Instituto Superior Técnico, Universidade de Lisboa

Network and Computer Security

Lab guide:

Firewalls

Revised on 2016-10-26

Alpha version: This is an early version and may contain some incorrect features.

Goals

- Configure a firewall using the *iptables* and *fwbuilder* applications.

Before beginning

Figure 1 shows the network topology configuration for this laboratory assignment. Based on the previous laboratory assignment (initial configuration - left box below), the idea is to perform the necessary configuration changes to obtain the configuration on the right box below (target configuration).

INITIAL CONFIGURATION:

• **VM1:**	subnet	*adapter*
1.	192.168.0.100	*eth0*
• **VM2:**		
1.	192.168.0.10	*eth0*
2.	192.168.1.254	*eth1*
• **VM3:**		
1.	192.168.1.1	*eth0*
• **VM4:**		
1.	192.168.1.2	*eth0*

TARGET CONFIGURATION:

• **VM1:**	subnet	*adapter*
1.	192.168.0.100	*eth0*
• **VM2:**		
1.	192.168.0.10	*eth0*
2.	192.168.1.254	*eth1*
3.	**192.168.2.254**	**eth2**
• **VM3:**		
1.	192.168.1.1	*eth0*
• **VM4:**		
1.	**192.168.2.1**	***eth0***

For that, you are advised to proceed as follows:

- Add a new adapter (*eth2*) to **VM2** for the subnet 192.168.2.0 and setting **VM2** as 192.168.2.254 on that adapter's configuration.
- Change **VM4** from subnet 192.168.1.0 to 192.168.2.0, setting it as 192.168.2.1 on that subnet.

Please revise the previous lab assignment for more instructions on how to obtain the initial configuration (left box on the bottom), taking into account whether you are using **rnl-virt** or **VirtualBox**.

Note for rnl-virt users: do not forget to recreate the virtual switches for subnets sw-1 and sw-2 as you also did in the previous laboratory assignment. You will also need to create a new sw-3.

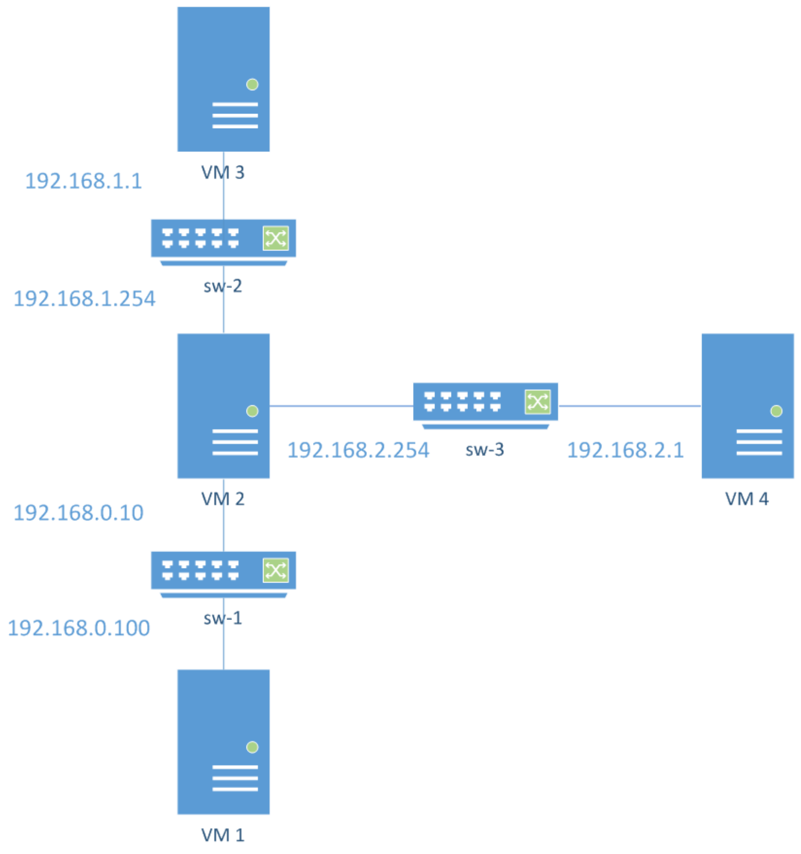


Figure 1 - Desired network topology to begin this assignment.

1. iptables

The native firewall software in Linux is part of the kernel. However, you can use the **iptables** tool (man iptables) to manage its rules.

1.1. Simple Rules

Experiment with some simple rules in **VM2**.

1.1.1. Reject ICMP packets

Execute:

```
$ sudo /usr/sbin/iptables -A INPUT -p icmp -j DROP
```

The previous command adds a rule to drop all incoming ICMP packets.

See the new rule by listing all rules managed by iptables:

```
$ sudo /usr/sbin/iptables -L
```

Test this new rule by sending a ping from **VM3** to **VM2**.

Question: Were you able to see the ping from VM3? Why not?

Use one of the following commands to erase this rule from **VM2**:

```
 $ sudo /usr/sbin/iptables –D INPUT 1
 $ sudo /usr/sbin/iptables –D INPUT –p icmp –j DROP
```

1.1.2. Ignore telnet connections

Confirm that you can establish a telnet connection to **VM2** (for example, try from **VM1**). Block these connections using the following command (in **VM2**):

```
$ sudo /usr/sbin/iptables –A INPUT –p tcp --dport 23 –j DROP
```

Check whether telnet connections to **VM2** are still possible.

Delete the previous rule by executing one of the following commands:

```
$ sudo /usr/sbin/iptables –D INPUT 1
$ sudo /usr/sbin/iptables –D INPUT –p tcp --dport 23 –j DROP
```

1.1.3. Ignore telnet connections from specific IP addresses

Ignore telnet connections from **VM1**:

```
$ sudo /usr/sbin/iptables –A INPUT –p tcp –s [host address] --dport 23 –j
DROP
```

Confirm that all machines except **VM1** are able to open a telnet connection with **VM2**.

1.1.4. Ignore telnet connections from a specific subnet

Ignore telnet connections from the subnet that includes **VM4**.

```
$ /usr/sbin/iptables –A INPUT –p tcp –s 192.168.2.0/24 --dport 23 –j DROP
```

At this point you should not be able to open a telnet connection to **VM2** from **VM4**.

Delete all existing rules:

```
$ sudo /usr/sbin/iptables –F
```

1.2. Redirect connections

The previous exercises used the INPUT chain from the Filter table. We will now use the PREROUTING chain in the NAT table in order to redirect network packets.

Execute:

```
$ sudo /usr/sbin/iptables -t nat -A PREROUTING --dst 192.168.0.10 -p tcp --
dport 23 –j DNAT  --to-destination 192.168.1.1
$ sudo /usr/sbin/iptables -t nat -L
```

The second command displays the NAT rules.

Make a telnet connection from **VM1** to **VM2**.

Confirm that the connection was established between **VM1** and **VM3** using the **netstat –t** command on **VM3**:

```
$ netstat -t
```

In order to redirect *http* traffic to **VM3** change from port 23 to 80 on the previous iptables command.

Use a browser in **VM1** and go to **`http://192.168.0.10`**. Run netstat –t to confirm that the connection is between **VM1** and **VM3**:

```
$ netstat -t
```

Delete all existing rules:

```
$ sudo /usr/sbin/iptables -F
$ sudo /usr/sbin/iptables -t nat -F
```

2. Fwbuilder

This section introduces **fwbuilder**, which is a cross-platform firewall management software. It is to be installed on **VM2**.

2.1. Install fwbuilder extensions

2.1.1. Download the *fwbuilder-extensions.iso* file from the course homepage (**rnl-virt** users may skip the download part as the image is available in the cd lists) at http://disciplinas.tecnico.ulisboa.pt/SIRS/2015-2016/labs/4/fwbuilder-extensions.iso.

2.1.2. Load the ISO into:

2.1.2.1. VirtualBox:

- Devices -> CD/DVD Devices -> Choose a virtual CD/DVD disk file
- Select *fwbuilder-extensions.iso*

2.1.2.2. rnl-virt:

- `$ rnl-virt vm insert-cd` **`VM2`** `fwbuilder-extensions`

2.1.3. Mount the CD-ROM in the virtual machine (skip this command if using **rnl-virt**):

Before running this command, check if the fwbuilder files are already in /media/cdr0. If they are, it is not necessary to run this command and you can skip to 2.1.4.

```
$ sudo mount -t iso9660 /dev/hdb /media/cdr0
```

2.1.4. Install the fwbuilder extension:

```
$ sudo rpm –i /media/cdr0/fwbuilder-ipt-2.0.9-1.pm.1.i586.rpm
```

2.2. *Simple rules*

Run **fwbuilder** (`$ sudo fwbuilder`) and create a new project.

2.2.1. *Create a new firewall*

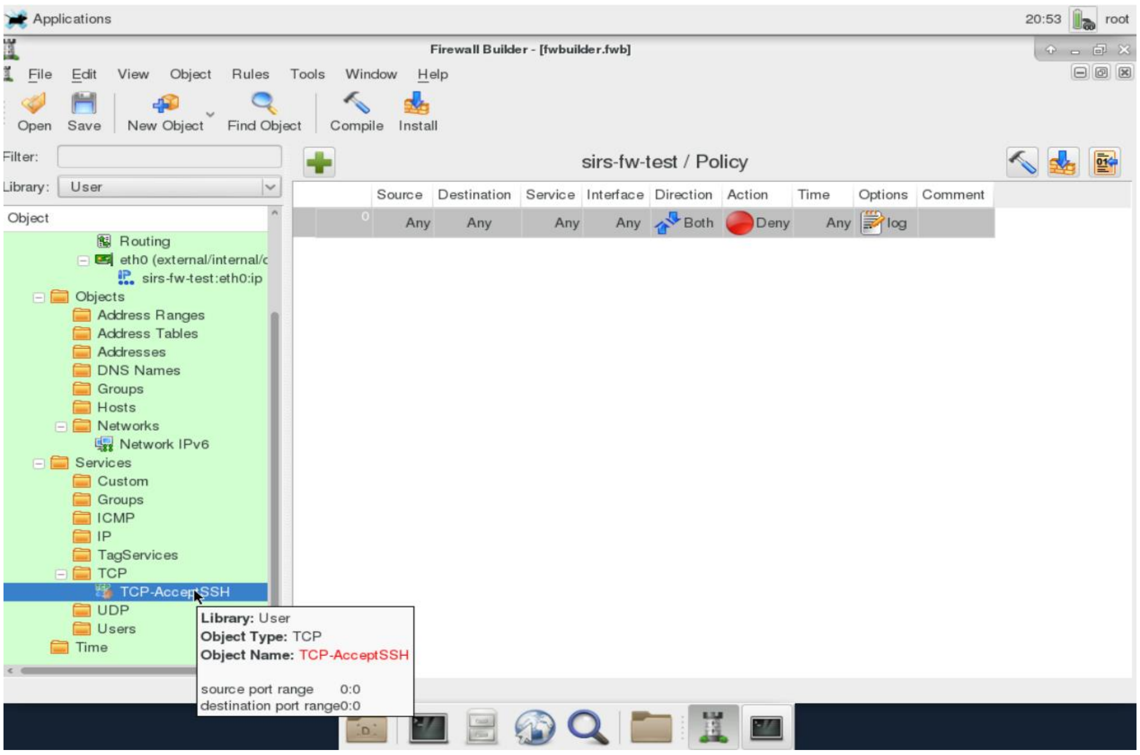
- Create new project file (File -> Save as…).
- **The firewall will be stored in an fwbuilder .fwb project file. Choose a name for the new project (e.g. sirs-firewall).**
- Click Save
- The main firewall configuration overview window should now be open. It is titled **Firewall Builder: <firewall project name>**. For the name suggested earlier, it will be **Firewall Builder: sirs-firewall.fwb**.
- Click Object -> New Object -> New Firewall**.**
- Configure the new firewall with:
    - ***Choose firewall software it is running:*** *iptables*
    - ***Choose OS the new firewall runs on:*** *Linux 2.4/2.6.* The name may be something such as **'sirs-fw-test'**.
- Click Next >
- The following window should have two radio buttons with only the **Configure interfaces manually** option selected.
- Click Next >
- Add the network interfaces. The information to be given to each network interface configuration may be displayed through the command:

  ```
  $ sudo /sbin/ifconfig
  ```
- For each, you should fill in the following fields:
    - **Name: ethX**
    - **Address: 192.168.Y.254**
    - **Netmask: 255.255.255.0**
    - **Label: external/internal/dmz**
- You should configure the interfaces in the firewall accordingly.
- Set one of the interfaces as a **management interface** (you may do this by right-clicking one of the interface icons in *Firewalls -> **sirs-fw-test*** assuming that was the name you gave the firewall).
- Save the current project file, in case something happens. ***The default location for it is the current user's home directory.***

2.2.2. *Accept ssh connections*

fwbuilder requires that the machine accepts ssh connections in order to install new firewall rules.

- Create a new TCP service with destination port 22 (Object -> New Object -> New TCP service). Call it, for example, **TCP-AcceptSSH**.
- Create a new rule (Rules -> Insert Rule).
- Drag the new service into the **Service** field (as depicted in Figure 2).
- Change the **Action** field to **Accept** (right-click on **Deny** to display a list where you can choose **Accept).**
- Click **Rules -> Install**. This will ask for a user and a password. Use the system's administrator credentials with user ***'root'*** and password ***'inseguro'***.
- Test the *ssh* connections. They should be working.



	Source	Destination	Service	Interface	Direction	Action	Time	Options	Comment
0	Any	Any	Any	Any	Both	Deny	Any	log	

Library: User
Object Type: TCP
Object Name: TCP-AcceptSSH

source port range 0:0
destination port range0:0

Figure 2 - Example TCP service creation and use in policy rule.

2.2.3. *Accept telnet connections*

- Check whether your current machine (VM2) is accepting telnet connections.
- Check all firewall rules with

  ```
  $ sudo iptables -L
  ```
- Create a new TCP service with destination port 23.
- Create a new rule accepting connections to the new service.
- Install the firewall.

- Test the telnet connections.

Questions: did you manage to create a telnet connection to VM2 in the beginning? What happened, and why? After executing these instructions, what happens when you try to telnet into VM2?

2.2.4. *Redirect telnet connections*

This is an introductory exercise to what you will find in 2.3. Make sure the relevant virtual machines have their gateway configured appropriately. If you are uncertain about the origin, destination and redirection of certain packets, **`tcpdump`** ***is a good way to make sense of the traffic.***

- Configure *eth0* as **external.**
- Add the IP address for **VM3** in **addresses** (Objects -> Address)
- Add the necessary rule in the NAT table.
- Set the original address, service and redirect address.
- Install the firewall and test this rule.

2.3. *Internal Network + DMZ*

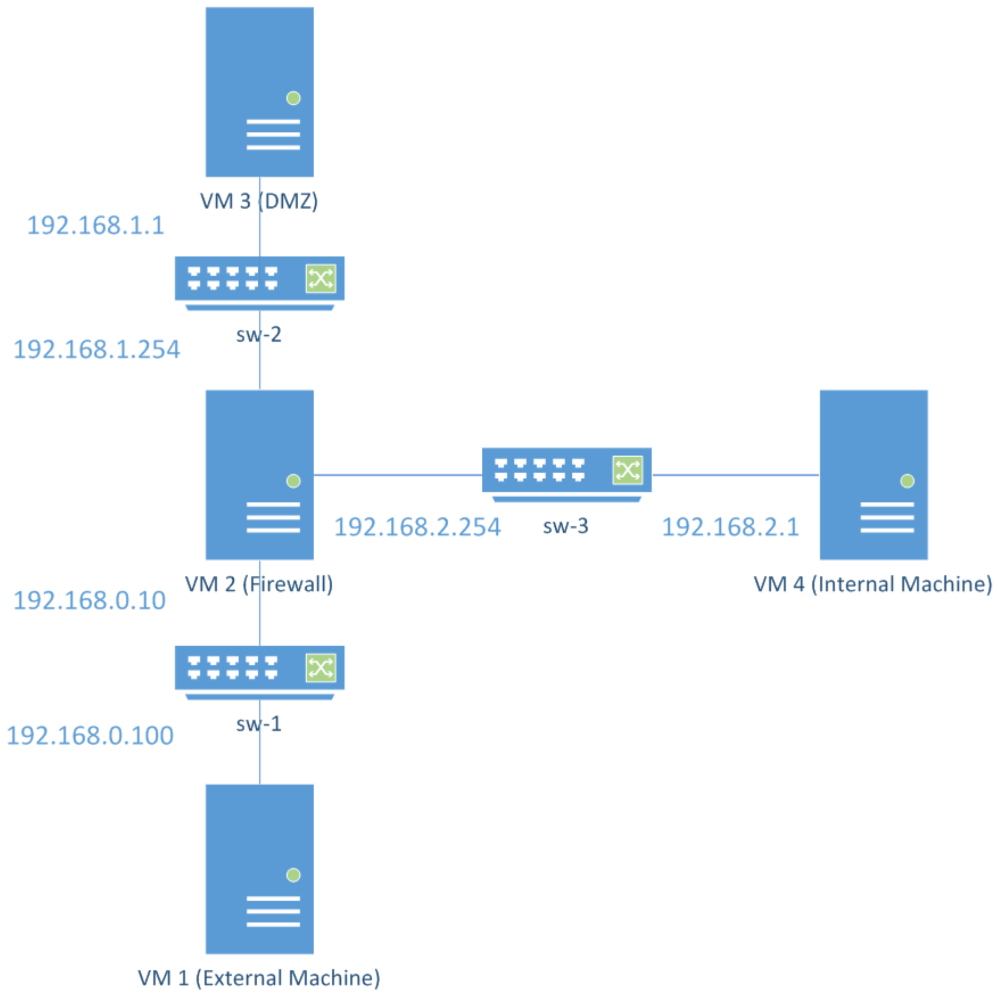


Figure 3 - network topology and individual machine roles in the *Internal Network + DMZ* scenario.

Use **fwbuilder** to configure the following requirements:

- **VM1** is an external machine:
  - **VM1** will only be able to open *ssh* (port 22) and *http* (port 80) connections with **VM2**.
- **VM2** is the *firewall*
  - All *http* (port 80) connections are redirected to **VM3**.
  - All *ssh* connections from the external network are redirected to **VM4**.
  - Requests from the internal network 192.168.4/8 are only accepted if destined to the *ssh* port.
  - All other traffic is rejected.
- **VM3** is a Web server in a DMZ:
  - Accepts *http* connections from both the internal and external networks.
  - Accepts *ssh* connections from the internal network.

    - Does not start any new connections.
- **VM4** is an internal machine:
    - Accepts *ssh* requests.
    - Is able to open *ssh* connections to both external network and DMZ.