iptables case-studies

Fabio Massimo Ercoli *

December 28th 2024

1 Set up Linux Server VM

To test a firewall what we really need is a server Linux distribution like Fedora server, that does not have by default a graphic environment, but it is only accessible by shell (local / remote).

To setup the VM I use Virt Manager as a virtual environment, that is based on the KVM virtualization.

These are commands to install and start it:

```
sudo dnf install @virtualization
sudo systemctl start libvirtd
virt-manager
```

The ISO of the operating system can be downloaded from the Fedora Server portal. In a few steps we can install our Fedora Server. We don't need much memory and CPUs, the OS is very lightweight. See figure 1 and 2.

In one of the picture we can see that we keep the default virtual network configuration, using *NAT*, that is the simplest way of accessing an external network from a virtual machine. A virtual machine with NAT enabled acts much like a real computer that connects to the Internet through a router.

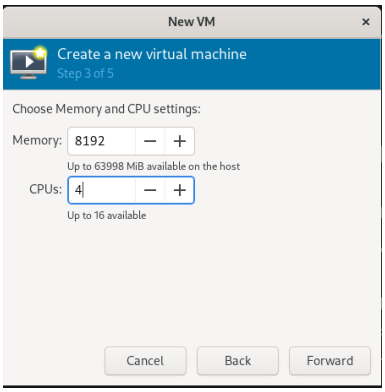
The *iptable* software is provided by default by the Linux distribution. Prompting:

```
iptables --version
```

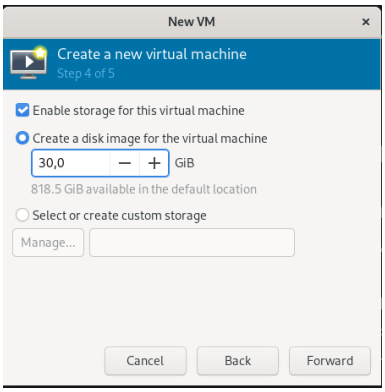
We get the version in use:

```
iptables v1.8.10 (nf_tables)
```

*More on me: my GitHub page - my Linkedin page.

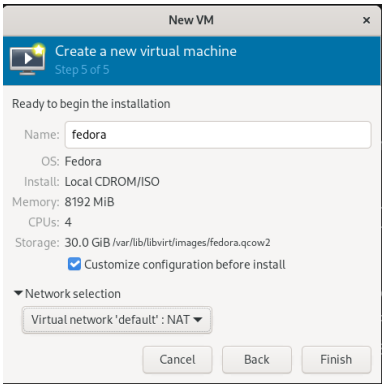


(a) Memory CPU

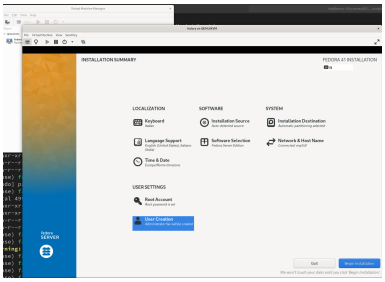


(b) Disk

Figure 1: Linux VM resources



(a) Network



(b) Installation settings

Figure 2: Configure Linux VM

2 Scenario: Allow the guest VM to access the internet

2.1 Information security requirements

The *guest* VM we have just created does not have any access to the external internet. With the NAT configuration all the traffic incoming to and outgoing from the *guest* VM is supposed to pass through the *host* operating system. The host operating system is another Fedora Linux, this time a workstation edition and probably it does not allow by default the traffic to pass through it.

We want to allow any connection from/to the external internet to pass through the *host* operating system in order to make the *guest* VM connected.

2.2 Test before the configuration

To test the fact that the VM is isolated from the external internet, we can try the following *wget* command:

```
wget www.facebook.com
```

This command blocks without returning if it is run from the *guest* VM console, while it works perfectly if it is run from the *host* OS.

2.3 Configure iptables to meet the requirements

In this case we need to verify the iptables configuration of the *host* operating system, in particular looking at the *filter* table, chain *FORWARD*.

To inspect all the chains of the filter table. We can run the command:

```
sudo iptables -vL
```

We can immediately notice that the default policy of the *FORWARD* is *DROP*. Furthermore, since we called the command with the *v* verbose option, we notice that some packets have been blocked:

```
Chain FORWARD (policy DROP 175 packets, 15712 bytes)
```

Inspecting the network we get the *guest* VM IP address, that in this case is *192.168.124.203*. This address is a virtual address managed by the virtual network defined on the *host* machine. We can add a rule on top of the chain:

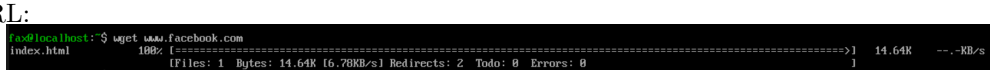
```
sudo iptables -I FORWARD -d 192.168.124.203 -j ACCEPT
sudo iptables -I FORWARD -s 192.168.124.203 -j ACCEPT
```

And verifying that is has been acquired:

```
Chain FORWARD (policy DROP 175 packets, 15712 bytes)
pkts bytes target     prot opt in     out     source               destination
0    0     ACCEPT     all  --  any    any     192.168.124.203      anywhere
0    0     ACCEPT     all  --  any    any     anywhere             192.168.124.203
```

2.4 Test after the configuration

Redoing the same *wget* on the guest VM, we can now access to the *www.facebook.com* URL:



Moreover can query again the *iptable* on the host machine and see:

```
Chain FORWARD (policy DROP 175 packets, 15712 bytes)
pkts bytes target     prot opt in     out     source               destination
595  37000 ACCEPT     all  --  any    any     192.168.124.203   anywhere
1079 1572K ACCEPT     all  --  any    any     anywhere          192.168.124.203
```

That some packets have been passed though the host machine matching the rules we have just defined.

Another nice thing that we can notice it that we didn't need to restart the *iptables* nor the *virt-manager* demons. The changes have been applied on the fly!

3 Scenario : prevent users to use a given portal

3.1 Information security requirements

We don't want people using VM can access to the *www.facebook.com* portal. We want to limit the distractions of our employees. We already tested that it is now possible to access the portal.

3.2 Configure iptables to meet the requirements

One option would be configure *FORWARD* chain on the *host* operating filter, another option would be to configure *iptables* on the *guest* VM.

I'll follow this second option.

Querying *iptables* of the *guest* VM, we can see here that the chains of the filter tables are all empty and have all *ACCEPT* as default policy.

We can create a *sh* script like the following one:

```
#!/bin/bash
for blockip in `resolveip facebook.com | cut -d "-" -f 6`; do
 sudo iptables -A OUTPUT -o eth0 -p tcp -d $blockip -j DROP
 sudo iptables -A INPUT -p tcp -s $blockip -j DROP
done
```

a couple of rules are added for each IP address to block. Running the script should add the rules on the *host* VM. Notice that *resolveip*, if not present, is supposed to be installed using the command:

```
sudo dnf install resolveip
```

3.3 Test after the configuration

Again, without the need of restarting the *ipconfig* service, we can check that now the command does not work anymore, and one of the blocking rule we've defined has matched some packets.

To check that this configuration does not affect the use of other sites , such as *redhat.com*, trying a:

```
wget www.redhat.dom
```

the packets are not blocked in this case:

4 Scenario: limiting the access to an HTTP server

4.1 Information security requirements

In our guest machine we're going to install an instance of Apache HTTPd server. We want to demonstrate how to forbid the use of the service from the host machine.

This is an example of limiting the access to the server.

Moreover, we want to extract the rules we need to define for this scenario in a different (user-defined) chain, so that they can be potentially reused.

4.2 Install the Apache service

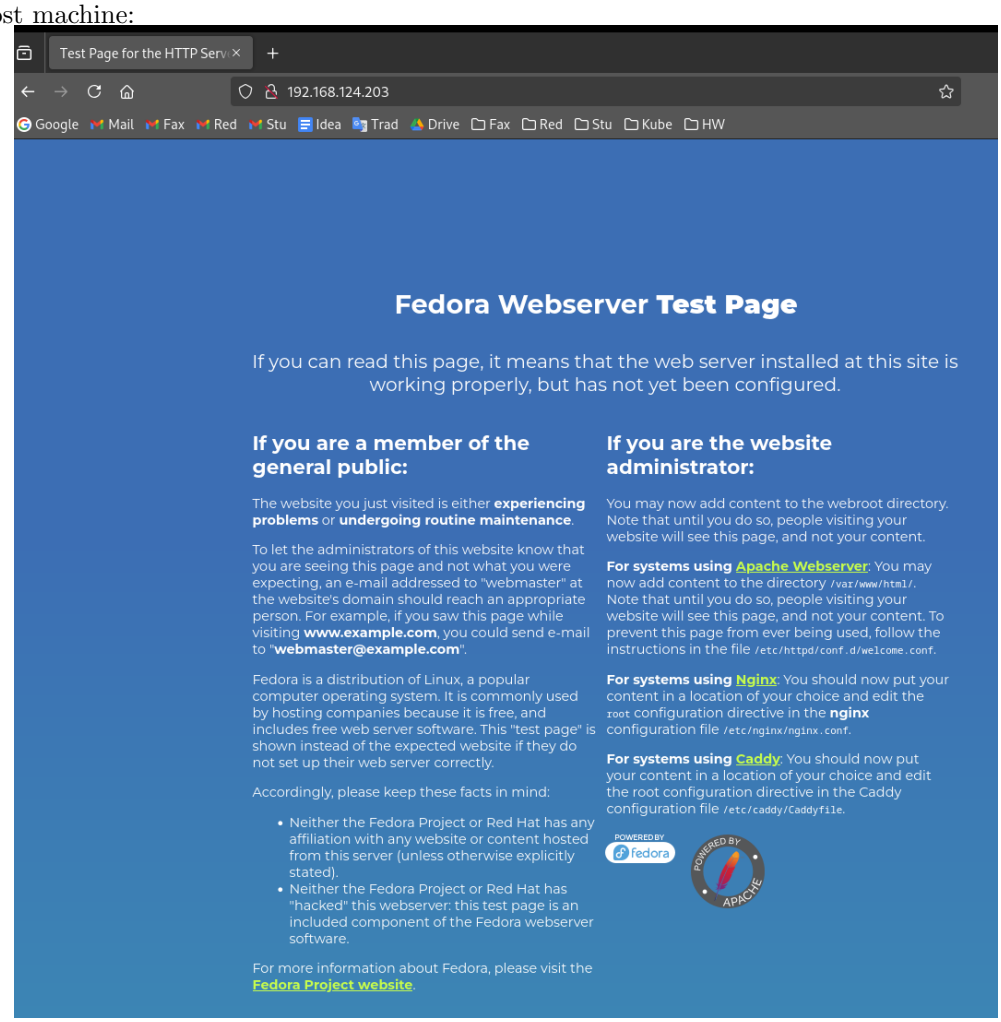
To install the Apache server on the VM machine we can run the following commands:

```
sudo dnf install httpd -y
sudo systemctl start httpd.service
sudo firewall-cmd --add-service=http
```

The first command install the binary on the filesystem, the second one starts the process as service OS (daemon), the last command allows to receive connections from outside on the port 80, using this time OS firewall. That is probably an application level firewall, since it has the concept of application level protocol (http in this case).

4.3 Test before the configuration

The Apache HTTP hello page is accessible from the browser running from the host machine:



4.4 Configure iptables to meet the requirements

We can have the IP address of the host machine looking at the Apache access logs:

```
sudo cat /var/log/httpd/access_log
```

The host machine seen from the guest machine has IP address: *192.168.124.1*.

I create two custom chains on the default IP table, filtering: *input-apache-chain*, *output-apache-chain*, deleting also possible pre-existing custom chains and associations:

```
sudo iptables -F
sudo iptables -X
sudo iptables -N input-apache-chain
sudo iptables -A input-apache-chain -p tcp -m multiport --dports 80,443
  -s 192.168.124.1 -m state --state NEW,ESTABLISHED,RELATED -j DROP
sudo iptables -A INPUT -j input-apache-chain
sudo iptables -N output-apache-chain
sudo iptables -A output-apache-chain -p tcp -m multiport --sports 80,443
  -d 192.168.124.1 -m state --state NEW,ESTABLISHED,RELATED -j DROP
sudo iptables -A OUTPUT -j output-apache-chain
```

Executing the *sudo iptables -vL* we get:

```
Chain INPUT (policy ACCEPT 374 packets, 32712 bytes)
pkts bytes target        prot opt in     out     source     destination
388  33552 input-apache-chain all  --  any    any     anywhere   anywhere

Chain OUTPUT (policy ACCEPT 102 packets, 9168 bytes)
pkts bytes target         prot opt in     out     source    destination
102  9168  output-apache-chain all  --  any    any     anywhere  anywhere

Chain input-apache-chain (1 references)
pkts bytes target     prot opt in     out     source         destination
14   840   DROP       tcp  --  any    any     _gateway       anywhere
multiport dports http,https state NEW,RELATED,ESTABLISHED

Chain output-apache-chain (1 references)
pkts bytes target     prot opt in     out     source         destination
0    0     DROP       tcp  --  any    any     anywhere       _gateway
multiport sports http,https state NEW,RELATED,ESTABLISHED
```

4.5 Test after the configuration

Immediately after applying the rule the Apache hello page became no longer callable from the Firefox run from the *host* machine, as desired.

5 Persist and reload the iptables configuration

Any change to the *iptables* configuration is lost, when the operating system is restated.

The current configuration can be serialized to some file using *iptables-save*:

```
sudo iptables-save > my-iptables
```

When the system is restarted, the configuration can be reloaded using *iptables-restore*:

```
sudo iptables-restore > my-iptables
```

The load can be automatized adding the latter command to some script that we know it is always run at startup. For instance *.bash_profile* or *.bashrc*. System administrators know that.

6 Extra: use SSH to configure the VM

Instead of using the bash provided directly by the virtual environment, we can use *SSH* to run commands on the remote *guest* VM operating system directly using the shell of the *host* machine.

We can connect to the VM *guest* OS, just typing:

```
ssh 192.168.124.203
```

In this case the username *fax* is the name of both the client and server, so it does not need to be retyped. Password is required. This can be avoided using client certificates, they also improve the security and efficiency...