Lesson 4
Network and Service Access Controls (firewalld and Fail2ban)

Contents

1. Overview of the netfilter, iptables and the firewalld

netfilter is the packet filtering framework inside the Linux 2.4.x and later kernel series. This framework enables packet filtering, network address [and port] translation (NA[P]T) and other packet mangling.
iptables is a widely used firewall tool that interfaces with the kernel's netfilter packet filtering framework. The successor of iptables, nftables, has been released since 2014 too.
firewalld is like the iptables service which talks to the netfilter framework in the kernel through the same interface, using the iptables commands. firewalld is installed and enabled in Oracle Linux 8 by default. There are two main differences between firewalld and iptables.
1. firewalld can change the firewall rules/settings during normal system operation without existing connections being lost.
2. firewalld provides both command line and GUI interfaces to the system administrators.

Ref: https://netfilter.org/

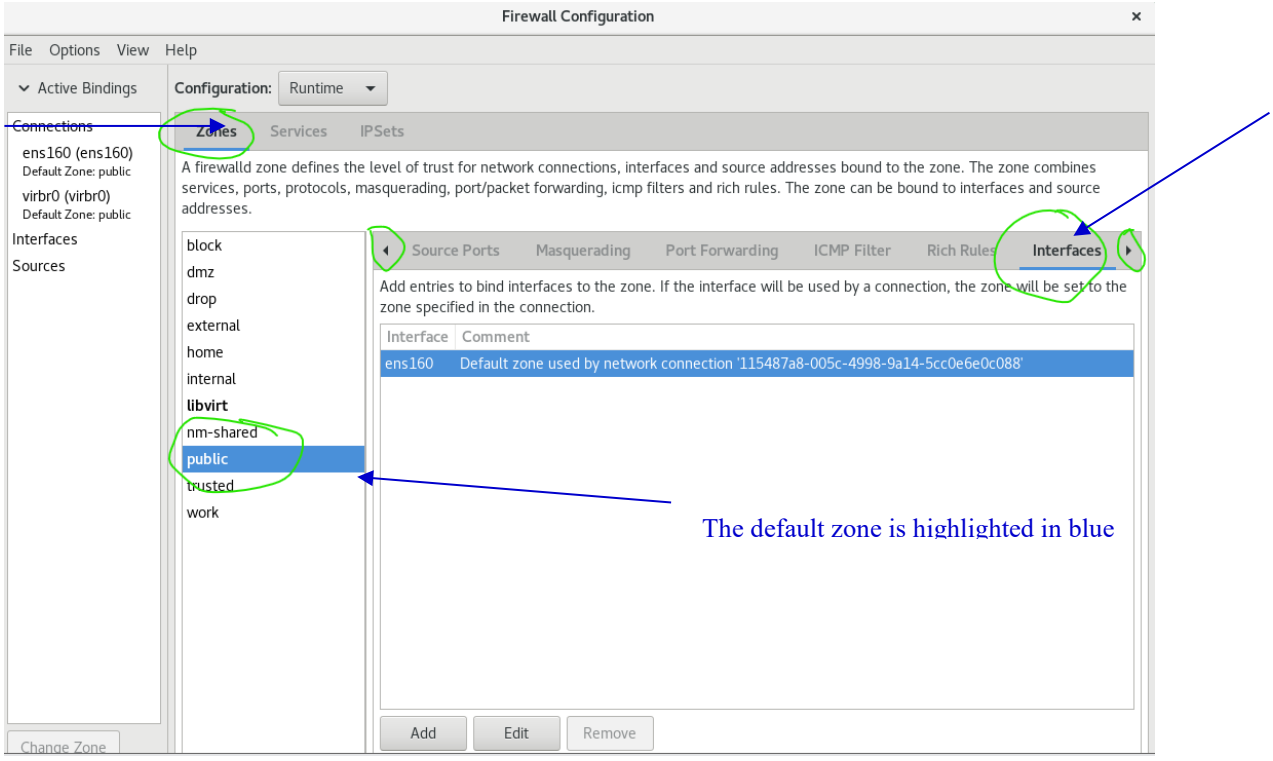
2. Exploring the zones and pre-defined services of firewalld

On any system:

1. Start the Firewalld GUI by going to Applications, Sundry, Firewall, or by running "firewall-config".

```
firewall-config
```

2. Click on the Zones tab to view the zones. Click on the Interfaces tab to view the network interfaces that are currently in each zone. (see following diagram)



3. You can also view the list of the zones and the details of them using one of the following command lines.

```
firewall-cmd  --get-zones
firewall-cmd --info-zone=<zone name>
firewall-cmd  --list-all-zones
```

4. View the default zone and its details

```
firewall-cmd  --get-default-zone
firewall-cmd --info-zone=`firewall-cmd --get-default-zone`
```

```
[root@server ~]# firewall-cmd --info-zone=`firewall-cmd --get-default-zone`
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: cockpit dhcpv6-client ftp http https ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@server ~]#
```

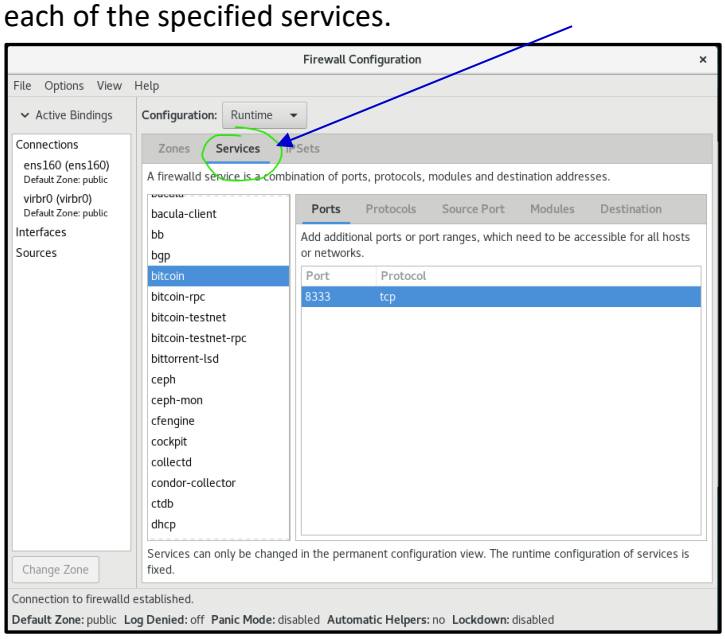
The above output reports the default zone name (i.e. public) , included interface, and the current services that are allowed to pass through the firewall.

Take note: the `firewall-cmd --get-default-zone` is enclosed in a pair of back quotes(`).

This is a simple shortcut to extract the output of a Linux command and use it as a command line argument for another Linux command.

5. In the Firewall GUI, click on the Services tab (the one beside the Zones tab) (see following diagram)

Scroll through the services listed to see which port and protocol have been specified for each of the specified services.



Your Task: Find out the ports/protocols for https, ftp , nfs and dns.

e.g. http: 80/tcp

https: _____

ftp: _____

nfs: _____

dns: _____

6. Click back on the Zones tab.

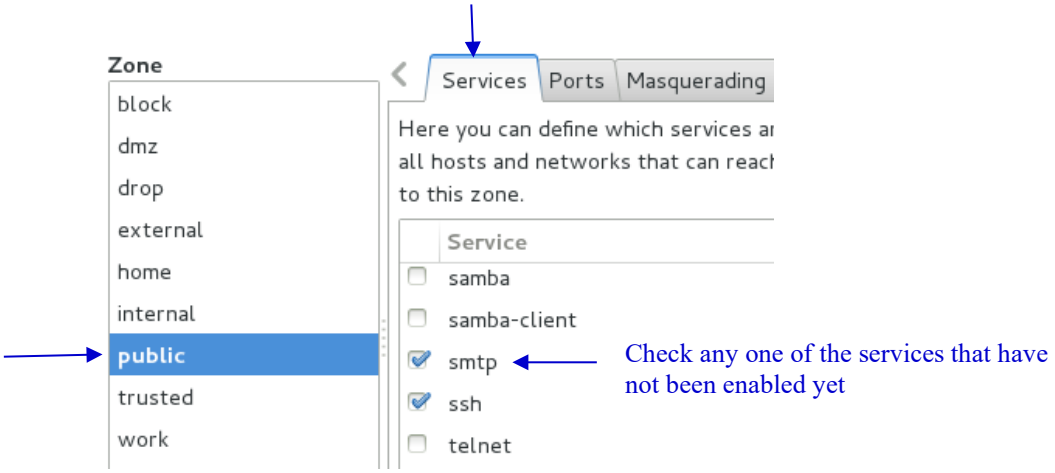
3. Adding and removing a service in firewalld for packet filtering

One of common Firewall configuration operations is managing the filtering rules. By default, all the ports are filtered (i.e. not allow to pass through the firewall) for the interfaces that belong to public zone. One simple way to allow certain ports (with their corresponding protocols) to be opened is to add in the exception rules by the network services names.

On any system:

1. In the Firewall GUI, check that for Configuration, "Runtime" is selected. (see following diagram)

2. While in the public zone, click on the Services tab to view the available services. Check any one of the services that have not been enabled yet. (see following diagram)

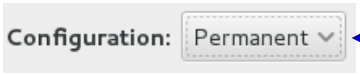


3. The change is applied immediately. List the services that are currently allowed through the firewall.

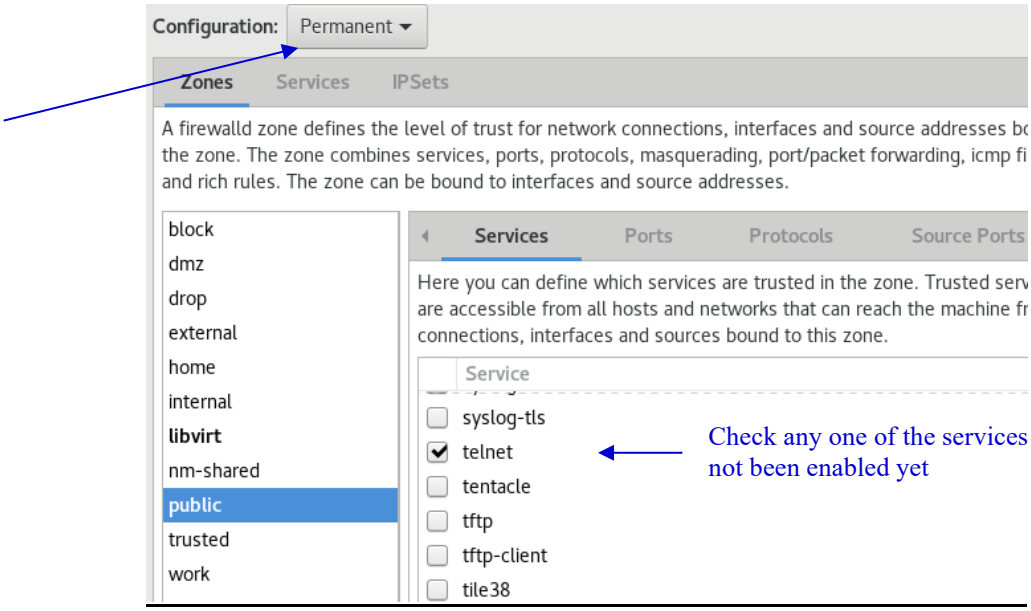
```
firewall-cmd --list-services
```

```
[root@server ~]# firewall-cmd --list-service
cockpit dhcpv6-client ftp http https smtp ssh
[root@server ~]#
```

4. In the Firewall GUI, change the Configuration to "Permanent". (see following diagram)



5. While in the public zone, click on the Services tab to view the available services. Check any one of the services that have not been enabled yet. (see following diagram)



6. As the Configuration is set to "Permanent", the change is not applied immediately, but is saved to a file, to be applied the next time the firewall is started.

   The change is saved to the public zone file. View this file to verify that the service you just added is listed.

```
cat /etc/firewalld/zones/public.xml
```

```
[root@server ~]# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas. You do not trust the other computers on
networks to not harm your computer. Only selected incoming connections are accep
ted.</description>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
  <service name="ftp"/>
  <service name="ssh"/>
  <service name="http"/>
  <service name="https"/>
  <service name="telnet"/>
</zone>
[root@server ~]#
```

7. Reload the firewall, through the GUI (Options menu->Reload Firewall).

8. Remove the service that you just added permanently, using the command line

```
firewall-cmd --permanent --zone=public
 --remove-service=telnet
```

Change to the name of the service that you previously added as permanent

9. Reload the firewall through the command line

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

Your task: Based on the firewall access control, try to control the block and allow configurations at your **server** then use your client to test and verify the operations.

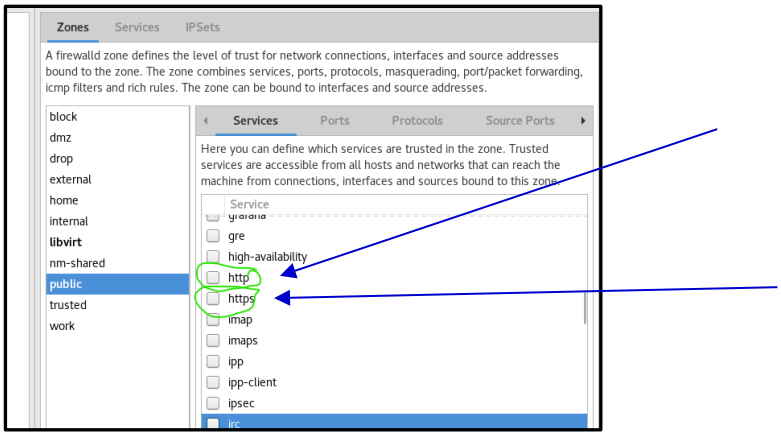
- Initial tests: verify you can access to the ftp and http services run on the server from the client.
- At the server side, block any external clients to access to the ftp service using firewall-cmd only.
- At the client side, verify that you cannot access to the ftp that runs on the server.
- At the server side, block any external clients to access to the http service using firewall-cmd only.
- At the server side, verify that you can still access to the http service that runs on the server itself.
- At the client side, verify that you cannot access to the http that runs on the server.
- At the server side, revert the firewall settings to the initial state.
- At the client side, verify you can again access to the ftp and http services run on the server.
- That's all.

4. Adding Rich Rules

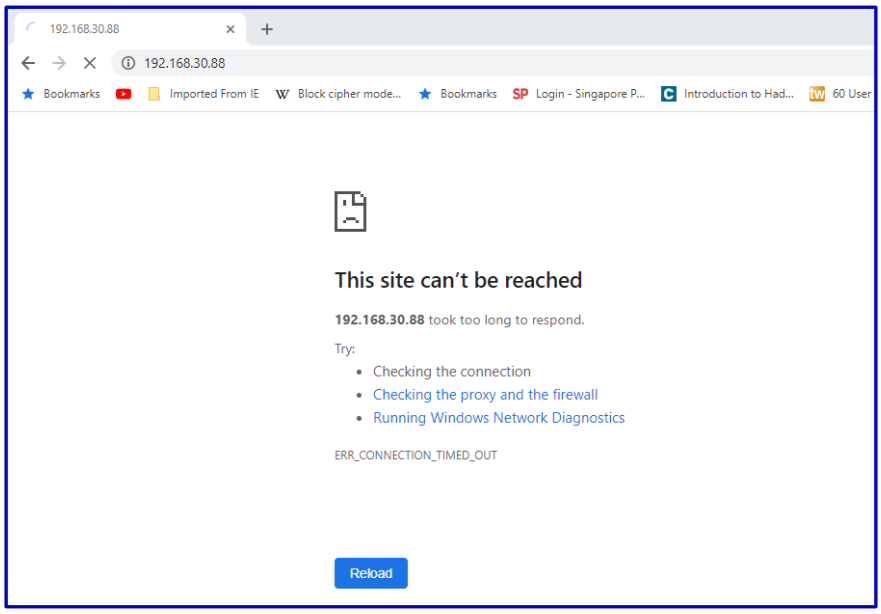
Configure the firewall using rich rules allows **more specific** control beyond the ports and protocols.

<u>On server:</u>

1. Check that the Apache Web Server is running and that you can browse the default web pages from the client system using http or https.
2. Redo the same test from your base PC (or your own notebook) browser. Ensure both of your client and your base PC can browse your server web page.
3. In the Firewall GUI, uncheck the services http and https. (see the following diagram)

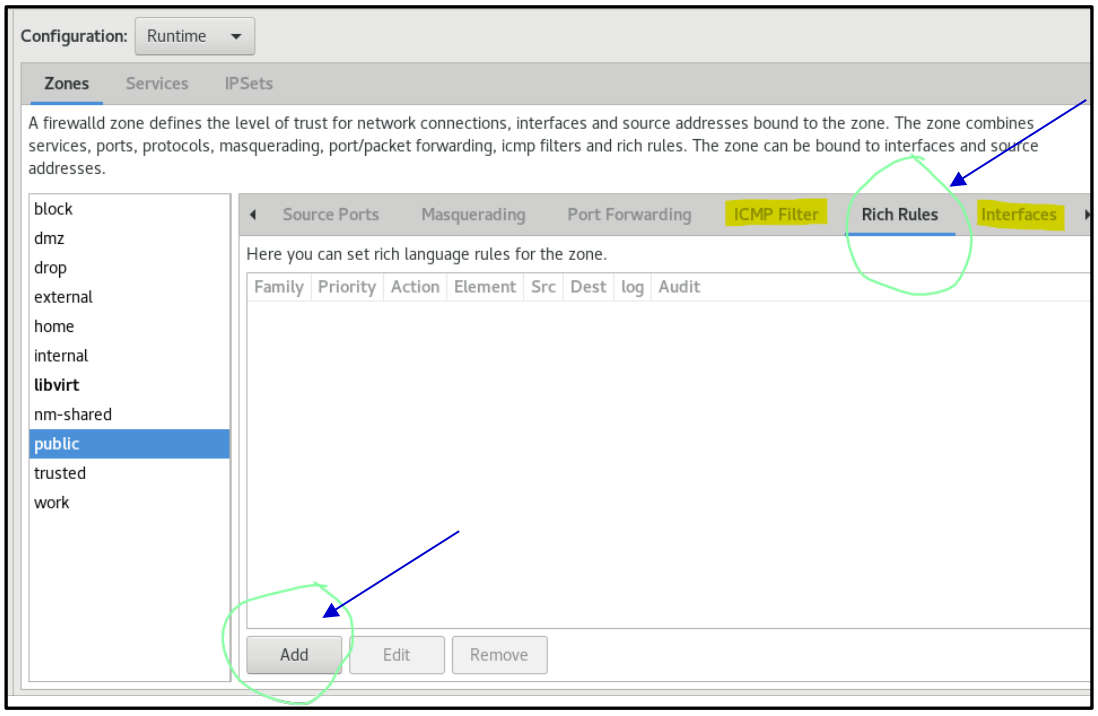


4. Now, verify that neither your client nor your base PC can browse the server web page (Try with both of the http and https protocols).

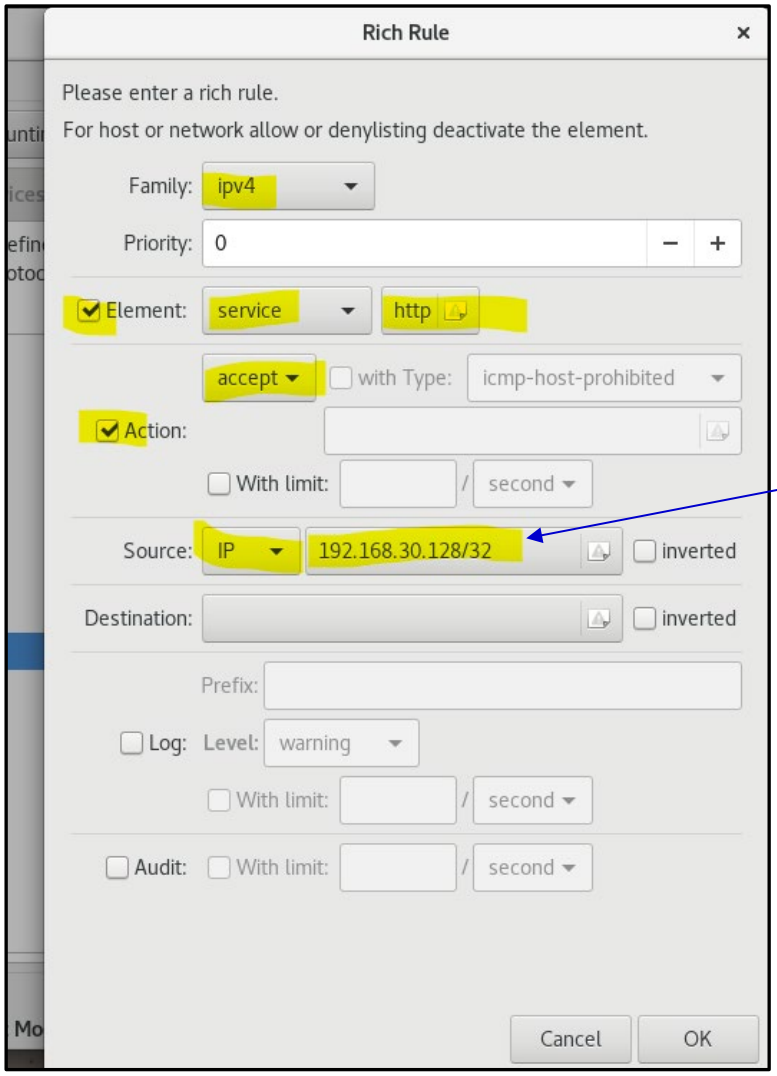


(To ensure that your browser is not displaying the cached web page. You may need to restart the browser and reload the web page to clear the cache.)

5. Use the Firewall GUI to add a rich rule (hint: look for a tab with the name 'rich rule' and press the Add button).



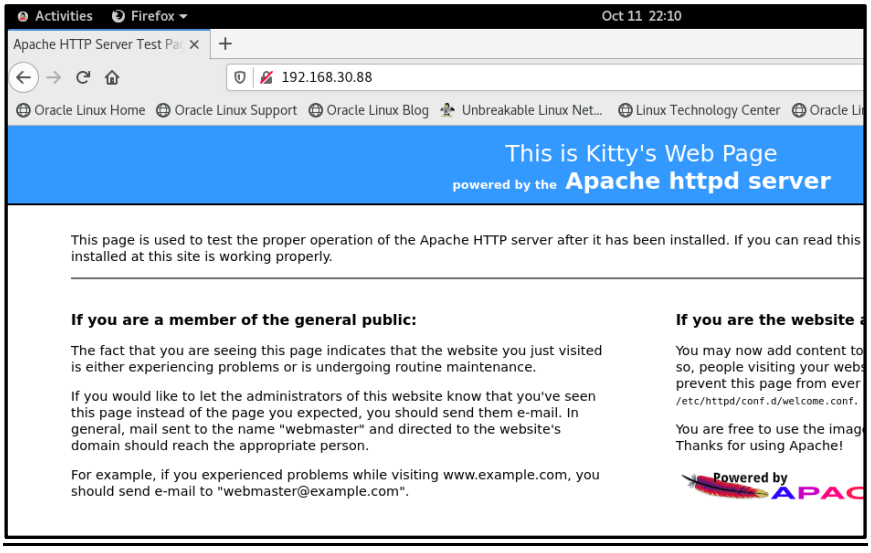
6. Define your new rich rule to accept HTTP traffic from only your client. (see following diagram for a sample of the rich rule)



Use the IP address of your client

(Take note that, you may have only added the above rich rule for runtime only.)

7. Test that your client can browse a web page from your server (using http only) but your Base PC cannot access it (Note : you may need to clear the browser cache history).



You will now try adding a rich rule using command line.

On server:

8. Add the following permanent rich rule that will allow any client in your subnet to connect to the FTP service on your server.

   Type at the command line: (Note: type in all in one single command line.)

```
firewall-cmd --permanent --zone=public
  --add-rich-rule='rule family=ipv4 service name=https
  source address=<Your LAN Segment>  accept'
```

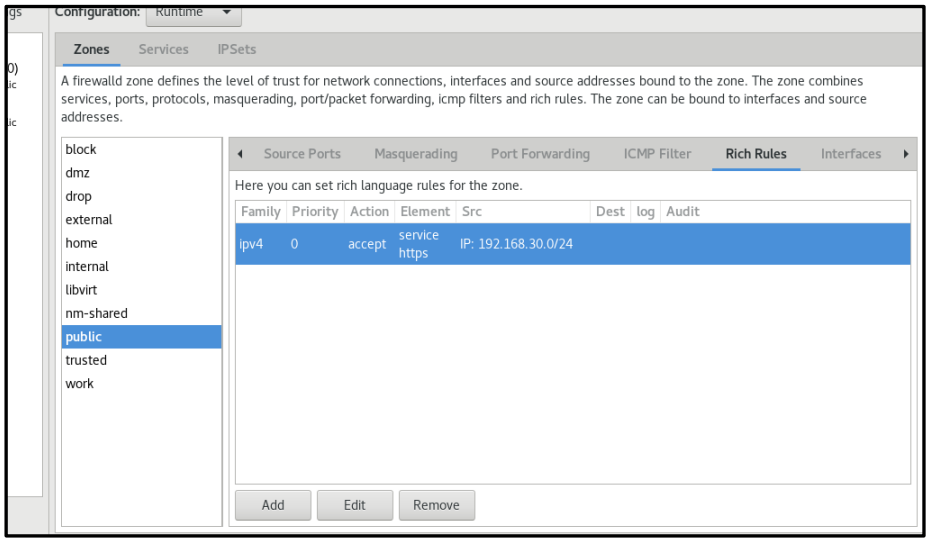
Change to your subnet segment (e.g. 192.168.30.0/24)

9. Reload the firewall. Type:

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

```
[root@server ~]# firewall-cmd --permanent --zone=public  --add-rich-rule='rule f
amily=ipv4 service name=https
  source address=192.168.30.0/24  accept'
success
[root@server ~]# firewall-cmd --reload
success
[root@server ~]#
```

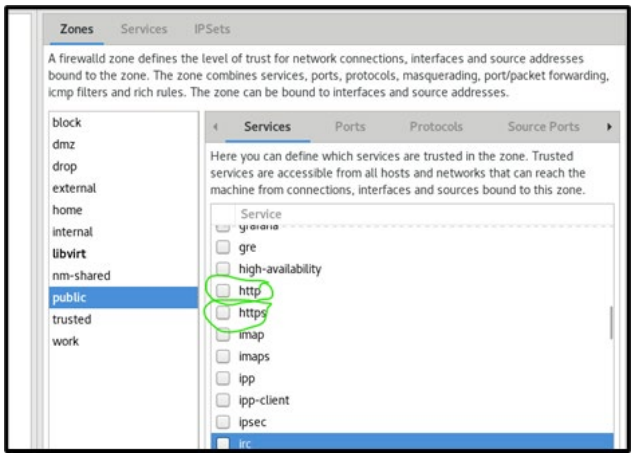
10. Using the Firewall GUI, check that the rich rule you just added is listed under the Rich Rules tab. (Note: You can refresh the rich rules tab by clicking at other tab then return to the rich rules tab.)



You may find the previous http related rule is gone and only the newly created https related rule is in the tab. (If this is the case, can you explain the reason ? Or you may ask your tutor for the answer.)

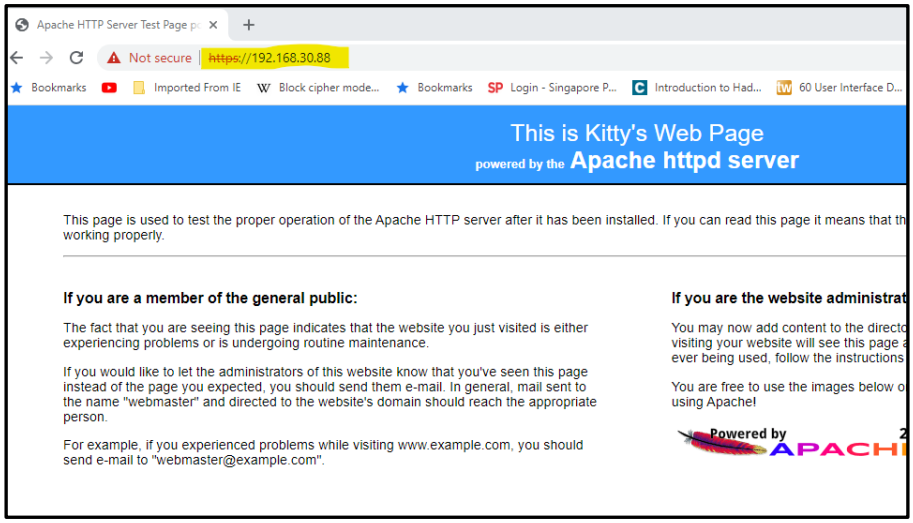
11. Switch to the service tab, ensure both of the http and https service entries are

unchecked:



On client and On your Host PC:

12. Test that you can access your server web page with https protocol now.



5. Logging Rules

You will now add logging options to the rich rules.

You may need this:
Caveat: Oracle Linux is pre-installed with gnome box (a virtualization platform) and it also enable the virbr0-nic interface (A virtual network interface) and the virbr0 device (A virtual bridge.). These two virtual devices may randomly inject annoying warning logging messages to the firewall logs. (probably due to the incomplete configuration of the gnome box). To ensure the warning messages will not confuse the following exercise. It's strongly recommended to stop and disable the virtualization service and the virbr0 related virtual devices by the following three commands at your server:

```
systemctl stop libvirtd
systemctl disable libvirtd
ip link set virbr0 down
```

```
[root@server ~]# systemctl stop libvirtd
Warning: Stopping libvirtd.service, but it can still be activated by:
  libvirtd-ro.socket
  libvirtd-admin.socket
  libvirtd.socket
[root@server ~]# systemctl disable libvirtd
Removed /etc/systemd/system/multi-user.target.wants/libvirtd.service.
Removed /etc/systemd/system/sockets.target.wants/virtlogd.socket.
Removed /etc/systemd/system/sockets.target.wants/virtlockd.socket.
Removed /etc/systemd/system/sockets.target.wants/libvirtd.socket.
Removed /etc/systemd/system/sockets.target.wants/libvirtd-ro.socket.
[root@server ~]# ip link set virbr0 down
[root@server ~]# nmcli device
DEVICE      TYPE      STATE      CONNECTION
ens160      ethernet  connected  ens160
virbr0      bridge    unmanaged  --
lo          loopback  unmanaged  --
virbr0-nic  tun       unmanaged  --
[root@server ~]#
```

(Before reboot, the virbr0 device is still shown in the output of nmcli device command.)
You also need to reboot the server VM.

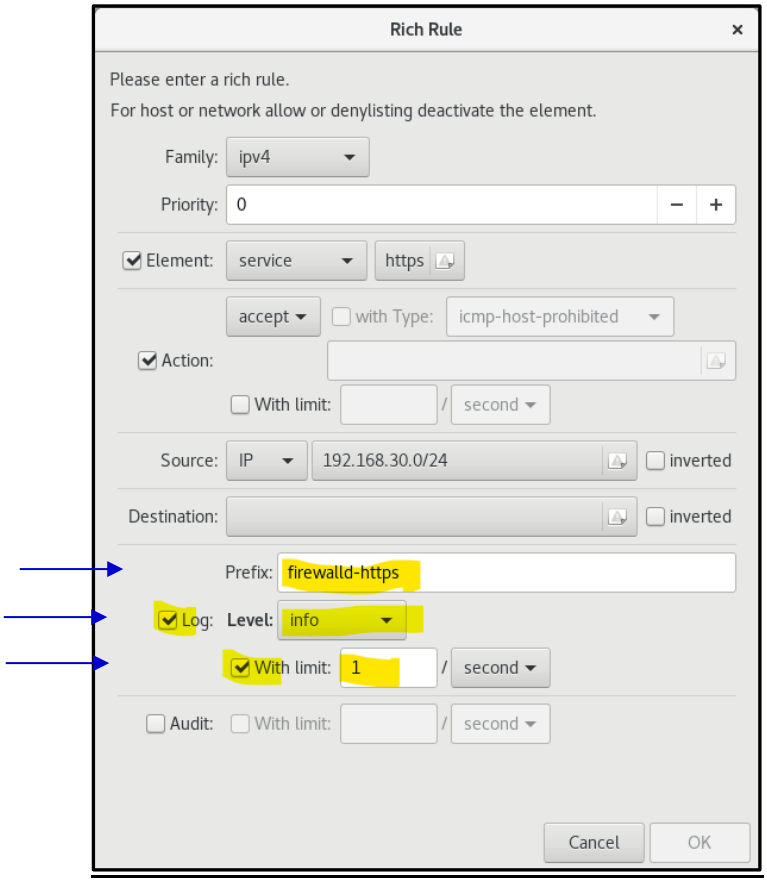
On server:

1. Verify you only have the loopback (lo) and ens160 devices :

```
nmcli device
```

```
[root@server ~]# nmcli device
DEVICE  TYPE      STATE      CONNECTION
ens160  ethernet  connected  ens160
lo      loopback  unmanaged  --
[root@server ~]#
```

2. In the Firewall GUI, under Rich Rules, **edit** the rich rule that you added for the https service.
3. Enable Log, and for the Level, choose **info**. (see following diagram)

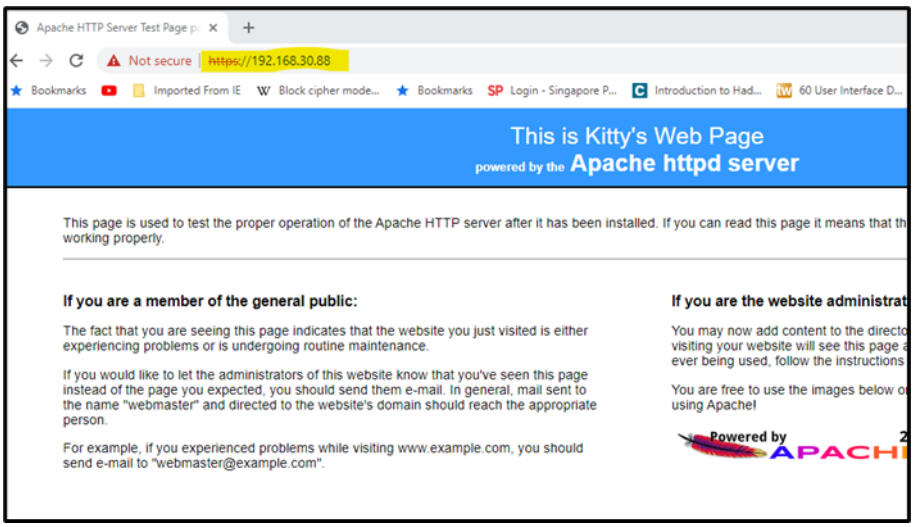


(Limit field set to 1 / second is to limit up to maximum 1 log per second.)

<u>On client or On your host PC:</u>

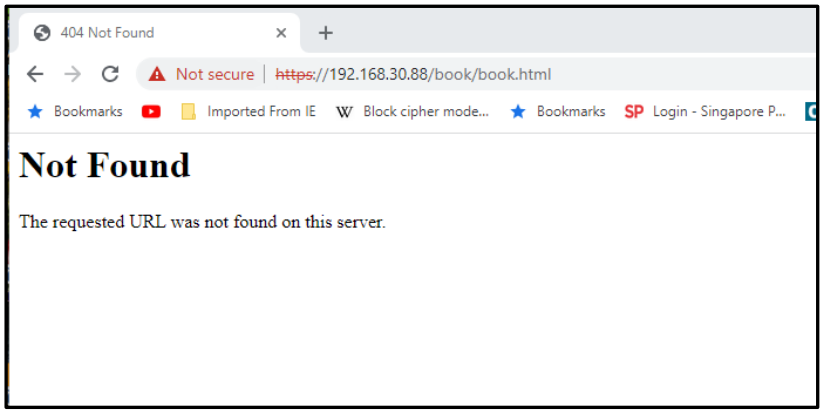
4. Use a web browser to access to a few different web pages of server.

   And include some invalid path. All these browsing attempts will trigger the logging operations.



(I have reloaded the default page.)

(I have visited the test_wsgi page.)



(I have visited the /book/book.html page, but it is not found.)

On server:

5. View the end of /var/log/messages to see the logged packet. Note the information captured from the packet (eg source address, source port, dest address, dest port, SYN flag of the packet).

```
[root@server log]# tail /var/log/messages
Oct 11 23:20:25 server kernel: IN=ens160 OUT= MAC=00:0c:29:9a:96:ff:00:50:56:c0:00:08:08:00 SRC=192.168.30.1 DST=192.168.30.88 L
EN=52 TOS=0x00 PREC=0x00 TTL=128 ID=25207 DF PROTO=TCP SPT=60606 DPT=443 WINDOW=64240 RES=0x00 SYN URGP=0
Oct 11 23:22:33 server kernel: firewalld-httpsIN=ens160 OUT= MAC=00:0c:29:9a:96:ff:00:50:56:c0:00:08:08:00 SRC=192.168.30.1 DST=
192.168.30.88 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=25219 DF PROTO=TCP SPT=62163 DPT=443 WINDOW=64240 RES=0x00 SYN URGP=0
Oct 11 23:22:33 server kernel: firewalld-httpsIN=ens160 OUT= MAC=00:0c:29:9a:96:ff:00:50:56:c0:00:08:08:00 SRC=192.168.30.1 DST=
192.168.30.88 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=25222 DF PROTO=TCP SPT=64400 DPT=443 WINDOW=64240 RES=0x00 SYN URGP=0
Oct 11 23:22:33 server kernel: firewalld-httpsIN=ens160 OUT= MAC=00:0c:29:9a:96:ff:00:50:56:c0:00:08:08:00 SRC=192.168.30.1 DST=
192.168.30.88 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=25233 DF PROTO=TCP SPT=59686 DPT=443 WINDOW=64240 RES=0x00 SYN URGP=0
Oct 11 23:22:50 server kernel: firewalld-httpsIN=ens160 OUT= MAC=00:0c:29:9a:96:ff:00:50:56:c0:00:08:08:00 SRC=192.168.30.1 DST=
192.168.30.88 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=25243 DF PROTO=TCP SPT=57329 DPT=443 WINDOW=64240 RES=0x00 SYN URGP=0
Oct 11 23:22:50 server kernel: firewalld-httpsIN=ens160 OUT= MAC=00:0c:29:9a:96:ff:00:50:56:c0:00:08:08:00 SRC=192.168.30.1 DST=
192.168.30.88 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=25245 DF PROTO=TCP SPT=53119 DPT=443 WINDOW=64240 RES=0x00 SYN URGP=0
Oct 11 23:22:50 server kernel: firewalld-httpsIN=ens160 OUT= MAC=00:0c:29:9a:96:ff:00:50:56:c0:00:08:08:00 SRC=192.168.30.1 DST=
192.168.30.88 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=25257 DF PROTO=TCP SPT=58062 DPT=443 WINDOW=64240 RES=0x00 SYN URGP=0
Oct 11 23:22:55 server kernel: firewalld-httpsIN=ens160 OUT= MAC=00:0c:29:9a:96:ff:00:50:56:c0:00:08:08:00 SRC=192.168.30.1 DST=
192.168.30.88 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=25268 DF PROTO=TCP SPT=57804 DPT=443 WINDOW=64240 RES=0x00 SYN URGP=0
Oct 11 23:22:55 server kernel: firewalld-httpsIN=ens160 OUT= MAC=00:0c:29:9a:96:ff:00:50:56:c0:00:08:08:00 SRC=192.168.30.1 DST=
192.168.30.88 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=25269 DF PROTO=TCP SPT=65125 DPT=443 WINDOW=64240 RES=0x00 SYN URGP=0
Oct 11 23:22:55 server kernel: firewalld-httpsIN=ens160 OUT= MAC=00:0c:29:9a:96:ff:00:50:56:c0:00:08:08:00 SRC=192.168.30.1 DST=
192.168.30.88 LEN=52 TOS=0x00 PREC=0x00 TTL=128 ID=25280 DF PROTO=TCP SPT=65135 DPT=443 WINDOW=64240 RES=0x00 SYN URGP=0
[root@server log]#
```

(/var/log/messages collects many log entries from the kernel directly. The prefix we have set earlier in the rich rules will help us to locate the firewalld related log entries.)

Take note that, the above is a firewall log (not the httpd server access log). It is generated at the moment the firewall has allowed a https connection. **Whether the connection leads to a valid page or invalid page is not known at this moment.**

If you check out the /var/log/httpd/ssl_access_log you will find the higher level and useful information:

```
[root@server httpd]# tail /var/log/httpd/ssl_access_log
192.168.30.1 - bob [11/Oct/2021:23:04:25 +0800] "GET /test_wsgi HTTP/1.1" 200 292
192.168.30.1 - - [11/Oct/2021:23:04:36 +0800] "GET /book/book.html HTTP/1.1" 404 196
192.168.30.1 - - [11/Oct/2021:23:04:39 +0800] "GET /book/book.html HTTP/1.1" 404 196
192.168.30.1 - - [11/Oct/2021:23:06:32 +0800] "GET /test_wsgi HTTP/1.1" 401 381
192.168.30.1 - bob [11/Oct/2021:23:06:38 +0800] "GET /test_wsgi HTTP/1.1" 200 292
192.168.30.1 - - [11/Oct/2021:23:06:39 +0800] "GET /favicon.ico HTTP/1.1" 404 196
192.168.30.1 - - [11/Oct/2021:23:06:45 +0800] "GET / HTTP/1.1" 200 3545
192.168.30.1 - - [11/Oct/2021:23:06:45 +0800] "GET /icons/apache_pb2.gif HTTP/1.1" 200 4234
192.168.30.1 - - [11/Oct/2021:23:06:52 +0800] "GET /book/book.html HTTP/1.1" 404 196
192.168.30.1 - - [11/Oct/2021:23:06:55 +0800] "GET /book/book.html HTTP/1.1" 404 196
[root@server httpd]#
```

(At the httpd access log, you can tell whether the https request is successful or not.)

Remove all the rich rules from the firewalld after you have completed the exercises.

6. Limiting frequency of packets to discourage port scanning

On server:

1. Remove all the rich rules you have set in the earlier exercise.
2. Allow http and ftp service.
3. Reload the firewall.

On client:

4. Login as root. Run a SYN scan against your server using nmap. Install the nmap package if it is not yet installed. The following command will scan port 1 to port 1000 of the target server.

```
nmap -sS <serverIP>
```

Depending on the configuration of the firewall on your server, you should see that some ports are opened.

To be able to compare the current and later port scanning result, rerun the nmap command within the time command:

```
time nmap -sS <serverIP>
```

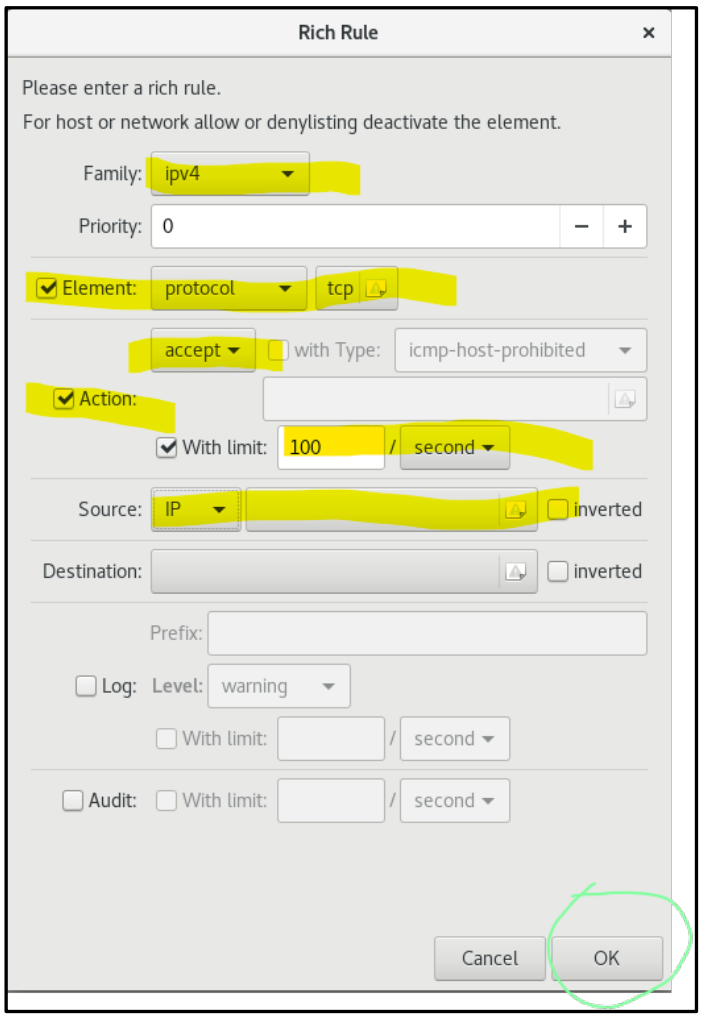
```
[root@client ~]# time nmap -sS 192.168.30.88
Starting Nmap 7.70 ( https://nmap.org ) at 2022-10-10 14:51 +08
Nmap scan report for 192.168.30.88
Host is up (0.00039s latency).
Not shown: 994 filtered ports
PORT     STATE  SERVICE
21/tcp   open   ftp
22/tcp   open   ssh
23/tcp   closed telnet
80/tcp   open   http
443/tcp  open   https
9090/tcp open   zeus-admin
MAC Address: 00:0C:29:0A:99:A9 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 14.29 seconds

real    0m14.304s
user    0m0.007s
sys     0m0.364s
[root@client ~]#
```

Based on the sample, it took about 14+ seconds to complete the SYNC port scan at this point. (You may run a few times to take the average result.)

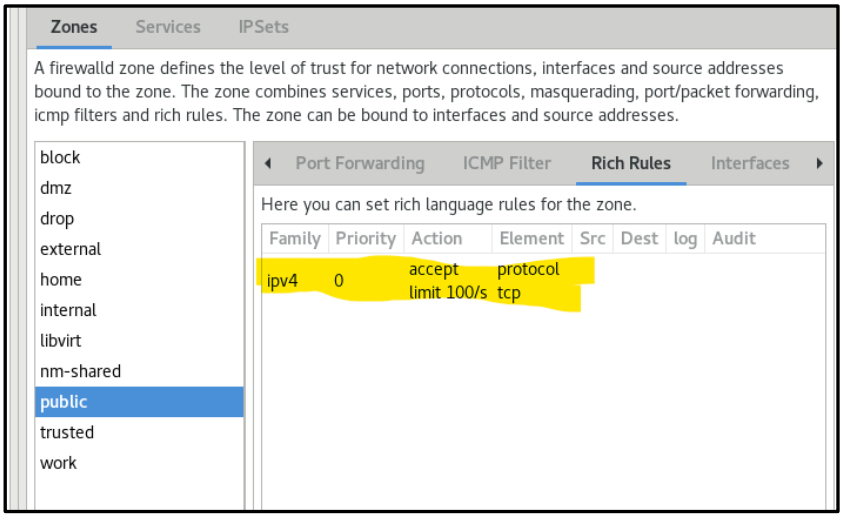
Also take note that, nmap is smart enough to figure out the other 994 ports are filtered. (ie. blocked by a firewall.)

On server:

5. In the Firewall GUI, add a Rich Rule that will limit 100 new TCP connections per second (see following diagram)



Note : By adding this rule, you are allowing all TCP connections to your server.



The above is the content of the Rich Rules Tab after applying the rule.

On client:

6. Run the SYN scan against your server again. This time, the scan takes a longer time to complete. It is because nmap has done many 'retry' attempts. At the same time, there are a lot of filtered ports (that nmap cannot determine their states) being reported. These confusing results are causing by the connection limits* set on the firewall.

```
[root@client ~]# time nmap -sS 192.168.30.88
Starting Nmap 7.70 ( https://nmap.org ) at 2022-10-10 14:57 +08
Nmap scan report for 192.168.30.88
Host is up (0.00059s latency).
Not shown: 984 closed ports
PORT      STATE    SERVICE
21/tcp    open     ftp
22/tcp    open     ssh
53/tcp    filtered domain
80/tcp    open     http
111/tcp   open     rpcbind
406/tcp   filtered imsp
443/tcp   open     https
445/tcp   filtered microsoft-ds
554/tcp   filtered rtsp
587/tcp   filtered submission
993/tcp   filtered imaps
995/tcp   filtered pop3s
1723/tcp  filtered pptp
2099/tcp  filtered h2250-annex-g
9090/tcp  open     zeus-admin
16016/tcp filtered unknown
MAC Address: 00:0C:29:0A:99:A9 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 97.43 seconds

real    1m37.439s
user    0m0.019s
sys     0m0.411s
[root@client ~]#
```

The total scanning time also increased to 1 minutes 37 seconds!

With this+, we can 'discourage' port scanning activities.

*Note on the SYNC packet responses:
In terms of network port scanning, there can be in three states: open, closed or not

reachable (no answer.).
When nmap reports its scanning result, it will base on these states.

There are 3 basic outcomes for a SYNC packet request of a connection:

a - The connection is made successfully; the receiving host return an ACK packet to the requesting client. Based on this, nmap reports the port is 'open' and there is no firewall to block this connection.

b - The connection attempt is failed because the receiving host returns an RST packet. Based on this, nmap reports the port is 'closed' and there is no service listening on the destinated port. It also implies there is no firewall setup between the sending host and the target host. (or, the firewall returns the RST packet while it is blocking the access.)

c - The connection attempt is also failed since there is no ACK nor RST packet returns. The receiving host just drops (or discard) the incoming packet (Due to the firewall denial and not return any response). This case will cause the request host to retry a few times until a timeout to determine that the port is not reachable (Due to firewall blockage or network outage). nmap will report this port is filtered. (Implies there is a firewall in between the client and the server.)

For the firewalld, it always returns an RST packet when a remote client is attempting to connect to a blocked port. It is to simulate the port is closed.

When we set the TCP is to have a limited acceptance rate (e.g. Maximum 100 packets per second.) Some of the connection attempts will be dropped by the firewalld and it will not be replied. It will trigger nmap to retry a few times. In this case, nmap may list the port as filtered (if all retries failed) or closed (if the retry get an answer.).

+Note on the discouragement:
There are two discourage factors:
a - The scanning time taken is much longer than the normal expected one. The above example is only applying the port scan on one target host. Imagine if the adversary is planning on scanning a network.
b - The misleading result may put off the interests for the adversary for further attempts.

7. Let's verify that firewalld is returning a RST packet for blocked port, type:

```
nmap -sS -p 53 <serverIP>
```

To do a SYNC scan to only one single port, 53 (Which was reported as filtered earlier)

```
[root@client ~]# nmap -sS  -p 53 192.168.30.88
Starting Nmap 7.70 ( https://nmap.org ) at 2022-10-11 16:49 +08
Nmap scan report for 192.168.30.88
Host is up (0.00039s latency).

PORT   STATE  SERVICE
53/tcp closed domain
MAC Address: 00:0C:29:0A:99:A9 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.71 seconds
[root@client ~]#
```

Since we are only scanning one specific port, the packet limitation measure will not affect the scan. Thus, nmap will get the RST packet from the firewalld and reports that port 53 is closed.

On server:

8. Stop the firewalld. Type:

```
systemctl stop firewalld
```

On client:

9. Run the SYN scan against your server again to find out how long (or short) it takes to complete the scanner when there is no firewall in place.

```
[root@client ~]# time nmap -sS 192.168.30.88
Starting Nmap 7.70 ( https://nmap.org ) at 2022-10-10 15:10 +08
Nmap scan report for 192.168.30.88
Host is up (0.00015s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
111/tcp  open  rpcbind
443/tcp  open  https
9090/tcp open  zeus-admin
MAC Address: 00:0C:29:0A:99:A9 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.77 seconds

real    0m1.787s
user    0m0.014s
sys     0m0.097s
[root@client ~]#
```

There are 6 open ports, including 111/tcp . nmap also reported 994 other ports are 'closed'.

(1000 - 6 = 994.) The time to take the nmap to complete the scan is much faster !.

On server:

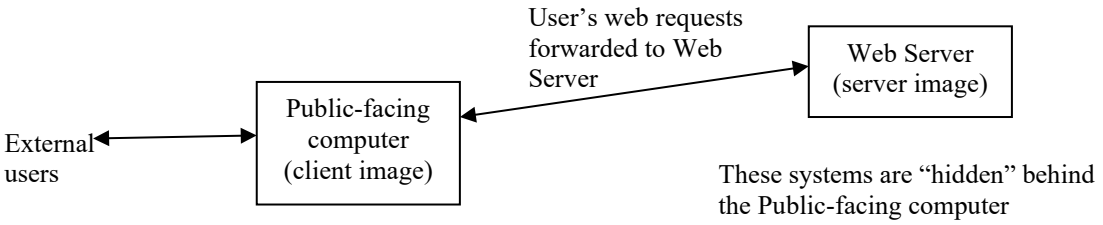
10. Restart the firewalled and clear the Rich Rule you have added earlier so that the rest of the practical is not affected

7. Reverse Proxy via Network Address Translation (NAT)

Instead of apply NAT to allow internal users to share a public IP address for outgoing network operations.

You will now implement Network Address Translation (NAT) through firewalld to achieve a 'reverse proxy' feature. By deploying reverse proxy, you can reduce the attack surface of your real server by not letting it to face public access directly.

The Web Server is installed on your server image, which will be "hidden" from the outside world behind your client image. External users only connect to the client, which acts as the public interface. The client will forward the user's requests to other systems such as your server image.



On server:

1. Start the Apache server if it is not running yet.

On client:

2. Stop the firewall on the client.

```
systemctl stop firewalld
```

3. Make sure the Apache server is not running on the client.

```
systemctl stop httpd
systemctl disable httpd
```

or

```
systemctl disable --now httpd
```

(disable --now will disable and stop the targeted service)

4. Ensure there is no program listening on port 80.

```
netstat -tunpa | grep 'LISTEN'
```

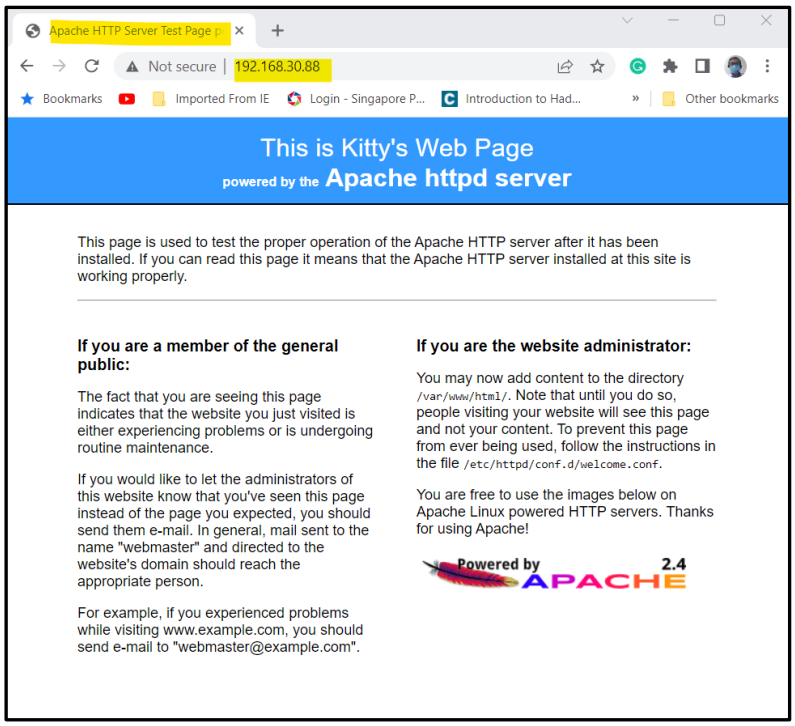
and/or

```
netstat -tunap | grep 'LISTEN'  | grep 80
```

```
File  Edit  View  Search  Terminal  Help
[root@client ~]# netstat -tunap | grep LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      1/systemd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1009/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      1011/cupsd
tcp6       0      0 :::111                  :::*                    LISTEN      1/systemd
tcp6       0      0 :::22                   :::*                    LISTEN      1009/sshd
tcp6       0      0 ::1:631                 :::*                    LISTEN      1011/cupsd
[root@client ~]# netstat -tunap | grep LISTEN | grep 80
[root@client ~]#
[root@client ~]#
```

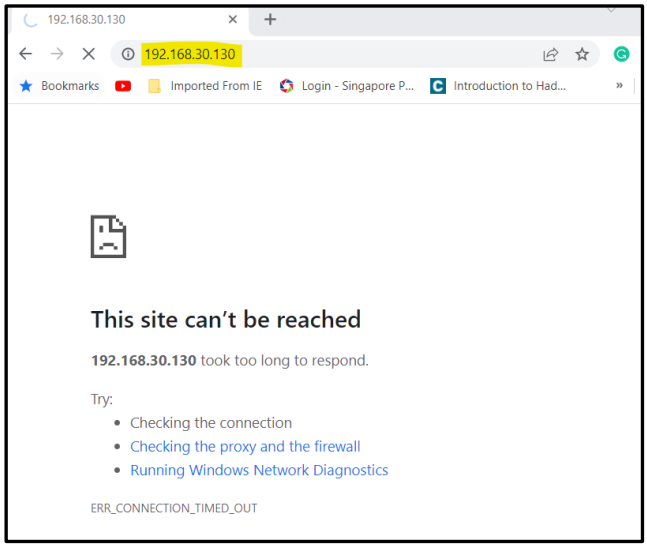
On Base PC:

5. Browse to http://<*server_ip*> to see the web page of your server image. (you may need to adjust the firewall on your server image to allow the Base PC to connect to the Apache web server on the server image)



(Note: The server IP of the VM in the above is 192.168.30.88.)

6. Browse to http://<*client_ip*>. You should get an error message as there is no web server running on your client image.



(Note: The client IP of the VM in the above is 192.168.30.130.)

On server:

7. Check the latest content (log entires) of the /var/log/httpd/access_log. You should be able to find the log entries that reflect the http requests from your Base PC to the apache server. (hint: tail /var/log/httpd/access_log)

```
[root@server ~]# tail /var/log/httpd/access_log
192.168.30.1 - - [11/Oct/2022:16:54:37 +0800] "GET / HTTP/1.1" 200 3545 "-" "Mozilla/5.0 (W
indows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/
537.36"
192.168.30.1 - - [11/Oct/2022:16:54:38 +0800] "GET /icons/apache_pb2.gif HTTP/1.1" 200 4234
 "http://192.168.30.88/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHT
ML, like Gecko) Chrome/105.0.0.0 Safari/537.36"
192.168.30.1 - - [11/Oct/2022:16:54:38 +0800] "GET /favicon.ico HTTP/1.1" 404 196 "http://1
92.168.30.88/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/105.0.0.0 Safari/537.36"
[root@server ~]#
```

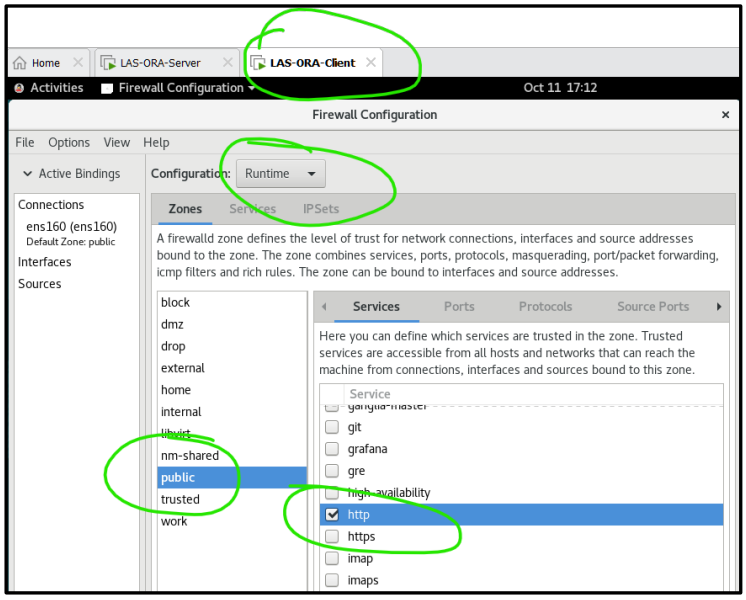
In the above, 192.168.30.1 is the ip address of the Base PC (This is the VMnet default settings.). You should verify the date/time of the log entry too.

On client:

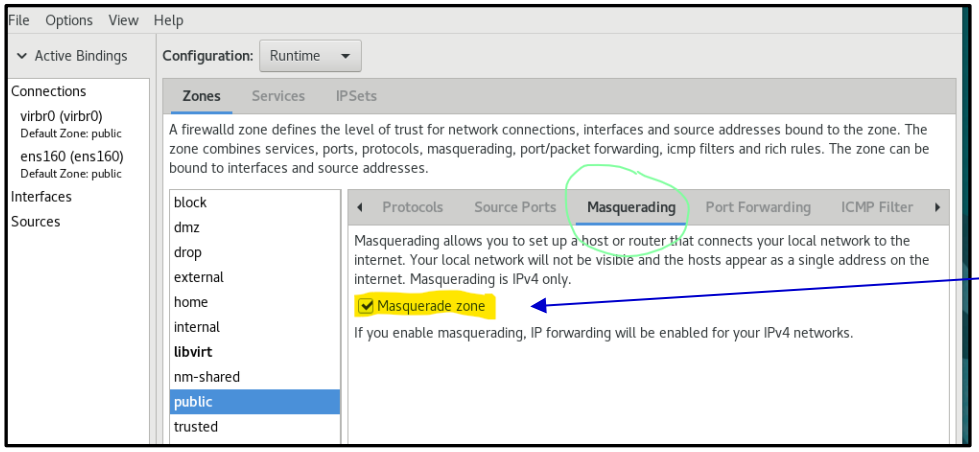
8. Enable the firewall on the client.

```
systemctl start firewalld
```

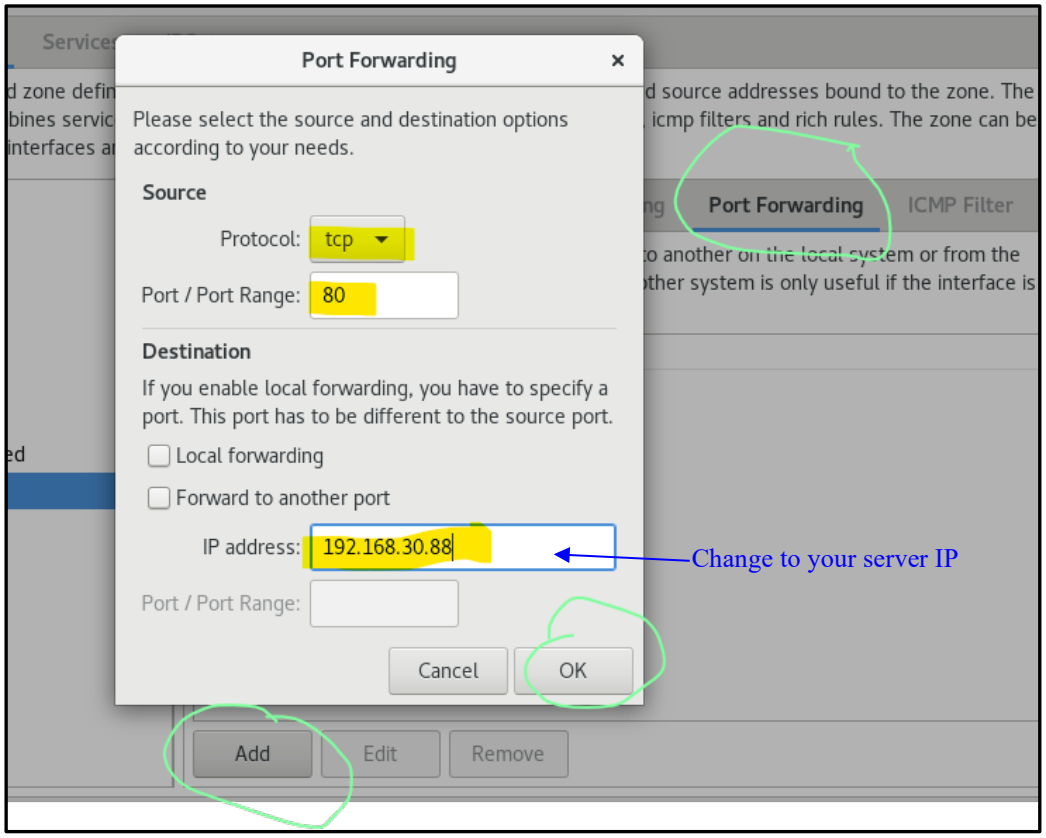
9. In the Firewall GUI, enable http service port (to allow incoming http request, even your client is not having a web server running).

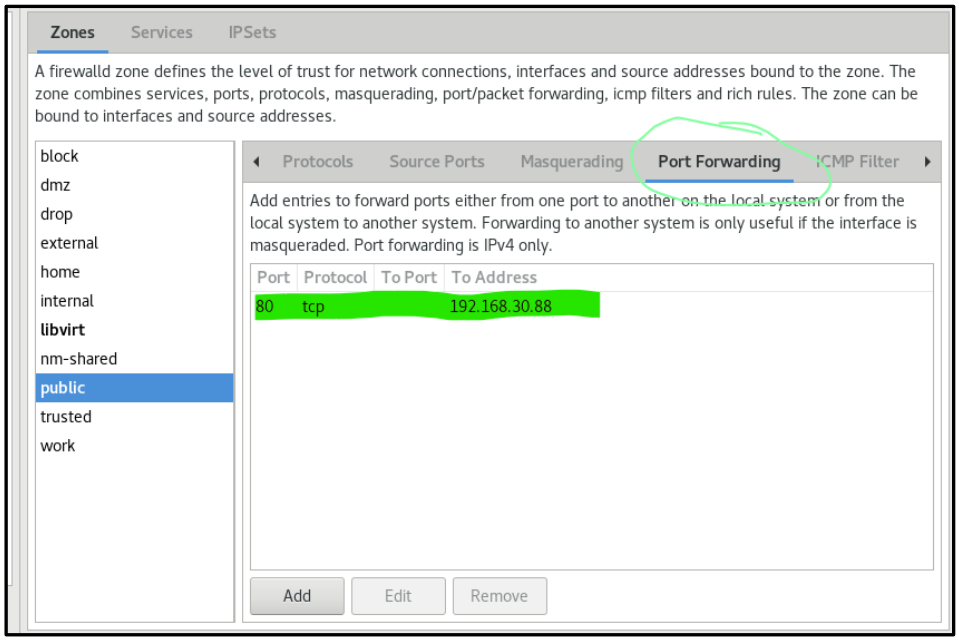


10. In the Firewall GUI, enable IP masquerading from the Masquerading tab, and check Enable Masquerade. (See the following diagram). This means all outgoing packets will be modified to have the same source IP as the client network interface card.



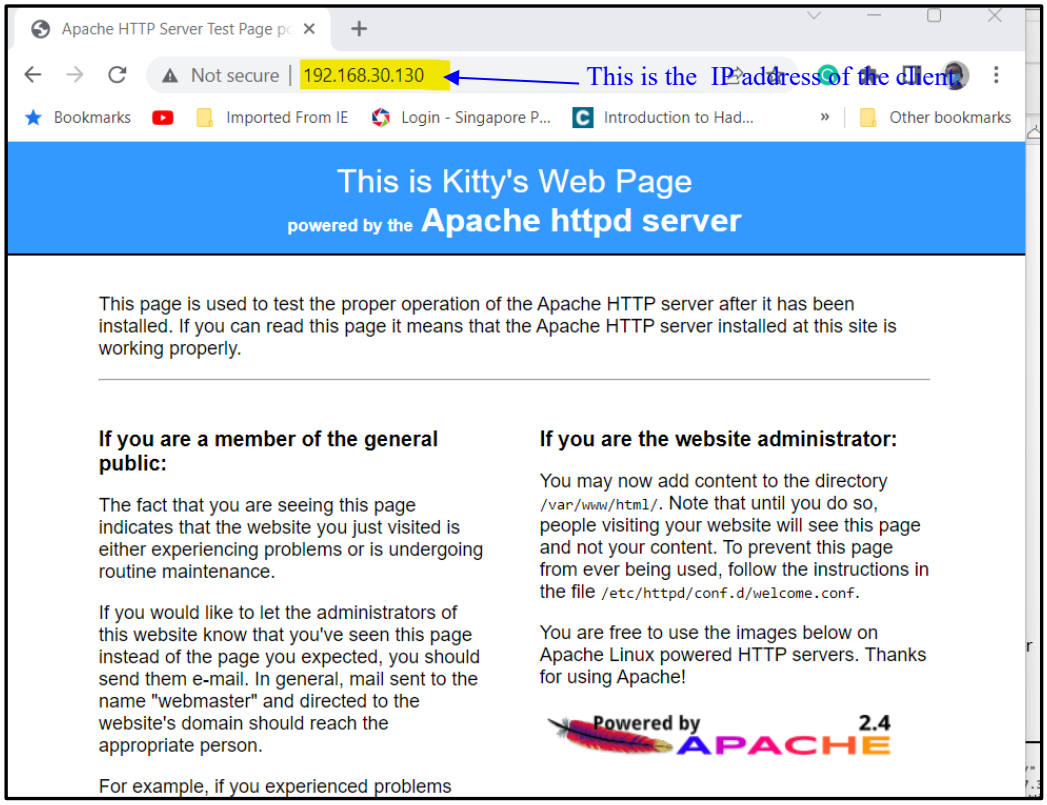
11. Next, move to the next tab, the Port Forwarding tab, and add the following entry to **forward** all incoming packets going to the Port 80 on your client to the Port 80 on your server (see following diagram)





<u>On Base PC:</u>

12. Browse to http://*client_ip*. Even though you are connecting to your client IP, your request has been forwarded to the server so you will see the web page of your server image. The server which receives the request would think the request is sent by the client (instead of from your base PC).



On server:

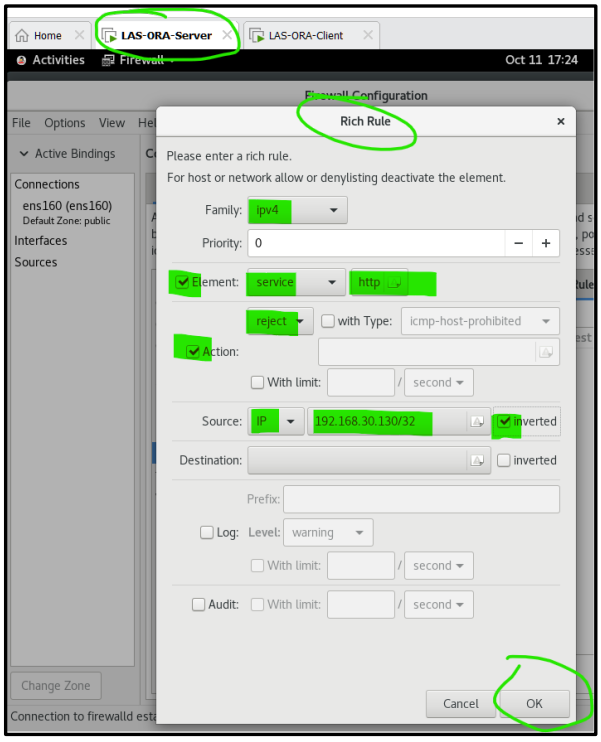
13. You can verify the above by examining the /var/log/httpd/access_log

You will find the access entries with the source IP of your client (instead of the Base PC) IP.

Moreover, the target destination addresses are also set t your client IP because the browser was really trying to access to the Client IP. It was due to the port mapping feature of the firewalld running on the client, mapped the request to your Server.

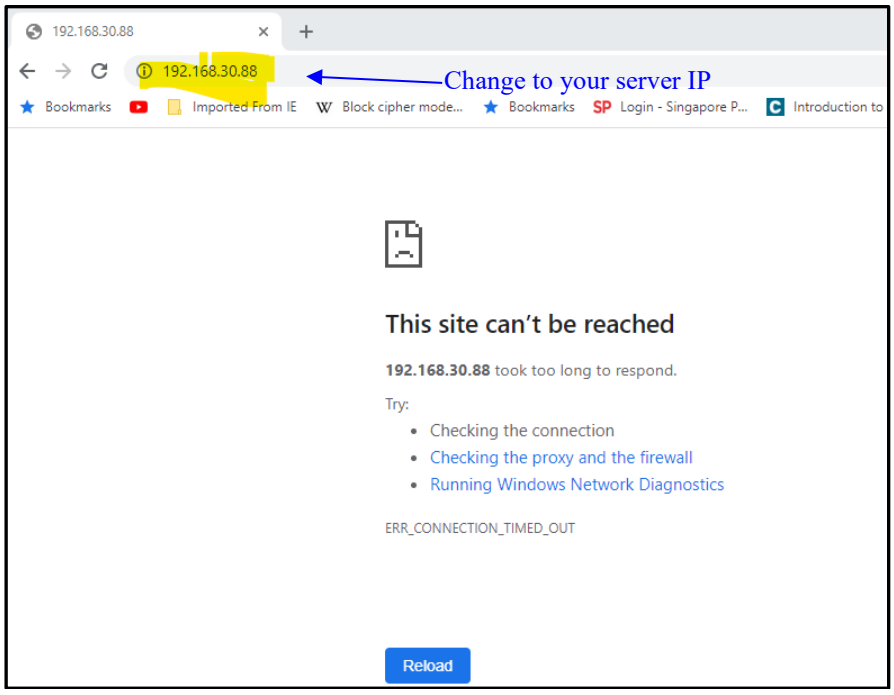
```
[root@server ~]# tail -5 /var/log/httpd/access_log
192.168.30.1 - - [11/Oct/2022:16:54:38 +0800] "GET /icons/apache_pb2.gif HTTP/1.1" 200 4234
 "http://192.168.30.88/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHT
ML, like Gecko) Chrome/105.0.0.0 Safari/537.36"
192.168.30.1 - - [11/Oct/2022:16:54:38 +0800] "GET /favicon.ico HTTP/1.1" 404 196 "http://1
92.168.30.88/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/105.0.0.0 Safari/537.36"
192.168.30.130 - - [11/Oct/2022:17:15:23 +0800] "GET / HTTP/1.1" 200 3545 "-" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safar
i/537.36"
192.168.30.130 - - [11/Oct/2022:17:15:23 +0800] "GET /icons/apache_pb2.gif HTTP/1.1" 200 42
34 "http://192.168.30.130/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36"
192.168.30.130 - - [11/Oct/2022:17:15:23 +0800] "GET /favicon.ico HTTP/1.1" 404 196 "http:/
/192.168.30.130/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik
e Gecko) Chrome/105.0.0.0 Safari/537.36"
[root@server ~]#
```

14. On your Server, change the firewall setting, to add a rich rule that rejects http access from everyone except your client.



(In this case, the client VM ip is 192.168.30.130/32. [The /32 denotes the network mask. It is optional in this case.])

15. On your base PC, verify that, you cannot access to your server web page directly. The only way to access to the web page is via the reverse proxy that set at your client VM.



Note: From the above exercise, you should learn that the reject action of a Rich Rule **overrides** the normal allow rule.

For your reference, we can run:

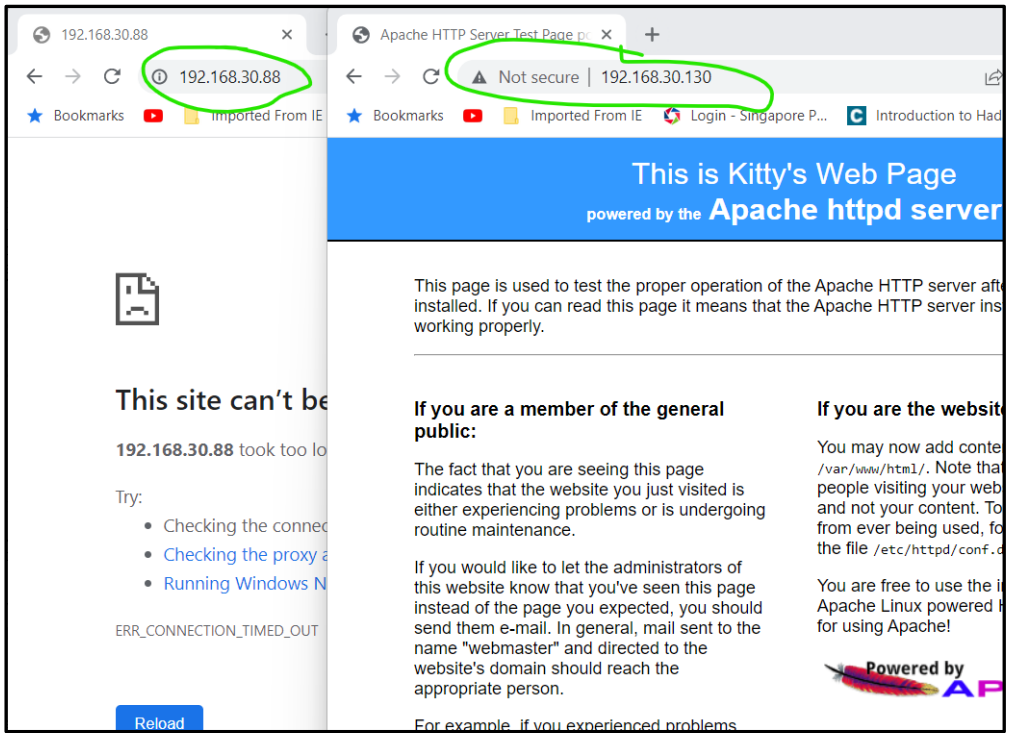
```
firewall-cmd --list-all
```

To show the current active firewall rules/settings.

```
[root@server ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: cockpit dhcpv6-client ftp http https ssh telnet
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
        rule family="ipv4" source NOT address="192.168.30.130/32" service name="http" reject
[root@server ~]#
```

As shown at the above, http service is allowed but it is overriding by the rich rule.

Of course, you can use the reverse proxy setting at the client VM to access to your server web page from the host PC via the client ip address.



Reset the firewalls of your systems

16. Reload the firewalls on both client and server to clear your runtime configuration.

```
[root@server ~]# firewall-cmd --reload
success
[root@server ~]#
```

The above is only for the server VM. Need to repeat it in the client VM too.

8. Controlling outgoing traffic by using the direct interface

Firewalld is excellent in filtering the incoming traffic, but it is not able to control the outgoing traffic. If we want to create rules to control the outgoing traffic, to disallow connections to certain external IP addresses, we must add rules using the direct interface.

On server:

1. Check that you can browse an external website using HTTPS (eg https://www.bbc.com)
2. Use the following to list your current rules that have been defined for the direct interface.

```
firewall-cmd  --direct  --get-all-rules
```

```
[root@server ~]# firewall-cmd --direct --get-all-rules
[root@server ~]#
```

(The above sample shows there is no direct rule found in your system.)

3. Add the following rule to block all outgoing ipv4 traffic through the direct interface. This rule has priority level set to 999. Rules with priority level 0 will be matched first. Rules associate with larger priority level value will be matched subsequently in ascending order. Thus, level 999 is considered to have a fairly low priority.

```
firewall-cmd --direct --add-rule ipv4 filter OUTPUT 999 -j
DROP
```

4. Check that you can no longer browse external websites nor any ipv4 related outgoing traffic.

```
[root@server ~]# firewall-cmd --direct --add-rule ipv4 filter OUTPUT 999 -j DROP
success
[root@server ~]# firewall-cmd --direct --get-all-rule
ipv4 filter OUTPUT 999 -j DROP
[root@server ~]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 8.8.8.8 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6134ms

[root@server ~]#
```

5. Add the following rule that has higher priority to allow outgoing icmp traffic.

```
firewall-cmd --direct --add-rule ipv4 filter OUTPUT 20 -p
icmp -j ACCEPT
```

You can verify the effect by issuing a ping to 8.8.8.8

```
[root@server ~]# firewall-cmd --direct --add-rule ipv4 filter OUTPUT 20 -p icmp -j ACCEPT
success
[root@server ~]# firewall-cmd --direct --get-all-rule
ipv4 filter OUTPUT 999 -j DROP
ipv4 filter OUTPUT 20 -p icmp -j ACCEPT
[root@server ~]# ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=15.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=11.3 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 11.348/13.272/15.196/1.924 ms
[root@server ~]#
```

6. Add the following rules to allow outgoing packets that belong to a connection that is already established

```
firewall-cmd --direct --add-rule ipv4 filter OUTPUT 30 -m
state --state ESTABLISHED,RELATED -j ACCEPT
```

(This rule is particularly useful to allow a server to reply to its clients' requests)

7. Add the following two rules to allow outgoing traffic to HTTPS websites (**TCP Port 443**) and queries to DNS servers (**UDP Port 53**)

```
firewall-cmd --direct --add-rule ipv4 filter OUTPUT 40 -p
tcp   --dport=443 -j ACCEPT
firewall-cmd --direct --add-rule ipv4 filter OUTPUT 50 -p
udp   --dport=53 -j ACCEPT
```

and one more if you want to access to HTTP sites

```
firewall-cmd --direct --add-rule ipv4 filter OUTPUT 60 -p
tcp  --dport=80 -j ACCEPT
```

Note: Although we set each of added rules with a unique priority. The system allows rules to set with the same priority level.

8. Check that you can browse an external website using HTTPS (eg www.bbc.com).
   Note that you may not be able to browse to external websites using HTTP (eg http://securitystartshere.org) if you have not allowed (have you ?) outgoing packets to the HTTP port.
9. Check that you cannot do a ssh to your client as your outgoing rules only allow HTTPS and DNS connections.

10. Caveat - Try to tell the difference of :

```
firewall-cmd --direct --get-all-rules
```

vs

```
firewall-cmd –list-all
```

Be careful when using rules through direct interface as these rules are not listed through the command **`firewall-cmd --list-all`**. You can view or edit them in the Firewall GUI under View->Direct Configuration.

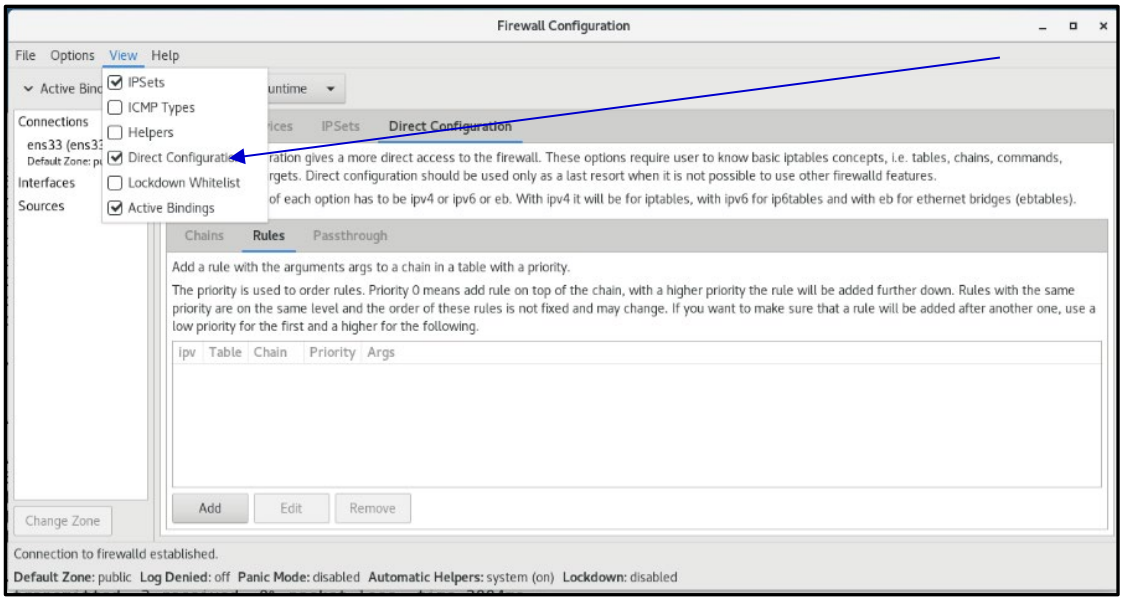
To make the rules permanent, you need to add the **--permanent** option. Permanent direct interface rules are stored in /etc/firewalld/direct.xml.

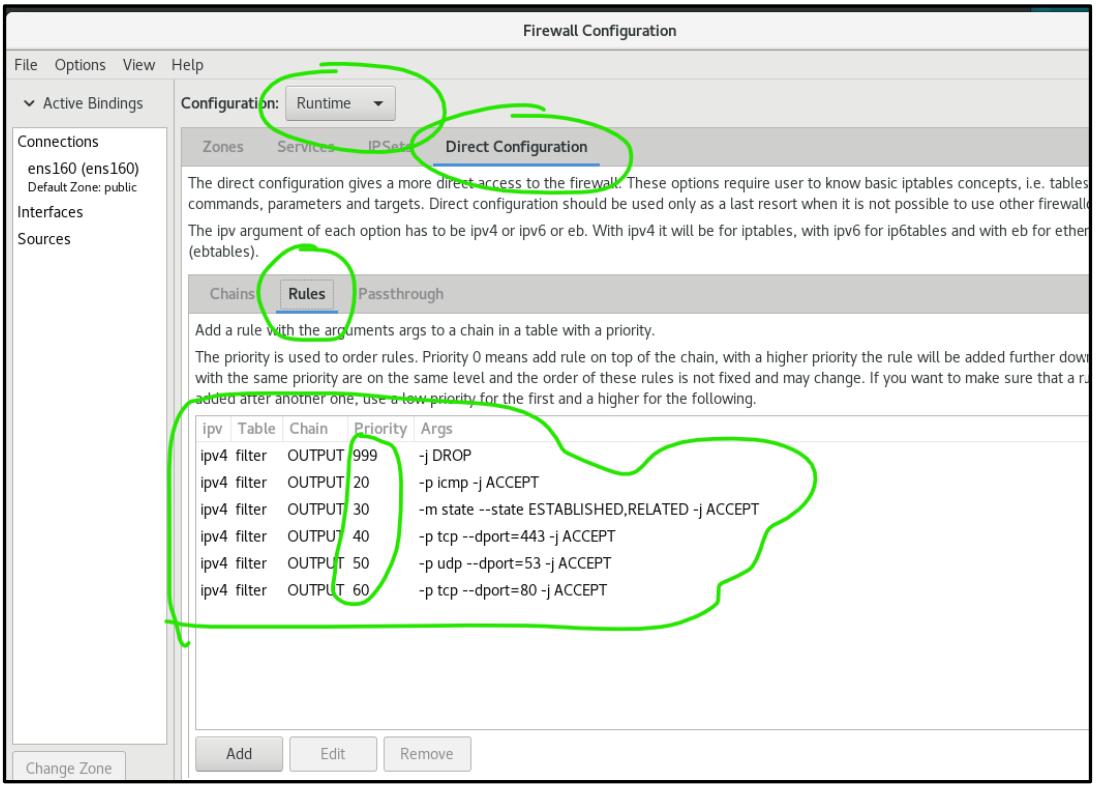
11. To remove the rules using direct interface:

```
firewall-cmd --direct --remove-rule ipv4 filter OUTPUT 20
-p icmp  -j ACCEPT
firewall-cmd --direct --remove-rule ipv4 filter OUTPUT 30
-m state --state ESTABLISHED,RELATED -j ACCEPT
firewall-cmd --direct --remove-rule ipv4 filter OUTPUT 40
-p tcp  --dport=443 -j ACCEPT
firewall-cmd --direct --remove-rule ipv4 filter OUTPUT 50
-p udp  --dport=53 -j ACCEPT
firewall-cmd --direct --remove-rule ipv4 filter OUTPUT 60
-p tcp  --dport=80 -j ACCEPT
firewall-cmd --direct --remove-rule ipv4 filter OUTPUT
999  -j DROP
```

or

you may use the Firewall GUI to remove all the direct interface rules. (This requires you to enable the viewing of the Direct Configuration).





12. Proceed to the next section after you have cleared all the direct interface rules.

9. Exploring the block zone and drop zone

So far, we only focus on setting up rules (simple, rich and direct) to the public zone which is tie to our ens160 network interface.
In fact, firewalld first check on the source of the incoming traffic to determine which zone to be used to accept the traffic. Since we have not defined any 'source' to any of the zones, thus, firewalld always bases on the interface (ie. ens160) and use public zone to handle the incoming traffic. In this section, we explore to use block zone and drop zone to see how 'source' can determine the zone selections.

<u>On client:</u>

1. Run the following to add in source to the block zone:

```
firewall-cmd --zone=block --add-source= <IP of your server>
```

e.g.

```
[root@client ~]# firewall-cmd --zone=block --add-source=192.168.30.88
success
```

The above is run successfully to set the source to my server IP. (192.168.30.88).

2. You may enter the following command to check the block zone configurations:

```
firewall-cmd --zone=block --list-all
```

```
[root@client ~]# firewall-cmd --zone=block --list-all
block (active)
  target: %%REJECT%%
  icmp-block-inversion: no
  interfaces:
  sources: 192.168.30.88
  services:
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

On Server:

3. To verify that you cannot connect to your client. You may try to ping it.

```
[student@server ~]$ ping 192.168.30.130  -c 3
PING 192.168.30.130 (192.168.30.130) 56(84) bytes of data.
From 192.168.30.130 icmp_seq=1 Packet filtered
From 192.168.30.130 icmp_seq=2 Packet filtered
From 192.168.30.130 icmp_seq=3 Packet filtered

--- 192.168.30.130 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2006ms
```

The above shows the effect of block zone is handling the ping request when the source of packets matches. The -c 3 option applied is to limit the ping to 3 times.

On Client:

4. To revert the firewall settings, remove the source configurations from the block zone, type:

   ```
   firewall-cmd --zone=block --remove-source= <IP of your server>
   ```

   and

   ```
   firewall-cmd --zone=block --list-all
   ```

   The 2nd command of the above displays the configurations of the block zone.

```
[root@client ~]# firewall-cmd --zone=block --remove-source=192.168.30.88
success
[root@client ~]#
[root@client ~]#
[root@client ~]# firewall-cmd --zone=block --list-all
block
  target: %%REJECT%%
  icmp-block-inversion: no
  interfaces:
  sources:
  services:
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@client ~]#
```

5. Repeat the Step 1 to Step but apply them to the drop zone. This will give you the 1st hand experience to find out the difference between 'block' and 'drop' effects.

10. Protecting from brute force attack to sshd

ssh is a common way to enable remote secured shell access for Linux / Unix administrators. However, once the sshd service is enabled, it may be subjected to brute force attack.
One way to avoid this attack is to limit the number of failed login attempts.
We will install Fail2ban-firewalld package to our server to implement such brute force attack counter measure. Fail2ban package works well together with firewalld.

Ref: ~https://devops.ionos.com/tutorials/install-fail2ban-on-centos-7-to-protect-ssh-via-firewalld/
~https://www.howtoforge.com/tutorial/how-to-install-fail2ban-on-centos/

Introduction of epel-release yum repository:
This repository provides Extra Packages for Enterprise Linux (EPEL) is a special interest group (SIG) from the Fedora Project that provides a set of additional packages for RHEL (and CentOS, and others) from the Fedora sources.

Ref: ~https://www.redhat.com/en/blog/whats-epel-and-how-do-i-use-it

<u>On Server:</u>

1. We need to add in epel-release repository to the system. To search and install, type:

```
dnf search oracle-epel-release
```


```
dnf install oracle-epel-release-el8
```

(The .x86_64 suffix is optional)

2. Check if fail2ban has been installed yet.

```
dnf info fail2ban-firewalld
```

or

```
dnf list installed | grep fail2ban-firewalld
```

3. If it has not been installed run dnf update, then install fail2ban package.

```
dnf update
dnf -y install fail2ban-firewalld
```

(Note: we only need fail2ban-firewalld for our setup. To install the full set of fail2ban, use dnf install fail2ban)

4. The default configuration folder of fail2ban is at /etc/fail2ban. The main configuration file that is related for banning access is named jail.conf. We can also add in additional 'jail' configuration settings by adding .local files into the /etc/fail2ban/jail.d subfolder. (a drop-in folder, in Linux terminology.)
Create a new text file 'sshd.local' into the /etc/fail2ban/jail.d subfolder.
Add the following lines into the file to define our 'jail' settings:

```
[sshd]
enabled = true
port = ssh
# you may comment out the following if you only want to log the failed login.
banaction = iptables-multiport
logpath = %(sshd_log)s
maxretry = 3
# connection will be banned for 600 seconds when tried more than 3 times
bantime = 600
```

5. Start the fail2ban service now. Type in:

```
systemctl start fail2ban
```

6. Verify if fail2ban service is running. Type in:

```
systemctl status fail2ban
```

```
[root@server jail.d]# systemctl start fail2ban
[root@server jail.d]# systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
   Loaded: loaded (/usr/lib/systemd/system/fail2ban.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2022-10-11 18:40:34 +08; 7s ago
     Docs: man:fail2ban(1)
  Process: 83679 ExecStartPre=/bin/mkdir -p /run/fail2ban (code=exited, status=0/SUCCESS)
 Main PID: 83681 (fail2ban-server)
    Tasks: 5 (limit: 11408)
   Memory: 15.2M
   CGroup: /system.slice/fail2ban.service
           └─83681 /usr/bin/python3.6 -s /usr/bin/fail2ban-server -xf start

Oct 11 18:40:34 server.example.com systemd[1]: Starting Fail2Ban Service...
Oct 11 18:40:34 server.example.com systemd[1]: Started Fail2Ban Service.
Oct 11 18:40:34 server.example.com fail2ban-server[83681]: Server ready
[root@server jail.d]#
```

(Note that the fail2an service is running but it is not enabled. It implies it will not automatically start in the next boot up.)

7. Verify if fail2ban is running and the sshd.jail is set. Type in:

```
fail2ban-client status
```

```
[root@server jail.d]# systemctl start fail2ban
[root@server jail.d]# fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:   sshd
[root@server jail.d]#
```

(The above shows that the sshd jail is ready.)

8. To check whether there is any banned ip. Type:

```
fail2ban-client status sshd
```

```
[root@server jail.d]# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 1
|  |- Total failed:     1
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
[root@server jail.d]#
```

9. Now the fail2ban is activated and ready to block connections from blacklisted IP addresses. The blacklist IP addresses will be defined in a customised iptables chain* which can be used to define iptables direct rules to deny / accept a set of IP addresses. Verify if your system has any iptables chain objects, type in:

```
iptables -L
```

or

```
iptables -L  <a known iptable chain name>
```

Note: There is no customised iptable chain defined in the system at the beginning even the fail2ban has been started.

```
[root@server ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@server ~]#
```

(Note: INPUT, FORWARD, and OUTPUT are system build-in default chains. And they are all empty at this point.).

On Client:

10. Verify if fail2ban is guarding your sshd from brute force attack by trying ssh to your server with wrong passwords.

```
[root@client ~]# ssh student@192.168.30.88
student@192.168.30.88's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Mon Oct 10 14:48:00 2022
[student@server ~]$ echo I am in
I am in
[student@server ~]$ exit
logout
Connection to 192.168.30.88 closed.
[root@client ~]# ssh student@192.168.30.88
student@192.168.30.88's password:
Permission denied, please try again.
student@192.168.30.88's password:
Permission denied, please try again.
student@192.168.30.88's password:
student@192.168.30.88: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
[root@client ~]#
[root@client ~]#
```

In the above session, the first ssh session to server from client is successful. The second attempt uses all wrong passwords then triggers the fail2ban effect.

On Server: (within 5 minutes)

11. Check if we can see there is a blacklisted ip for sshd access, Type:

```
fail2ban-client status sshd
```

```
[root@server ~]# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     3
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   192.168.30.130
[root@server ~]#
```

12. Verify if a new iptables chain has been defined by fail2ban and if there are any direct rule in the chain to block your client IP from accessing to the server.

```
[root@server ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
f2b-sshd   tcp  --  anywhere             anywhere             multiport dports ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain f2b-sshd (1 references)
target     prot opt source               destination
REJECT     all  --  192.168.30.130       anywhere             reject-with icmp-port-unreachable
RETURN     all  --  anywhere             anywhere
[root@server ~]#
```

As depicted at the above, a new chain appears with the name, f2b-sshd. A new rule is added to the default INPUT chain to channel all the incoming ssh traffic to this f2b-sshd chain. Ssh traffic is defined by the destination port equals to ssh (dports ssh).

In the definition of the f2b-sshd chain, we can see the client IP has been blocked by a REJECT rule (The first rule). The last rule in the f2b-ssh chain allows other source ip packet to proceed by returning the flow to the INPUT chain.

13. Verify if the /var/log/secure file has logged the invalid ssh access attempts :

```
cat /var/log/secure | grep -i 'failed password'
```

```
[root@server ~]# tail /var/log/secure | grep 'Failed password'
Oct 11 18:45:01 server sshd[83902]: Failed password for student from 192.168.30.130 port 34430 ssh2
Oct 11 18:45:06 server sshd[83902]: Failed password for student from 192.168.30.130 port 34430 ssh2
Oct 11 18:45:11 server sshd[83902]: Failed password for student from 192.168.30.130 port 34430 ssh2
[root@server ~]#
```

(Note: grep with -i switch is to allow non-case sensitive matching)

> For you information, the working concept of fail2ban is based on the log content to determine if failed attempts have made on various services. SSH is one of them, based on the similar approach, administrator can use fail2ban to monitor and ban access on other type of services, such as web service, email service and etc ….

14. Wait for 5 minutes or longer and verify what will happen to the f2b-sshd Chain and the iptable rule:

```
[root@server ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
f2b-sshd   tcp  --  anywhere             anywhere             multiport dports ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain f2b-sshd (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
[root@server ~]#
```

As shown , the iptables chains are remaining. But there are no more REJECT rules.

15. Re-run the fail2ban-client status sshd:

```
[root@server ~]# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     3
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:     1
   `- Banned IP list:
[root@server ~]#
```

16. Reset your server setting by stopping the fail2ban sshd jail **gracefully**. Type:

```
fail2ban-client stop sshd
```

17. Now you can stop your fail2ban server. Type:

```
systemctl stop fail2ban
```

```
[root@server ~]# fail2ban-client stop sshd
Jail stopped
[root@server ~]# systemctl stop fail2ban
[root@server ~]#
[root@server ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@server ~]#
```

As shown, fail2ban-client will remove the customised f2b-sshd chain from the iptables.

Additional Reference:

Home page of the firewalld - https://firewalld.org/

firewall-cmd Cheat Sheet - https://cheatography.com/mikael-leberre/cheat-sheets/firewall-cmd/

Running a quick NMAP scan - https://www.redhat.com/sysadmin/quick-nmap-inventory

Port Scanning Basics - https://nmap.org/book/man-port-scanning-basics.html

Linux security: Protect your systems with fail2ban - https://www.redhat.com/sysadmin/protect-systems-fail2ban

End of Practical