up a NAT for guest2 while it is running
 - But what if the VM moves to another host?
 - But what if the VM is shut down?
 - But what if you need to change the firewall for some other service?
- How would you capture this information and set it as needed?
- How do you make ebtables aware of your iptables/ip6tables changes?

Of Ports and Addresses

- IPv6 has been coming “any day now” since 1998
 - Or was it 2004, or was it 2012, or
- Whenever it shows up, do we, as systems admins, care?
 - If I want ssh accessible, do I care if it is IPv4 or IPv6?
 - If I do care, do I know I care?
 - If I open port 80, should I have to sort out IPv4 or IPv6 addresses?
- You can use iptables or ip6tables
 - but they are a bit different.

So what is the state?

- How would you check to see if the running state is the default state?
- How would you check to see if a given port is open?
- How would you make sure your IPv4 and IPv6 rules match?

FirewallD's purpose is to solve these problems

Firewalld has “zones”

- A zone is simply a master category for a group of rules.
 - A few default zones are provided:
 - Work, public, external, internal, dmz, and so on
 - These provided zones may have different defaults
 - The default zone is “public”
- Why have zones?
 - Your server may sit on two physical networks: public and VM-interconnect
 - Should the rules for VM-interconnect be the same as public?
 - Your server may treat one subnet as special:
 - All hosts in 192.168.20.0/24 can see my NFSv4 shares, access my iscsi luns, and use my apache server. No one else can.

Firewalld's Default State: Public Zone

- Unless explicitly allowed, REJECT all traffic
- Allow ICMP
- Allow SSH
- Allow DHCPv6 Client Responses
- Allow responses to “my traffic”
 - RELATED, ESTABLISHED
- For “most” services you can enable them by name rather than by port/protocol
 - http is port 80/tcp, we know this - so does firewalld
 - ssh is port 22/tcp, we know this - so does firewalld
 - If you find a common service that isn't already known, we can get it defined!
- Your command line tool is firewall-cmd

Setting a Persistent Rule

- Persistent rules are not activated automatically.
 - You must reload the daemon config
 - `systemctl reload firewalld`
 - `firewall-cmd --reload`
- Let's enable apache access in the permanent rules:
 - `firewall-cmd --permanent --zone=public --add-service=http`
 - `firewall-cmd --permanent --zone=public --add-service=https`
- Let's enable mysql access in the permanent rules:
 - `firewall-cmd --permanent --zone=public --add-service=mysql`
- Let's enable port 8000/tcp in the permanent rules:
 - `firewall-cmd --permanent --zone=public --add-port=8000/tcp`

Firewall-cmd acts on your “default” zone if you omit the `--zone=`

Setting a Temporary Rule

- Temporary rules are not saved
 - but they are active RIGHT NOW.
- Let's enable apache access in the temporary rules:
 - `firewall-cmd --zone=public --add-service=http`
 - `firewall-cmd --zone=public --add-service=https`
- Let's enable mysql access in the temporary rules:
 - `firewall-cmd --zone=public --add-service=mysql`
- Let's enable port 8000/tcp in the permanent rules:
 - `firewall-cmd --zone=public --add-port=8000/tcp`

Firewall-cmd acts on your “default” zone if you omit the `--zone=`

Note How I Didn't Set IP Information?

- FirewallD is IPv4 and IPv6 aware.
 - Unless you specify an IP stack, it assumes you mean IPv4 and IPv6
- You don't have to directly manage the IPv6 space unless you actually want to.

Using Zones To Simplify Rules

- You can make your own zone or use the provided ones
 - `firewall-cmd --permanent --new-zone=example`
- Let's make the example zone meet the following rules:
 - All hosts in `192.168.20.0/24` can see my NFSv4 shares, access my iscsi luns, and use my apache server. No one else can.
 - `firewall-cmd --permanent --zone=example --add-source=192.168.20.0/24`
 - `firewall-cmd --permanent --zone=example --add-service=nfs --add-service=iscsi --add-service=http --add-service=https`
 - `firewall-cmd --reload`

But my rules are “messy”! It is OK!

- Sometimes the simple looking rules are too simple looking
 - Allow new IPv4 and IPv6 connections for service ftp and log 1 per minute using the audit subsystem
 - `firewall-cmd --zone=example --add-rich-rule='rule service name="ftp" audit limit value="1/m" accept'`
 - Forward IPv6 port/packets receiving from 1:2:3:4:6:: on port 4011 with protocol tcp to 1::2:3:4:7 on port 4012
 - `firewall-cmd --add-rich-rule='rule family="ipv6" source address="1:2:3:4:6::" forward-port to-addr="1::2:3:4:7" to-port="4012" protocol="tcp" port="4011"'`

Reading the Status

- To get the running status
 - firewall-cmd –list-all-zones
 - Output snipped to fit on slide

example

```
interfaces:
sources: 192.168.20.0/24
services: http https iscsi nfs
ports:
masquerade: no
forward-ports:
icmp-blocks:
rich rules:
rule service name="ftp" audit limit
value="1/m" accept
```

```
public (default, active)
interfaces: eno1
sources:
services: dhcpv6-client ssh
ports:
masquerade: no
forward-ports:
icmp-blocks:
rich rules:
rule family="ipv6" source address="1:2:3:4:6::"
forward-port port="4011" protocol="tcp"
to-port="4012" to-addr="1::2:3:4:7"
```

Reading the Status

- Or `firewall-cmd --zone=example --list-all`
 - Just the information on this zone
- Or you could try:
 - `firewall-cmd --zone=example --list-services`
 - It returns a space separated list of services, in this case:
 - `http https iscsi nfs`

This May Be Neat, But So What?

- Here is some simple puppet code for opening firewall ports.
 - This could be more optimized, but how would you do this with EL5 or EL6?
 - The puppet forge jpopelka/firewalld module works well
 - Jpopelka is one of the firewalld developers

```
service{'httpd': enable=>'true',ensure=>'running'}
```

```
package{'httpd': ensure=>'installed',  
        notify=>Exec['httpfirewall']}
```

```
exec{'httpfirewall':  
     command=>'firewall-cmd --permanent --add-service=http',  
     notify=>Service['firewalld']}
```

There is MUCH more!

- You can define your own custom services
 - Listing as many ports as you wish
- The config files are stored under `/etc/firewalld/`
- You can bind a specific interface to a given zone or zones
- Everything (and I do mean everything) you can configure is exposed over dbus.
 - There is a direct programmatic API into the runtime behavior

For More Information

- RHEL 7 Security Guide, chapter 4.5
- firewalld comes with a very extensive set of manpages.
 - firewall-cmd (1) - firewalld command line client
 - firewall-config (1) - firewalld GUI configuration tool
 - firewall-offline-cmd (1) - firewalld offline command line client
 - firewalld (1) - Dynamic Firewall Manager
 - firewalld.conf (5) - firewalld configuration file
 - firewalld.dbus (5) - firewalld D-Bus interface description
 - firewalld.direct (5) - firewalld direct configuration file
 - firewalld.icmptype (5) - firewalld icmptype configuration files
 - firewalld.lockdown-whitelist (5) - firewalld lockdown whitelist configuration file
 - firewalld.richlanguage (5) - Rich Language Documentation
 - firewalld.service (5) - firewalld service configuration files
 - firewalld.zone (5) - firewalld zone configuration files
 - firewalld.zones (5) - firewalld zones