Instituto Superior Técnico, Universidade de Lisboa
Network and Computer Security

Lab guide:
Implementation and Analysis of a Virtual Computer Network

rnl-virt version for RNL Labs at Alameda campus. Revised on 2016-10-10

Goals

- Implement and test a virtual computer network.
- Perform a simple TCP/IP packet analysis.

1. Introduction

This guide describes how to use the RNL computers and **rnl-virt**, the RNL virtualization system. The tool's documentation is available at https://rnl.tecnico.ulisboa.pt/servicos/virtualizacao/ and should be referenced for additional command details.

rnl-virt relies on *libvirt* that is based on QEMU. *Libvirt* is a special daemon used to create, launch and shutdown virtual machines. The physical machine is called the **Host** whereas the virtual machines are called **Guests**. Each guest will use a version of Caixa Mágica Linux with low memory requirements.

Before you begin, logon to a lab machine running Linux using your IST account (e.g. **ist123456**). You must restore and save your files for each work session.

Use the alternative temporary folder `/var/tmp` for storing your work session files. `/var/tmp` is a temporary file system directory, so files are deleted on logoff. However, the files are deleted only when an explicit logoff is performed, so if the machine reboots unexpectedly due to a system crash, you *should* be able to log again and still recover the files. This is **not** the behaviour of the temporary folder `/tmp`. For example, we recommend creating a directory whose name is based on your student number. So, to prepare your working environment, execute the following command (replacing 123456 with your own student number):

```
$ mkdir /var/tmp/ist123456
$ cd /var/tmp/ist123456
```

Your home folder ~/ is **not** recommended for direct use in work sessions because it is an AFS directory with increased latency due to the distributed file system overhead. However you can use it to save backups of your virtual disks. You can also save your backups in an external USB drive.

2. Create the virtual networks and machines

This assignment encompasses the creation of virtual networks, for which three separate virtual machine instances will need to be created, as depicted in Figure 1. Each instance will have a differential disk, based on the original disk. The differences are stored in files.

Figure 1 – In this laboratory assignment, you will need to create three virtual machines, which will be used to explore communication inside virtual networks and to test packet forwarding.

First, we are going to create the differential disks, one for each virtual machine:

```
$ rnl-virt disk create sirs-1 SIRS
$ rnl-virt disk create sirs-2 SIRS
$ rnl-virt disk create sirs-3 SIRS
```

These names (**sirs-1.qcow2, sirs-2.qcow2**, and **sirs-3.qcow2**) are suggestions. They are, respectively, the disks of instance **VM1**, **VM2** and **VM3**. These are the files where all changes are stored, so create backups as needed (any virtual machine you wish to backup must be shut down first).

We have created the virtual hard disks; next we will create the virtual machine instances named **VM1**, **VM2** and **VM3**. But before we do so, we must take note: we are going to create two networks: one between **VM1** and **VM2** and another between **VM2** and **VM3**, as can be seen in Diagram 1. Since we are using the **rnl-virt** system, we must create *virtual switches*, one per network/subnet. These switches

need to be associated to the virtual machines **before** they start running, so we shall create the switches now:

```
$ rnl-virt switch create sw-1 --hub
$ rnl-virt switch create sw-2 --hub
```

The hub option means that the virtual frames are broadcast in the network.

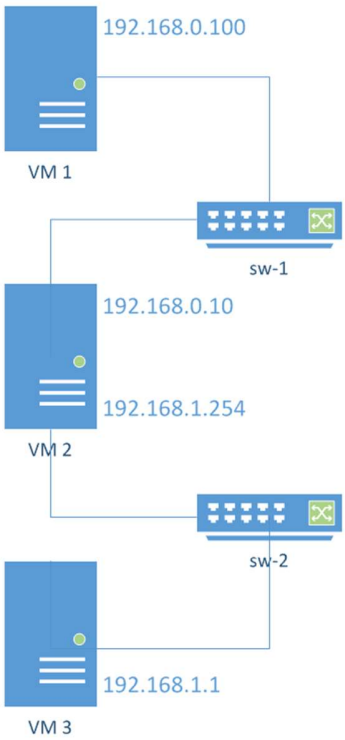

Diagram 1 - Intended network layout. VM1 and 2 are connected in a virtual network. VM 2 and 3 are connected in another virtual network. VM2 will operate as a gateway between the two subnets.

This will have created *virtual switches* **sw-1** and **sw-2** (again, these names are suggestions). Now that both the differential disks and the switches exist, we must create the virtual machine instances, associating each one to the target disk file and switch:

```
$ rnl-virt vm create VM1 SIRS sirs-1.qcow2 --switch sw-1
$ rnl-virt vm create VM2 SIRS sirs-2.qcow2 --switch sw-1 sw-2
$ rnl-virt vm create VM3 SIRS sirs-3.qcow2 --switch sw-2
```

Notice that **VM2** was associated to both *virtual switches*: it shall be connected to both subnets and it will later on be configured as a gateway to allow communication between **VM3** and **VM1**.

Next step, we must start and open each of the virtual machine instances. However, when we open the graphical interface of a virtual machine, the terminal gets blocked, so we must open new terminal tabs (on the RNL computer that is being used) and, for each tab, open a specific virtual machine:

```
$ rnl-virt vm start VM1
```

```
$ rnl-virt vm open VM1

<CTRL+SHIFT+T> to open a new terminal tab

$ rnl-virt vm start VM2
$ rnl-virt vm open VM2

<CTRL+SHIFT+T> to open a new terminal tab

$ rnl-virt vm start VM3
$ rnl-virt vm open VM3
```

You may now login with user "`root`" and password "`inseguro`" on each virtual machine.

Some useful shortcuts are:

- **CTRL+ALT** – release mouse.
- **CTRL+ALT+F** – maximize window.
- **CTRL+ALT+SHIFT+2/1** – open/close the QEMU console.

To change the display resolution in graphics mode, use the application `xLucas` and follow the instructions. To shut down the virtual machine, open a terminal/shell inside the virtual machine and execute the following command:

```
$ sudo /sbin/poweroff
```

3. Configure the machines to use the virtual networks

We will now configure the machines to the virtual network that connects **VM1** to **VM2** and the one that connects **VM2** and **VM3**.

3.1 Configure the virtual network for VM1 and VM2

We will now configure the IP network (supported by *virtual switch* **sw-1**) with static IP addresses. **VM1** and **VM2** will talk using a subnet. We will use the private IP addresses 192.168.0.0/24 (meaning that the subnet mask is 255.255.255.0 – we can have 254 addresses to use - from 192.168.0.1 to 192.168.0.254 – 192.168.0.255 is reserved for broadcast).

Diagram 2 presents and overview of the desired configuration. The IP address of **VM1** will be 192.168.0.100 and **VM2** will be 192.168.0.10. The values ending with 100 and 10 are arbitrary, they could be any value between 1 and 254.

Diagram 2 – Expected layout of the subnet in which VM1 and VM2 will communicate using sw-1.

Follow the configuration procedure for both **VM1** and **VM2**. The presented file contents are just examples that you will need to adjust to the specific needs.

Create a network interface configuration file named `/etc/sysconfig/network/ifcfg-eth0` (or `ifcfg-eth-id-` followed by the MAC address). Using a basic editor – like `vi` - set the IP and *netmask* values.

The following is an **example** of the network interface configuration file contents:

```
BOOTPROTO='static'
MTU=''
REMOTE_IPADDR=''
STARTMODE='onboot'
DEVICE='eth0'
IPADDR='192.168.0.100'
NETMASK='255.255.255.0'
BROADCAST=''
```

Change the file `/etc/sysconfig/network/routes` to define the routing rules.

The following is an **example** of the routing file contents:

```
 127.0.0.0 0.0.0.0 255.0.0.0  lo
default 192.168.0.10  eth0
```

Routes file format details

The `/etc/sysconfig/network/routes` file defines a routing table. Each line is a routing rule defined as "column1 column2 column3" and finally the interface name.

- **Column 1** is used to provide a **destination**, which may be a network address, a host address or the string `default` to specify the default gateway. Whether the given address is a host or a network is determined by the netmask given in the third column. The network mask for a host is always 255.255.255.255. Everything else specifies a network route. Network route means a route to a network, whereas a host route is a route to a single host;
- **Column 2** is either the **IP address of the router** which should be used to reach the destination detailed in the previous column, or the special value 0.0.0.0, which means that all traffic headed to the destination will be given to the device specified in the third column (if a gateway address is given, the device specification is optional);
- **Column 3** contains the **netmask**.

For example, in the following rule, eth0 would be optional, since the special value 0.0.0.0 is being used in the second column:

```
192.168.0.0 0.0.0.0 255.255.255.0  eth0
```

The final line (`default` rule, if it exists) defines the default routing and it is typically used to forward all other IP addresses to the external Internet connection, should it exist.

To finish creating this network, reload the network interfaces of both **VM1** and **VM2**:

```
$ sudo /etc/init.d/network force-reload
```

To check that the configuration is correct:

```
$ /sbin/ifconfig
$ /sbin/route
```

VM2 should now be able to ping **VM1**:

```
$ ping 192.168.0.100
```

And vice-versa, **VM1** should now be able to ping **VM2**:

```
$ ping 192.168.0.10
```

NOTE: On **rnl-virt**, for each virtual machine, check that the MAC addresses are different. For the configuration to work properly, **there cannot be repeated MAC addresses across all virtual machines**. To confirm this, for each virtual machine, execute:

```
$ /sbin/ifconfig
```

In the output, there will be sections beginning with eth0, eth1 and so on (depending on the number of adapters you have). Check a line like '`ethY encap:Ethernet HWaddr XX:XX:XX:XX:XX:XX`'. For example:

```
eth0  Link encap:Ethernet      HWaddr       08:00:27:19:58:A7
```

If there are repeated MAC addresses, use the following command in the machine where you want to change the MAC address, for the adapter that is repeated:

```
$ sudo /sbin/ip link set eth0 address 00:00:00:00:00:11
$ sudo /etc/init.d/network force-reload
```

This would change eth0's MAC address to `00:00:00:00:00:11`.

3.2 Configure the virtual network for VM2 and VM3

We will now configure the IP network (supported by *virtual switch* **sw-2**) with static IP addresses. **VM2** and **VM3** will talk using another subnet. We will use the private IP addresses 192.168.1.0/24. Diagram 3 presents the intended configuration. The IP address of **VM2** should be 192.168.1.254 and the address of **VM3** should be 192.168.1.1. Again, 254 and 1 are arbitrary values between 1 and 254.

Diagram 3 – Expected layout of the subnet in which VM2 and VM3 will communicate using sw-2.

In **VM3** configure eth0 and define the default routing. Here is an example:

```
127.0.0.0 0.0.0.0 255.0.0.0 lo
default 192.168.1.254 dev eth0
```

Do not forget that VM2 will have two ifcfg-ethX files: one for eth0 (which already exists at this point) and another for eth1.

In **VM2** configure eth1: eth0 is connected to sw-1 and eth1 is connected to sw-2. Define routing rules in `/etc/sysconfig/network/routes`:

```
127.0.0.0 0.0.0.0 255.0.0.0 lo
192.168.0.0 0.0.0.0 255.255.255.0  eth0
192.168.1.0 0.0.0.0 255.255.255.0  eth1
default 0.0.0.0 255.255.255.0 eth0
```

To finish creating this network, reload the network interfaces of both **VM2** and **VM3**:

```
$ sudo /etc/init.d/network force-reload
```

To check that the configuration is correct:

```
$ /sbin/ifconfig
$ /sbin/route
```

VM2 should now be able to ping **VM3**:

```
$ ping 192.168.1.1
```

VM3 should now be able to ping **VM2**:

```
$ ping 192.168.1.254
```

3.3 Configure VM2 as gateway

Since **VM2** will be the default gateway for **VM3**, IP forwarding must be enabled. This will allow **VM3** to communicate with machines outside the subnet 192.168.1.X.

Open the file `/etc/sysconfig/sysctl` and set the `IP_FORWARD` value to `yes`;

Load the configurations into the *kernel*:

```
$ sudo /etc/init.d/boot.ipconfig restart
```

Confirm that the flag value was updated to 1:

```
$ /sbin/sysctl net.ipv4.conf.all.forwarding
```

3.4 Configure NAT (Network Address Translation)

During the previous step, the connectivity test between **VM3** with **VM1** failed. Since NAT was not enabled in **VM2**, the host sent the reply to its gateway, which eventually dropped the packet. Use the `iptables` command (`man iptables`) in **VM2** to correct this behaviour. NAT will do the source and destination mapping.

```
$ sudo /usr/sbin/iptables –P FORWARD ACCEPT
$ sudo /usr/sbin/iptables –F FORWARD
$ sudo /usr/sbin/iptables –t nat -F
$ sudo /usr/sbin/iptables –t nat –A POSTROUTING  –o eth0 –j MASQUERADE
```

Test again the connectivity between **VM3** and **VM1** and check that it is working now:

```
$ ping 192.168.0.100
```

4. Monitor network traffic

To monitor the network traffic, we may use **VM2** (or another machine, e.g. a **VM4**, also connected to the network) to run `tcpdump` and capture all network traffic. Make sure you can detect ICMP packets originating at **VM3** and destined to **VM1** (using ping). Use `tcpdump` with options `-X` and `-XX` and identify the IP addresses, MAC addresses and protocol in a given packet.
While still running `/usr/sbin/tcpdump`, open a telnet connection between **VM1** and **VM2** using user "*fireman"* and password "*inseguro*". Verify that you can capture both the username and password with tcpdump.

You have successfully eavesdropped communications… ***But what is the difference between executing*** `telnet` ***from VM1 to VM3 with and without NAT? Use*** `tcpdump` ***to analyse the output and compare the differences.***

5. Gracefully turn off the virtual machines

To gracefully close the virtual machines which you deployed using the **rnl-virt** command, execute the following on **VM1**, **VM2** and **VM3**:

```
$ sudo /sbin/poweroff now
```

Now, on the host RNL machine, the above command should be enough for the machines to power off and their **rnl-virt** windows to close automatically. If so, **we are done!**
If they do not, we must ask to stop the running virtual machine instances launched during this assignment. It may occur that after running these commands, the system will say the machines were already stopped (due to executing the command above). Ask **rnl-virt** to close the virtual machines anyway (if the virtual machines have been shut down, use the `-f` flag for the commands below):

```
$ rnl-virt vm stop VM1
$ rnl-virt vm stop VM2
$ rnl-virt vm stop VM3
```

After all the virtual machines have stopped, if their windows are still open on the host system (RNL computer) you may close their respective windows by pressing the X button on the top-right corner of the window.