Chapter 36 Firewalls - Notes

36.2 Introduction

Firewalls are used to control both incoming and outgoing access to your systems and local network, and are an essential security facility in modern networks, where intrusions and other kinds of attacks are a fact of life on any computer connected to the internet. You can control the level of trust afforded on traffic across particular interfaces, and/or with particular network addresses.

36.3 Learning Objectives:

- Understand what firewalls are and why they are necessary.
- Know what tools are available both at the command line and using graphical interfaces.
- Discuss about **firewalld** and the **firewall-cmd** programs.
- Know how to work with **zones**, **sources**, **services**, and **ports**.

36.4 What Is a Firewall?

Firewall: network security system that monitors and controls all network traffic. Applies **rules** on both incoming and outgoing network connections and packets and builds flexible barrier (i.e., firewalls) depending on the level of trust of a given connection.

Firewalls can be hardware or software based. They are found both in network routers, as well as in individual computers, or network nodes. Many firewalls also have routing capabilities.

36.5 Packet Filtering

Almost all firewalls based on **Packet Filtering**.

Information transmitted across networks in form of packets, and each one of these packets has:

- Header
- Payload
- Footer.

Header and footer contain information about destination and source addresses, what kind of packet it is, which protocol it obeys, various flags, which packet number this is in a stream, all sorts of other metadata about transmissions. Actual data in payload.

Packet filtering intercepts packets at one or more stages in network transmission, including application, transport, network, datalink.

Firewall establishes set of rules by which each packet may be:

- Accepted or rejected based on content, address, etc.
- Mangled in some way
- Redirected to another address
- Inspected for security reasons
- Etc.

Various utilities exist for establishing rules and actions to be taken as the result of packet filtering.

36.6 Firewall Generations

Early firewalls (dating back to the late 1980's) based on **packet filtering**: content of each network packet inspected and either dropped, rejected, or sent on. No consideration given about **connection state**: what stream of traffic the packet was part of.

Next generation of firewalls based on **stateful filters**, which also examine **connection state** of packet, to see if it is a new connection, part of an already existing one, or part of none. Denial of service attacks can bombard this kind of firewall to try and overwhelm it.

Third generation of firewalls called **Application Layer Firewalls**, aware of the kind of application and protocol the connection is using. Can block anything which should not be part of the normal flow.

36.7 Firewall Interfaces and Tools

Configuring your system's firewall can be done by:

- Using relatively low-level tools from the command line, combined with editing various configuration files in the `/etc` subdirectory tree:

  'iptables'

  'firewall-cmd'

  'ufw'

- Using robust graphical interfaces:

  'system-config-firewall'

  'firewall-config'

  'gufw'

  'yast'

Will work with lower-level tools for following reasons:

- Change less often than graphical ones
- Tend to have larger set of capabilities
- Tend to be quite different and each confined to GUI. Vary little from distribution to distribution, while only one family of distributions

Disadvantage: can seem more difficult to learn at first. In following, will concentrate on use of modern **firewalld** package, includes both **firewall-cmd** and **firewall-config**. FOr distributions which don't have it by default, can be installed from source rather easily, as will do if necessary in exercise.

36.8 Why We Are Not Working with iptables

More firewall installations today actually use **iptables** package on user side. This currently interfaces same kernel firewall implementation code as **firewalld**, which will be discussed more in detail.

Decided not to teach **iptables** because it requires much more time to get to useful functionality.

However, **iptables** discussed in detail in next course in Linux Foundation system administrator sequence: *LFS311 - Linux for System Engineers*/*LFS211 - Linux Networking and Administration*.

36.9 firewalld

firewalld: **Dynamic Firewall Manager**. Utilizes network/firewall **zones** which have defined levels of trust for network interfaces or connections. Supports both IPv4 and IPv6 protocols.

In addition, separates **runtime** and **permanent** (persistent) changes to configuration, and also includes interfaces for services/applications to add firewall rules.

Configuration files kept in `/etc/firewalld` and `/usr/lib/firewalld`. Files in `/etc/firewalld` override those in other directory and are the ones system administrators should work on.

Command line tool actually **firewall-cmd** which will be discussed. Run before getting any further:

```
$ firewall-cmd --help
Usage: firewall-cmd [OPTIONS...]
....
Status options
   --state                Return and print firewalld state
   --reload               Reload firewall and keep state information
   --complete-reload      Reload firewall and loose state information
   --runtime-to-permanent  Create permanent from runtime configuration
....
```

which runs about 200 lines, too long to be included here.

Note: will see that almost all options rather obvious, as well named. As a service, **firewalld** replaces older **iptables** Error to run both services, **firewalld** and **iptables**, at same time.

36.10 firewalld Service Status

firewalld: service which needs to be running to use and configure the firewall. Enabled/disabled, or started/stopped in usual way:

```
$ sudo systemctl [enable/disable] firewalld
$ sudo systemctl [start/stop] firewalld
```

Can show current state in either of the following ways:

```
$ sudo systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
    Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled)
    Active: active (running) since Tue 2015-04-28 12:00:59 CDT; 5min ago
 Main PID: 777 (firewalld)
    CGroup: /system.slice/firewalld.service
            777  /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid
Apr 28 12:00:59 CenOS7 systemd[1]: Started firewalld - dynamic firewall daemon.
$ sudo firewall-cmd --state
running
```

Note: if you have more than one network interface when using IPv4, have to turn on **ip forwarding**. Can do this at runtime by doing either of:

```
$ sudo sysctl net.ipv4 ip_forward=1
$ echo 1 > /proc/sys/net/ipv4/ip_forward (needs to be run as root to get echo to work properly)
```

However, this is not persistent. To do that, have to add following line to `/etc/sysctl.conf`:

```
net.ipv4.ip_forward=1
```

and then reboot or type:

```
$ sudo sysctl -p
```

to read in new setting without rebooting.

36.11 Zones

firewalld works with **zones**, each of which has defined level of trust and certain known behavior for incoming/outgoing packets. Each interface belongs to particular zone (normally, it is **Network Manager** which informs **firewalld** which zone is applicable), but this can be changed with **firewallcmd** or the **firewall-config** GUI.

The zones:

- **drop**

  All incoming packets dropped with no reply. Only outgoing connections are permitted.

- **block**

  All incoming network connections rejected. Only permitted connections are those from within the system.

- **public**

  DO not trust any computers on the network; only certain consciously selected incoming connections are permitted.

- **external**

  Used when masquerading is being used, such as in routers. Trust levels are the same as in public.

- **dmz (Demilitarized Zone)**

  Used when access to some (but not all) services are to be allowed to the public. Only particular incoming connections are allowed.

- **work**

  Trust (but not completely) connected nodes to be not harmful. Only certain incoming connections are allowed.

- **home**

  Mostly trust the other network nodes, but till select which incoming connections are allowed.

- **internal**

  Similar to **work** zone.

- **trusted**

  All network connections are allowed.

On system installation, most, if not all Linux distributions, will select the **public** zone as default for all interfaces.

The differences between some of the zones mentioned not obvious, do not need to go into that much detail. Note: one should not use a more open zone than necessary.

36.12 Zone Management

Get the default zone:

```
$ sudo firewall-cmd --get-default-zone
public
```

Obtain a list of zones currently being used:

```
$ sudo firewall-cmd --get-active-zones
public
  interfaces: eno16777736
```

List all available zones:

```
$ sudo firewall-cmd --get-zone
block dmz drop external home internal public trusted work
```

To change the default zone to **trusted** and then change it back:

```
$ sudo firewall-cmd --set-default-zone=trusted
success
$ sudo firewall-cmd --set-default-zone=public
success
```

To assign an interface temporarily to a particular zone:

```
$ sudo firewall-cmd --zone=internal --change-interface=eno1
success
```

To assign an interface to a particular zone permanently:

```
$ sudo firewall-cmd --permanent --zone=internal --change-interface=eno1
success
```

which creates the file `/etc/firewalld/zones/internal.xml`.

To ascertain the zone associated with a particular interface:

```
$ sudo firewall-cmd --get-zone-of-interface=eno1
public
```

Finally, to get all details about a particular zone:

```
$ sudo firewall-cmd --permanent --zone=public --list-all
public (default, active)
  interfaces: eno16777736
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

36.13 Source Management

Any zone can be bound not just to network interface, but also to particular network addresses. Packet associated with zone if:

- it comes from source address already bound to the zone; or if not,
- it comes from an interface bound to the zone.

Any packet not fitting the above criteria is assigned to the default zone (i.e., usually **public**).

To assign a source to a zone (permanently):

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=192.168.1.0/24
success
```

This says anyone with an IP address of `192.168.1.x` will be added to the **trusted** zone.

Note: can remove previously assigned source from zone by using `--remove-source` option, or change zone by using `--change-source`.

Can list the sources bound to a zone with:

```
$ sudo firewall-cmd --permanent --zone=trusted --list-sources
192.168.1.0/24
```

In both of above commands, if you leave out the `--permanent` option, you get only the current runtime behavior.

36.14 Service Management

So far, have assigned particular interfaces and/or addresses to zones, but haven't delineated what **services** and **ports** should be accessible within a zone.

To see all the services available:

```
$ sudo firewall-cmd --get-services
RH-Satellite-6 amanda-client bacula bacula-client dhcp dhcpv6 dhcpv6-client dns ftp \
high-availability http https imaps ipp ipp-client ipsec kerberos kpasswd ldap ldaps \
```

```
libvirt libvirt-tls mdns mountd ms-wbt mysql nfs ntp openvpn pmcd pmproxy pmwebapi \
pmwebapis pop3s postgresql proxy-dhcp radius rpc-bind samba samba-client smtp ssh \
telnet tftp tftp-client transmission-client vnc-server wbem-https
```

or, to see those currently accessible in a particular zone:

```
$ sudo firewall-cmd --list-services --zone=public
dhcpv6-client ssh
```

To add a service to a zone:

```
$ sudo firewall-cmd --permanent --zone=home --add-service=dhcp
success
$ sudo firewall-cmd --reload
```

Second command, w ith `reload`, needed to make change effective. Also possible to add new services by editing files in `/etc/firewalld/services`.

36.15 Port Management

Port management very similar to service management:

```
$ sudo firewall-cmd --zone=home --add-port=21/tcp
success
$ sudo firewall-cmd --zone=home --list-ports
21/tcp
```

w here by looking at `/etc/services`, can ascertain that port 21 corresponds to **ftp**:

```
$ grep " 21/tcp" /etc/services
ftp                21/tcp
```

##

Back to top

Previous Chapter - Table of Contents - Next Chapter