How To Set Up a Firewall Using FirewallD on CentOS 7

digitalocean.com/community/tutorials/how-to-set-up-a-firewall-using-firewalld-on-centos-7

Introduction

Firewalld is a firewall management solution available for many Linux distributions which acts as a frontend for the iptables packet filtering system provided by the Linux kernel. In this guide, we will cover how to set up a firewall for your server and show you the basics of managing the firewall with the `firewall-cmd` administrative tool (if you'd rather use `iptables` with CentOS, follow this guide).

Note: There is a chance that you may be working with a newer version of firewalld than was available at the time of this writing, or that your server was set up slightly differently than the example server used throughout this guide. Thus, the behavior of some of the commands explained in this guide may vary depending on your specific configuration.

Basic Concepts in Firewalld

Before we begin talking about how to actually use the `firewall-cmd` utility to manage your firewall configuration, we should get familiar with a few basic concepts that the tool introduces.

Zones

The `firewalld` daemon manages groups of rules using entities called "zones". Zones are basically sets of rules dictating what traffic should be allowed depending on the level of trust you have in the networks your computer is connected to. Network interfaces are assigned a zone to dictate the behavior that the firewall should allow.

For computers that might move between networks frequently (like laptops), this kind of flexibility provides a good method of changing your rules depending on your environment. You may have strict rules in place prohibiting most traffic when operating on a public WiFi network, while allowing more relaxed restrictions when connected to your home network. For a server, these zones are not as immediately important because the network environment rarely, if ever, changes.

Regardless of how dynamic your network environment may be, it is still useful to be familiar with the general idea behind each of the predefined zones for `firewalld`. In order from **least trusted** to **most trusted**, the predefined zones within `firewalld` are:

- **drop**: The lowest level of trust. All incoming connections are dropped without reply and only outgoing connections are possible.

- **block**: Similar to the above, but instead of simply dropping connections, incoming requests are rejected with an `icmp-host-prohibited` or `icmp6-adm-prohibited` message.
- **public**: Represents public, untrusted networks. You don't trust other computers but may allow selected incoming connections on a case-by-case basis.
- **external**: External networks in the event that you are using the firewall as your gateway. It is configured for NAT masquerading so that your internal network remains private but reachable.
- **internal**: The other side of the external zone, used for the internal portion of a gateway. The computers are fairly trustworthy and some additional services are available.
- **dmz**: Used for computers located in a DMZ (isolated computers that will not have access to the rest of your network). Only certain incoming connections are allowed.
- **work**: Used for work machines. Trust most of the computers in the network. A few more services might be allowed.
- **home**: A home environment. It generally implies that you trust most of the other computers and that a few more services will be accepted.
- **trusted**: Trust all of the machines in the network. The most open of the available options and should be used sparingly.

To use the firewall, we can create rules and alter the properties of our zones and then assign our network interfaces to whichever zones are most appropriate.

Rule Permanence

In firewalld, rules can be designated as either permanent or immediate. If a rule is added or modified, by default, the behavior of the currently running firewall is modified. At the next boot, the old rules will be reverted.

Most `firewall-cmd` operations can take the `--permanent` flag to indicate that the non-ephemeral firewall should be targeted. This will affect the rule set that is reloaded upon boot. This separation means that you can test rules in your active firewall instance and then reload if there are problems. You can also use the `--permanent` flag to build out an entire set of rules over time that will all be applied at once when the reload command is issued.

Install and Enable Your Firewall to Start at Boot

`firewalld` is installed by default on some Linux distributions, including many images of CentOS 7. However, it may be necessary for you to install firewalld yourself:

```
sudo yum install firewalld
```

After you install `firewalld`, you can enable the service and reboot your server. Keep in mind that enabling firewalld will cause the service to start up at boot. It is best practice to create your firewall rules and take the opportunity to test them before configuring this behavior in order to avoid potential issues.

- sudo systemctl enable firewalld
- sudo reboot

When the server restarts, your firewall should be brought up, your network interfaces should be put into the zones you configured (or fall back to the configured default zone), and any rules associated with the zone(s) will be applied to the associated interfaces.

We can verify that the service is running and reachable by typing:

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

output

```
running
```

This indicates that our firewall is up and running with the default configuration.

Getting Familiar with the Current Firewall Rules

Before we begin to make modifications, we should familiarize ourselves with the default environment and rules provided by the daemon.

Exploring the Defaults

We can see which zone is currently selected as the default by typing:

```
firewall-cmd --get-default-zone
```

output

```
public
```

Since we haven’t given `firewalld` any commands to deviate from the default zone, and none of our interfaces are configured to bind to another zone, that zone will also be the only “active” zone (the zone that is controlling the traffic for our interfaces). We can verify that by typing:

```
firewall-cmd --get-active-zones
```

output

```
public
  interfaces: eth0 eth1
```

Here, we can see that our example server has two network interfaces being controlled by the firewall (`eth0` and `eth1`). They are both currently being managed according to the rules defined for the public zone.

How do we know what rules are associated with the public zone though? We can print out the default zone's configuration by typing:

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

output

```
public (default, active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0 eth1
  sources:
  services: ssh dhcpv6-client
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

We can tell from the output that this zone is both the default and active and that the `eth0` and `eth1` interfaces are associated with this zone (we already knew all of this from our previous inquiries). However, we can also see that this zone allows for the normal operations associated with a DHCP client (for IP address assignment) and SSH (for remote administration).

Exploring Alternative Zones

Now we have a good idea about the configuration for the default and active zone. We can find out information about other zones as well.

To get a list of the available zones, type:

```
firewall-cmd --get-zones
```

output

```
block dmz drop external home internal public trusted work
```

We can see the specific configuration associated with a zone by including the `--zone=` parameter in our `--list-all` command:

```
sudo firewall-cmd --zone=home --list-all
```

output

```
home
  interfaces:
  sources:
  services: dhcpv6-client ipp-client mdns samba-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

You can output all of the zone definitions by using the `--list-all-zones` option. You will probably want to pipe the output into a pager for easier viewing:

```
sudo firewall-cmd --list-all-zones | less
```

Selecting Zones for your Interfaces

Unless you have configured your network interfaces otherwise, each interface will be put in the default zone when the firewall is booted.

Changing the Zone of an Interface

You can transition an interface between zones during a session by using the `--zone=` parameter in combination with the `--change-interface=` parameter. As with all commands that modify the firewall, you will need to use `sudo` .

For instance, we can transition our `eth0` interface to the “home” zone by typing this:

```
sudo firewall-cmd --zone=home --change-interface=eth0
```

output

```
success
```

Note

Whenever you are transitioning an interface to a new zone, be aware that you are probably modifying the services that will be operational. For instance, here we are moving to the “home” zone, which has SSH available. This means that our connection shouldn’t drop. Some other zones do not have SSH enabled by default and if your connection is dropped while using one of these zones, you could find yourself unable to log back in.
We can verify that this was successful by asking for the active zones again:

```
firewall-cmd --get-active-zones
```

output

```
home
  interfaces: eth0
public
  interfaces: eth1
```

Adjusting the Default Zone

If all of your interfaces can best be handled by a single zone, it's probably easier to just select the best default zone and then use that for your configuration.

You can change the default zone with the `--set-default-zone=` parameter. This will immediately change any interface that had fallen back on the default to the new zone:

```
sudo firewall-cmd --set-default-zone=home
```

output

```
success
```

Setting Rules for your Applications

The basic way of defining firewall exceptions for the services you wish to make available is easy. We'll run through the basic idea here.

Adding a Service to your Zones

The easiest method is to add the services or ports you need to the zones you are using. Again, you can get a list of the available services with the `--get-services` option:

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

output

```
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client bitcoin bitcoin-rpc bitcoin-testnet
bitcoin-testnet-rpc ceph ceph-mon cfengine condor-collector ctdb dhcp dhcpv6 dhcpv6-client dns
docker-registry dropbox-lansync elasticsearch freeipa-ldap freeipa-ldaps freeipa-replication freeipa-
trust ftp ganglia-client ganglia-master high-availability http https imap imaps ipp ipp-client ipsec iscsi-
target kadmin kerberos kibana klogin kpasswd kshell ldap ldaps libvirt libvirt-tls managesieve mdns
mosh mountd ms-wbt mssql mysql nfs nrpe ntp openvpn ovirt-imageio ovirt-storageconsole ovirt-
vmconsole pmcd pmproxy pmwebapi pmwebapis pop3 pop3s postgresql privoxy proxy-dhcp ptp
pulseaudio puppetmaster quassel radius rpc-bind rsh rsyncd samba samba-client sane sip sips smtp
smtp-submission smtps snmp snmptrap spideroak-lansync squid ssh synergy syslog syslog-tls telnet
tftp tftp-client tinc tor-socks transmission-client vdsm vnc-server wbem-https xmpp-bosh xmpp-client
xmpp-local xmpp-server
```

Note

You can get more details about each of these services by looking at their associated `.xml` file within the `/usr/lib/firewalld/services` directory. For instance, the SSH service is defined like this:

/usr/lib/firewalld/services/ssh.xml

```
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>SSH</short>
  <description>Secure Shell (SSH) is a protocol for logging into and executing commands on remote machines. It provides secure encrypted communications. If you plan on accessing your machine remotely via SSH over a firewalled interface, enable this option. You need the openssh-server package installed for this option to be useful.</description>
  <port protocol="tcp" port="22"/>
</service>
```

You can enable a service for a zone using the `--add-service=` parameter. The operation will target the default zone or whatever zone is specified by the `--zone=` parameter. By default, this will only adjust the current firewall session. You can adjust the permanent firewall configuration by including the `--permanent` flag.

For instance, if we are running a web server serving conventional HTTP traffic, we can allow this traffic for interfaces in our “public” zone for this session by typing:

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

You can leave out the `--zone=` if you wish to modify the default zone. We can verify the operation was successful by using the `--list-all` or `--list-services` operations:

```
sudo firewall-cmd --zone=public --list-services
```

output

```
dhcpv6-client http ssh
```

Once you have tested that everything is working as it should, you will probably want to modify the permanent firewall rules so that your service will still be available after a reboot. We can make our “public” zone change permanent by typing:

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

output

```
success
```

You can verify that this was successful by adding the `--permanent` flag to the `--list-services` operation. You need to use `sudo` for any `--permanent` operations:

```
sudo firewall-cmd --zone=public --permanent --list-services
```

output

```
dhcpv6-client http ssh
```

Your “public” zone will now allow HTTP web traffic on port 80. If your web server is configured to use SSL/TLS, you’ll also want to add the `https` service. We can add that to the current session and the permanent rule-set by typing:

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

What If No Appropriate Service Is Available?

The firewall services that are included with the firewalld installation represent many of the most common requirements for applications that you may wish to allow access to. However, there will likely be scenarios where these services do not fit your requirements.

In this situation, you have two options.

Opening a Port for your Zones

The easiest way to add support for your specific application is to open up the ports that it uses in the appropriate zone(s). This is as easy as specifying the port or port range, and the associated protocol for the ports you need to open.

For instance, if our application runs on port 5000 and uses TCP, we could add this to the “public” zone for this session using the `--add-port=` parameter. Protocols can be either `tcp` or `udp`:

```
sudo firewall-cmd --zone=public --add-port=5000/tcp
```

output

```
success
```

We can verify that this was successful using the `--list-ports` operation:

```
sudo firewall-cmd --zone=public --list-ports
```

output

```
5000/tcp
```

It is also possible to specify a sequential range of ports by separating the beginning and ending port in the range with a dash. For instance, if our application uses UDP ports 4990 to 4999, we could open these up on “public” by typing:

```
sudo firewall-cmd --zone=public --add-port=4990-4999/udp
```

After testing, we would likely want to add these to the permanent firewall. You can do that by typing:

- sudo firewall-cmd --zone=public --permanent --add-port=5000/tcp
- sudo firewall-cmd --zone=public --permanent --add-port=4990-4999/udp
- sudo firewall-cmd --zone=public --permanent --list-ports

output

```
success
success
5000/tcp 4990-4999/udp
```

Defining a Service

Opening ports for your zones is easy, but it can be difficult to keep track of what each one is for. If you ever decommission a service on your server, you may have a hard time remembering which ports that have been opened are still required. To avoid this situation, it is possible to define a service.

Services are simply collections of ports with an associated name and description. Using services is easier to administer than ports, but requires a bit of upfront work. The easiest way to start is to copy an existing script (found in `/usr/lib/firewalld/services`) to the `/etc/firewalld/services` directory where the firewall looks for non-standard definitions.

For instance, we could copy the SSH service definition to use for our “example” service definition like this. The filename minus the `.xml` suffix will dictate the name of the service within the firewall services list:

```
sudo cp /usr/lib/firewalld/services/ssh.xml /etc/firewalld/services/example.xml
```

Now, you can adjust the definition found in the file you copied:

```
sudo vi /etc/firewalld/services/example.xml
```

To start, the file will contain the SSH definition that you copied:

/etc/firewalld/services/example.xml

```
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>SSH</short>
  <description>Secure Shell (SSH) is a protocol for logging into and executing commands on remote machines. It provides secure encrypted communications. If you plan on accessing your machine remotely via SSH over a firewalled interface, enable this option. You need the openssh-server package installed for this option to be useful.</description>
  <port protocol="tcp" port="22"/>
</service>
```

The majority of this definition is actually metadata. You will want to change the short name for the service within the `<short>` tags. This is a human-readable name for your service. You should also add a description so that you have more information if you ever need to audit the service. The only configuration you need to make that actually affects the functionality of the service will likely be the port definition where you identify the port number and protocol you wish to open. This can be specified multiple times.

For our “example” service, imagine that we need to open up port 7777 for TCP and 8888 for UDP. By entering INSERT mode by pressing `i`, we can modify the existing definition with something like this:

/etc/firewalld/services/example.xml

```
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>Example Service</short>
  <description>This is just an example service.  It probably shouldn't be used on a real system.</description>
  <port protocol="tcp" port="7777"/>
  <port protocol="udp" port="8888"/>
</service>
```

Press `ESC`, then enter `:x` to save and close the file.

Reload your firewall to get access to your new service:

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

You can see that it is now among the list of available services:

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

output

```
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client bitcoin bitcoin-rpc bitcoin-testnet
bitcoin-testnet-rpc ceph ceph-mon cfengine condor-collector ctdb dhcp dhcpv6 dhcpv6-client dns
docker-registry dropbox-lansync elasticsearch example freeipa-ldap freeipa-ldaps freeipa-replication
freeipa-trust ftp ganglia-client ganglia-master high-availability http https imap imaps ipp ipp-client
ipsec iscsi-target kadmin kerberos kibana klogin kpasswd kshell ldap ldaps libvirt libvirt-tls
managesieve mdns mosh mountd ms-wbt mssql mysql nfs nrpe ntp openvpn ovirt-imageio ovirt-
storageconsole ovirt-vmconsole pmcd pmproxy pmwebapi pmwebapis pop3 pop3s postgresql privoxy
proxy-dhcp ptp pulseaudio puppetmaster quassel radius rpc-bind rsh rsyncd samba samba-client sane
sip sips smtp smtp-submission smtps snmp snmptrap spideroak-lansync squid ssh synergy syslog
syslog-tls telnet tftp tftp-client tinc tor-socks transmission-client vdsm vnc-server wbem-https xmpp-
bosh xmpp-client xmpp-local xmpp-server
```

You can now use this service in your zones as you normally would.

Creating Your Own Zones

While the predefined zones will probably be more than enough for most users, it can be helpful to define your own zones that are more descriptive of their function.

For instance, you might want to create a zone for your web server, called “publicweb”. However, you might want to have another zone configured for the DNS service you provide on your private network. You might want a zone called “privateDNS” for that.

When adding a zone, you must add it to the permanent firewall configuration. You can then reload to bring the configuration into your running session. For instance, we could create the two zones we discussed above by typing:

```
sudo firewall-cmd --permanent --new-zone=publicweb
sudo firewall-cmd --permanent --new-zone=privateDNS
```

You can verify that these are present in your permanent configuration by typing:

```
sudo firewall-cmd --permanent --get-zones
```

output

```
block dmz drop external home internal privateDNS public publicweb trusted work
```

As stated before, these won’t be available in the current instance of the firewall yet:

```
firewall-cmd --get-zones
```

output

```
block dmz drop external home internal public trusted work
```

Reload the firewall to bring these new zones into the active configuration:

- sudo firewall-cmd --reload
- firewall-cmd --get-zones

```
output

block dmz drop external home internal privateDNS public publicweb trusted work
```

Now, you can begin assigning the appropriate services and ports to your zones. It’s usually a good idea to adjust the active instance and then transfer those changes to the permanent configuration after testing. For instance, for the “publicweb” zone, you might want to add the SSH, HTTP, and HTTPS services:

- sudo firewall-cmd --zone=publicweb --add-service=ssh
- sudo firewall-cmd --zone=publicweb --add-service=http
- sudo firewall-cmd --zone=publicweb --add-service=https
- sudo firewall-cmd --zone=publicweb --list-all

```
output

publicweb
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh http https
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Likewise, we can add the DNS service to our “privateDNS” zone:

- sudo firewall-cmd --zone=privateDNS --add-service=dns
- sudo firewall-cmd --zone=privateDNS --list-all

```
output

privateDNS
  interfaces:
  sources:
  services: dns
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

We could then change our interfaces over to these new zones to test them out:

- sudo firewall-cmd --zone=publicweb --change-interface=eth0
- sudo firewall-cmd --zone=privateDNS --change-interface=eth1

At this point, you have the opportunity to test your configuration. If these values work for you, you will want to add the same rules to the permanent configuration. You can do that by re-applying the rules with the `--permanent` flag:

- sudo firewall-cmd --zone=publicweb --permanent --add-service=ssh
- sudo firewall-cmd --zone=publicweb --permanent --add-service=http
- sudo firewall-cmd --zone=publicweb --permanent --add-service=https
- sudo firewall-cmd --zone=privateDNS --permanent --add-service=dns

After permanently applying these your rules, you can restart your network and reload your firewall service:

- sudo systemctl restart network
- sudo systemctl reload firewalld

Validate that the correct zones were assigned:

```
firewall-cmd --get-active-zones
```

```
output

privateDNS
  interfaces: eth1
publicweb
  interfaces: eth0
```

And validate that the appropriate services are available for both of the zones:

```
sudo firewall-cmd --zone=publicweb --list-services
```

```
output

http https ssh
```

```
sudo firewall-cmd --zone=privateDNS --list-services
```

```
output

dns
```

You have successfully set up your own zones! If you want to make one of these zones the default for other interfaces, remember to configure that behavior with the `--set-default-zone=` parameter:

```
sudo firewall-cmd --set-default-zone=publicweb
```

Conclusion

You should now have a fairly good understanding of how to administer the firewalld service on your CentOS system for day-to-day use.

The firewalld service allows you to configure maintainable rules and rule-sets that take into consideration your network environment. It allows you to seamlessly transition between different firewall policies through the use of zones and gives administrators the ability to abstract the port management into more friendly service definitions. Acquiring a working knowledge of this system will allow you to take advantage of the flexibility and power that this tool provides.