Instituto Superior Técnico, Universidade de Lisboa
Network and Computer Security

Lab guide:

Implementation and Analysis of a Virtual Computer Network

VirtualBox version. Revised on 2016-10-10

Goals

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

1. Introduction

The laboratory assignment uses a virtual network consisting of multiple virtual machines supported by Oracle's VirtualBox virtualization software. VirtualBox works with Windows, Mac and Linux operating systems, although specific instructions may vary slightly.
The physical machine is called the **Host** whereas the virtual machines are called **Guests**. The host will be the computer running VirtualBox. Each guest will use a version of Caixa Mágica Linux with low memory requirements.

You can use the manual command to read the documentation about the Linux commands:

```
$ man ln
```

1.1 Create a virtual machine instance

Each virtual machine instance will have its own virtual disk.

The virtual disk image can be obtained at: https://ftp.rnl.ist.utl.pt/sirs/vm-SIRS2014-15.vdi.gz

Create a virtual machine named **vm1** by executing the following steps:

1. Open Virtual Box
2. Create a Virtual Machine:
   - **Machine → New**
   - Name: vm1
   - Operating system: Linux, version: Linux 2.6 (32 bits)
   - Use default value for memory (256 MB)
   - **Do not add a virtual drive**
3. Add the virtual drive as IDE:
   - **Machine → Settings → Storage**
   - Add an IDE Hard Disk
   - Choose existing disk. Add the **.vdi** file downloaded before
4. Test the Virtual Machine:
   - Start. Log in with the user *root* and the password "*inseguro"*
   - If the GUI does not appear, run `startx`.
     i. If `startx` fails, do: `sudo X-configure`
     ii. Then: `cp /root /xorg.conf.new /etc/X11/xorg.conf`
     iii. Finally: `startx`

By default, you can use <**Right CTRL**> to release the mouse pointer from the guest back to the host. To change the display resolution in graphics mode, use the application `xLucas`.

1.2 Copy the virtual machine instance

We will now create a total of 3 virtual machine instances: 1, 2 and 3.
To create a copy, use the Machine - Clone command:

1. Power-off virtual machine 1 (**vm1**)
2. VirtualBox menu: select virtual machine 1
3. Select Machine -> Clone...
4. Change name to vm2. Select "Reinitialize the MAC Address of all network cards", so that the interface MAC address differs on different virtual machines.
5. Select Full Clone that will create several virtual disks. Linked clone also would work but the cloned machine is created from a snapshot of machine 1, leaving a file dependency.
6. Click on Clone

Apply the same procedure to create vm3.

1.3 Create the virtual network

We will now configure the network settings of each Guest.

VM2 will access the **physical network** and the virtual network.

- To configure the access to the physical network, go to Network Adapters and Other Hardware configuration:

1. VirtualBox menu: select vm2 -> Settings... -> Network
2. Network menu: select **Adapter 1**. Click on "Enable Network Adapter" to enable the adapter, if the checkbox is not already checked. **Attached to: "Bridged Adapter".**
3. Name: Select the default network card on Windows ("Intel(R) Pro...", "RealTek...", "Wifi","Ethernet", etc) or the default network interface (eth0, wlan0, etc) on Linux, so that the virtual machine virtual is added to the host virtual network.
4. Expand **Advanced**
    a. Adapter Type: PCnet-FAST III
    b. Promiscuous Mode: Allow All
    c. MAC Address: use default value
    d. Verify that "cable connected" is enabled.

Note: VM1 and VM2 will be connected to the same physical network, so you must follow steps 1-4 above for VM1 as well.

- To configure the **virtual network**:

1. VirtualBox menu: select vm2 -> Settings... -> Network
2. Network menu: select **Adapter 2**. Click on "Enable Network Adapter" to enable the adapter, if the checkbox is not already checked. **Attached to: "Internal Network".**
3. Name: Select the default network card on Windows ("Intel(R) Pro...", "RealTek...", "Wifi","Ethernet", etc) or the default network interface (eth0, wlan0, etc) on Linux, so that the virtual machine virtual is added to the host virtual network. ***Note: if the only option is 'intnet", choose that option.***
4. Expand **Advanced**
    a. Adapter Type: PCnet-FAST III
    b. Promiscuous Mode: Allow All
    c. MAC Address: use default value
    d. Verify that "cable connected" is enabled.

Note: VM2 and VM3 will be connected to the same virtual network, so you must follow steps 1-4 above for VM3, but use Adapter 1 for VM3.

1.4 Configure the IP network

The physical and virtual networks will be different IP subnets.
For the virtual network, we will use the private IP addresses 192.168.1.0/24 (meaning that the subnet mask is 255.255.255.0 – we can have 254 addresses to use - from 192.168.1.1 to 192.168.1.254 – 192.168.1.255 is reserved for broadcast).
The following diagram presents the desired network layout. IP addresses may vary depending on the host network settings and the virtual network settings.

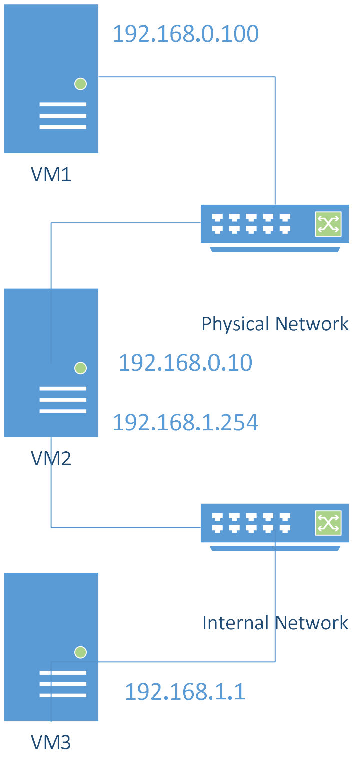

Diagram 1 – Intended network layout. VM1 and 2 are connected in the physical network. VM 2 and 3 are connected in a virtual network. VM2 will operate as a gateway between the two subnets.

On **VM1**, delete any unnecessary `ifcfg-` files in the `/etc/sysconfig/network/` folder:

```
$ sudo rm /etc/sysconfig/network/ifcfg-eth-id-*
```

Afterwards, create a network interface configuration file named `/etc/sysconfig/network/ifcfg-eth0` (or `ifcfg-eth-id-` followed by the MAC address):

```
$ sudo touch /etc/sysconfig/network/ifcfg-eth0
```

Note: since a lot of the files edited are located in the `/etc/sysconfig/network/` directory, it is advised to navigate to it to make command execution easier:

```
$ cd /etc/sysconfig/network/
```

Using a basic editor – like `vi` - set the *IP* and *Netmask* values (you may use the following **example** contents)

```
$ sudo vi /etc/sysconfig/network/ifcfg-eth0
```

Press the 'i' key on the keyboard to enter editing mode in **vi** and write the following content:

```
BOOTPROTO='static'

MTU=''

REMOTE_IPADDR=''

STARTMODE='onboot'

DEVICE='eth0'

IPADDR='192.168.0.100'

NETMASK='255.255.255.0'

BROADCAST=''
```

When you finish writing, press 'Esc' to exit editing mode and then write the following command and press enter (this is done still inside **vi**):

```
:wq
```

Change the file `/etc/sysconfig/network/routes` to define a default gateway address for interface *eth0*. For **example**, change the file contents to the following:

```
 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.

Reload the network interfaces:

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

To check that the configuration is correct:

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

Apply the same procedure to configure **VM2** with IP 192.168.0.10, as indicated in **Diagram 1**. To do this, in **VM2** delete any unnecessary `ifcfg-` files in the `/etc/sysconfig/network/` folder:

```
$ sudo rm /etc/sysconfig/network/ifcfg-eth-id-*
```

Afterwards, create a network interface configuration file named `/etc/sysconfig/network/ifcfg-eth0` (or `ifcfg-eth-id-` followed by the mac address):

```
$ sudo touch /etc/sysconfig/network/ifcfg-eth0
```

Using a basic editor – like `vi` - set the *IP* and *Netmask* values (you may use the following example contents)

```
$ sudo vi /etc/sysconfig/network/ifcfg-eth0
```

Write the following content and save the changes:

```
BOOTPROTO='static'
MTU=''
REMOTE_IPADDR=''
STARTMODE='onboot'
DEVICE='eth0'
IPADDR='192.168.0.10'
NETMASK='255.255.255.0'
BROADCAST=''
```

Change the file `/etc/sysconfig/network/routes` to define a default gateway address for interface *eth0*. For example, change the file contents to the following:

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

Reload the network interfaces:

```
$ 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** (and vice-versa):

```
$ ping 192.168.0.100
```

Configure **VM3**, with IP 192.168.1.1 and a rule in the `/etc/sysconfig/network/routes` file to use **VM2** as default gateway.

Configure **VM2** with IP 192.168.1.254 on eth1.

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

Define routing rules in `/etc/sysconfig/network/routes` for **VM2**:

```
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
```

After reloading the network configurations, **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
```

1.5 Configure a 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.

On VM2, 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
```

1.6 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
```

1.7 Monitor network traffic

To monitor network traffic, use **VM2** 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 **tcpdump**, open a **telnet** connection between machines 1 and 2 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.***