ProLUG – firewalld configuration

Required Materials

Rocky 9.3 or equivalent

Root or sudo command access

EXERCISES (Warmup to quickly run through your system and familiarize yourself)

1. cd ~
2. pwd (should be /home/<yourusername>)
3. cd /tmp
4. pwd (should be /tmp)
5. cd
6. pwd (should be /home/<yourusername>)
7. mkdir lab_firewalld
8. cd lab_firewalld
9. touch testfile1
10. ls
11. touch testfile{2..10}
12. ls
13. seq 10
14. seq 1 10
15. seq 1 2 10
    man seq and see what each of those values mean. It's important to know the behavior if you intend to ever use the command, as we often do with counting (for) loops.
16. for i in `seq 1 10`; do touch file$i; done;
17. ls

    Think about some of those commands and when you might use them. Try to change command #15 to remove all of those files (rm -rf file$i)

Prelab work – Screen as an administration tool.

There's a useful tool out there called screen that can make administration work much easier for your system. Your system likely does not have screen, so let's install it and look quickly at how it works.

1. rpm –qa | grep –i screen
   Do you have it?
2. yum whatprovides screen

Look at the packages that provide screen. What repo is it going to pull from? We'll work with repos in a couple of weeks, but this is something to know about your packages; where do they come from?

3. yum install screen or yum –y install screen
4. man screen
   Read that first page about screen and get a feel for what it does.

   We're going to test what screen does real quick so you can understand the concept of nohup. Screen creates a nohup environment where your shell remains attached even if you are no longer connected. More information can be found here if you're curious
   https://en.wikipedia.org/wiki/Nohup

   So if we want to test, do this to see what happens without screen in a normal environment:

1. watch ls -l
   this will execute ls –l every 2 seconds in your terminal
2. click the "x" at the top right of the window, killing your putty shell
3. Log back into your server
   Is your command still running? (Of course not)

   If we want to test while running screen to nohup our process:

1. screen
   You will be in a screen terminal session although everything will look the same.
2. watch ls -l
   this will execute ls –l every 2 seconds in your terminal
3. click the "x" at the top right of the window, killing your putty shell
4. Log back into your server
   Is your command still running? (Of course not, not in this new shell
5. screen –r
   You will reconnect to screen and now see if your process is still running.

Screen is this tremendously useful tool and has a lot of options that can be found in the man pages. The main ones that I use are the control + a key combination (C-a) notation + some other key. Try a few of these to see screen's behavior

C-a c (so hit ctrl+a and then the letter c, do that a a few times to create windows)

C-a n (so hit ctrl+a and then the letter n, to move through the windows forward - next)

C-a p (so hit ctrl+a and then the letter p, to move through the windows in reverse – previous)

C-a a (will toggle the last window and your current window back and forth)

Once you've done this and have a few windows created you can exit at any time to close each individual window. You can do commands in each window like date or id to see the differences as you toggle through them.

The one last item, which we're not going to go into here that you should probably check out on your own, is your .screenrc file. If you create a file in /home/<yourusername/ named .screenrc you can change the default startup behavior of your screen sessions. Feel free to go find some online (google: .screenrc file) and you will find many examples that you can use. Screen is as simple and useful, or rich featured and robust as you want to spend the time making it. It can be extremely powerful for administration and engineering environment design.

My screen file is as follows (this can be copied and pasted):

```
$ cat .screenrc

# GNU Screen - main configuration file
# All other .screenrc files will source this file to inherit settings.
# Author: Christian Wills - cwills.sys@gmail.com

# Allow bold colors - necessary for some reason
attrcolor b ".I"

# Tell screen how to set colors. AB = background, AF=foreground
termcapinfo xterm 'Co#256:AB=\E[48;5;%dm:AF=\E[38;5;%dm'

# Enables use of shift-PgUp and shift-PgDn
termcapinfo xterm|xterms|xs|rxvt ti@:te@

# Erase background with current bg color
defbce "on"

# Enable 256 color term
term xterm-256color

# Cache 30000 lines for scroll back
defscrollback 30000

# New mail notification
#backtick 101 30 15 $HOME/bin/mailstatus.sh

hardstatus alwayslastline
# Very nice tabbed colored hardstatus line
hardstatus string '%{= Kd} %{= Kd}%-w%{= Kr}[%{= KW}%n %t%{= Kr}]%{= Kd}%+w %-= %{KG} %H%
{KW}|%{KY}%101`%{KW}|%D %M %d %Y%{= Kc} %C%A%{-}'

# change command character from ctrl-a to ctrl-b (emacs users may want this)
#escape ^Bb

# Hide hardstatus: ctrl-a f
bind f eval "hardstatus ignore"
# Show hardstatus: ctrl-a F
bind F eval "hardstatus alwayslastline"

# Display a caption string below, appearing like tabs and
```

```
# # displaying the window number and application name (by default).
caption always
caption string "%{kw}%-w%{wr}%n %t%{-}%+w"
```

LAB

Check Firewall Status and settings

A very important thing to note before starting this lab. You're connected into that server on ssh via port 22. If you do anything to lockout port 22 in this lab, you will be blocked from that connection and we'll have to reset it.

1. Check firewall status

[root@schampine ~]# systemctl status firewalld

```
firewalld.service - firewalld - dynamic firewall daemon
Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset:
enabled)
Active: inactive (dead) since Sat 2017-01-21 19:27:10 MST; 2 weeks 6 days ago
 Main PID: 722 (code=exited, status=0/SUCCESS)

Jan 21 19:18:11 schampine firewalld[722]: 2017-01-21 19:18:11 ERROR: COMMAND....
Jan 21 19:18:13 schampine firewalld[722]: 2017-01-21 19:18:13 ERROR: COMMAND....
Jan 21 19:18:13 schampine firewalld[722]: 2017-01-21 19:18:13 ERROR: COMMAND....
Jan 21 19:18:13 schampine firewalld[722]: 2017-01-21 19:18:13 ERROR: COMMAND....
Jan 21 19:18:13 schampine firewalld[722]: 2017-01-21 19:18:13 ERROR: COMMAND....
Jan 21 19:18:14 schampine firewalld[722]: 2017-01-21 19:18:14 ERROR: COMMAND....
Jan 21 19:18:14 schampine firewalld[722]: 2017-01-21 19:18:14 ERROR: COMMAND....
Jan 21 19:18:14 schampine firewalld[722]: 2017-01-21 19:18:14 ERROR: COMMAND....
Jan 21 19:27:08 schampine systemd[1]: Stopping firewalld - dynamic firewall.....
Jan 21 19:27:10 schampine systemd[1]: Stopped firewalld - dynamic firewall ...n.
Hint: Some lines were ellipsized, use -l to show in full.
```

2. If necessary start the firewalld daemon
   systemctl start firewalld

3. Set the firewalld daemon to be persistent through reboots
   systemctl enable firewalld

   verify with systemctl status firewalld again from step 1.

4. Check which zones exist
   firewall-cmd --get-zones

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

5. Checking the values within each zone
   firewall-cmd --list-all --zone=public

```
public (default, active)
  interfaces: wlp4s0
  sources:
  services: dhcpv6-client ssh
```

```
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

6. Checking the active and default zones
   firewall-cmd --get-default
   ```
   public
   ```
   firewall-cmd --get-active
   ```
   public
     interfaces: wlp4s0
   ```

   Note: this also shows which interface the zone is applied to. Multiple interfaces and zones can be applied.

   So now you know how to see the values in your firewall. Use steps 4 and 5 to check all the values of the different zones to see how they differ.

Set the firewall active and default zones

We know the zones from above, set your firewall to the different active or default zones. Default zones are the ones that will come up when the firewall is restarted.

It may be useful to perform an `ifconfig -a` and note your interfaces for the next part:

ifconfig -a | grep -i flags
```
[root@rocky ~]# ifconfig -a | grep -i flags
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
ens32: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
```

1. Changing the default zones (This is permanent over a reboot, other commands require --permanent switch)
   firewall-cmd --set-default-zone=work
   ```
   success
   ```
   firewall-cmd --get-active-zones
   ```
   work
     interfaces: wlp4s0
   ```

   Attempt to set it back to the original public zone and verify.
   Set it to one other zone, verify, then set it back to public.

2. Changing interfaces and assigning different zones (use another interface from your earlier ifconfig -a)
   firewall-cmd --change-interface=virbr0 --zone dmz
   ```
   success
   ```
   firewall-cmd --add-source 192.168.11.0/24 --zone=public

```
success
```

firewall-cmd --get-active-zones

```
dmz
  interfaces: virbr0
work
  interfaces: wlp4s0
public
  sources: 192.168.200.0/24
```

Working with ports and services

We can be even more granular with our ports and services. We can block or allow services by port number, or we can assign port numbers to a service name and then block or allow those service names.

1. List all services assigned in firewalld

   firewall-cmd --get-services

   ```
   RH-Satellite-6 amanda-client bacula bacula-client dhcp dhcpv6 dhcpv6-client dns
   freeipa-ldap freeipa-ldaps freeipa-replication ftp high-availability http https
   imaps ipp ipp-client ipsec iscsi-target 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 rsyncd samba samba-client
   smtp ssh telnet tftp tftp-client transmission-client vdsm vnc-server wbem-https
   ```

   This next part is just to show you where the service definitions exist. They are simple xml format and can easily be manipulated or changed to make new services. This would require a restart of the firewalld service to re-read this directory.

   ls /usr/lib/firewalld/services/

   ```
   amanda-client.xml         iscsi-target.xml  pop3s.xml
   bacula-client.xml         kerberos.xml      postgresql.xml
   bacula.xml                kpasswd.xml       proxy-dhcp.xml
   dhcpv6-client.xml         ldaps.xml         radius.xml
   dhcpv6.xml                ldap.xml          RH-Satellite-6.xml
   dhcp.xml                  libvirt-tls.xml   rpc-bind.xml
   dns.xml                   libvirt.xml       rsyncd.xml
   freeipa-ldaps.xml         mdns.xml          samba-client.xml
   freeipa-ldap.xml          mountd.xml        samba.xml
   freeipa-replication.xml   ms-wbt.xml        smtp.xml
   ftp.xml                   mysql.xml         ssh.xml
   high-availability.xml     nfs.xml           telnet.xml
   https.xml                 ntp.xml           tftp-client.xml
   http.xml                  openvpn.xml       tftp.xml
   imaps.xml                 pmcd.xml          transmission-client.xml
   ipp-client.xml            pmproxy.xml       vdsm.xml
   ipp.xml                   pmwebapis.xml     vnc-server.xml
   ipsec.xml                 pmwebapi.xml      wbem-https.xml
   ```

   cat /usr/lib/firewalld/services/http.xml

   ```
   <?xml version="1.0" encoding="utf-8"?>
   ```

```
<service>
  <short>WWW (HTTP)</short>
  <description>HTTP is the protocol used to serve Web pages. If you plan to make
your Web server publicly available, enable this option. This option is not
required for viewing pages locally or developing Web pages.</description>
  <port protocol="tcp" port="80"/>
</service>
```

2. Adding a service or port to a zone

firewall-cmd --set-default-zone=public
`success` #making sure we're working on public zone
firewall-cmd --list-services
`dhcpv6-client ssh` #note we only have 2 services
firewall-cmd --permanent --add-service ftp
`success` #This is only permanent if we use the --permanent switch
firewall-cmd --reload
`success`
firewall-cmd --get-default-zone
`public` #verifying we're in the right zone
firewall-cmd --list-services
`dhcpv6-client ftp ssh` #Verifying that ftp is indeed added

Alternatively, we can do almost the same thing but not use a defined service name. If I just want to allow port 1147 through for TCP traffic, it is very simple as well.

firewall-cmd --permanent --add-port=1147/tcp
`success`
firewall-cmd --reload
`success`
[root@schampine services]# firewall-cmd --list-ports
`1147/tcp`

3. Removing unwanted services or ports

To remove those values and permanently fix the configuration back we simply use remove where we had before used add:

firewall-cmd --permanent --remove-service=ftp
`success`
firewall-cmd --permanent --remove-port=1147/tcp
`success`
firewall-cmd --reload
`success`
firewall-cmd --list-services
`dhcpv6-client ssh`

firewall-cmd --list-ports

Before making any more changes I recommend running the list commands above with >> /tmp/firewall.orig on them so you have all your original values saved somewhere in case you need them.

So now take this and set up some firewalls on the interfaces of your system.
Change the default ports and services assigned to your different zones (at least 3 zones)
Read the `man firewall-cmd` command or `firewall-cmd –help` to see if there are any other userful things you should know.