Name 1: Léo Lucien Alvarez
Name 2: Francisco Javier Blázquez Martínez

COM-407: TCP/IP NETWORKING

LAB EXERCISES (TP) 1
INTRODUCTION TO MININET

October 2nd, 2020
Deadline: October 14, 2020 at 23.55 PM

Abstract

In this TP you will use Mininet, a virtualized environment used to emulate hosts and switches. Mininet is already installed on the virtual machine created in the previous lab. You will perform basic network configuration tasks and learn to mount a man-in-the-middle attack. For a bonus, you will have the chance to evaluate how the performance of Mininet scales with the number of emulated hosts and routers.

1 BACKGROUND KNOWLEDGE: IPERF

Iperf is a widely used tool for network performance measurement and tuning by measuring the maximum network throughput a server can handle. Iperf has both client and server functionalities, and can create data streams to measure the throughput between the two ends in one or both directions. The data streams can be either TCP or UDP.

1.1 TCP CLIENT AND SERVER

In order to launch an iperf TCP server use:

```
$ iperf -s
```

In order to launch an iperf TCP client use:

```
$ iperf -c iperf_server_ip_address
```

1.2 UDP Client and Server

In order to launch an iperf UDP server use:

```
$ iperf -s -u
```

In order to launch an iperf UDP client use:

```
$ iperf -c iperf_server_ip_address -u
```

There are also publicly available iperf servers, in this lab you will use `iperf.he.net`.

Q1/ Start Wireshark on your **host** (i.e. personal computer) and **guest** (i.e. virtual machine inside personal computer) machines. Then connect a TCP iperf client on your guest machine to `iperf.he.net` (`iperf -c iperf.he.net`. **Note: If you cannot connect with this command, try to change the server port to listen on using the `-p` option: `iperf -p port_number -c iperf.he.net` with port_number equal to** 5201**,** 5202**, etc.**) Compare the traffics captured on your host and guest machines. What are the source and destination IP addresses and port numbers of the packets coming from the server? Are there any differences in IP addresses and port numbers between two captures? If you see any differences, explain why.

[A1] On Host:
Source=216.218.207.42

2 Mininet: A Virtualized Environment for Networking

Mininet "creates a realistic virtual network, running real kernel, switch and application code, on a single machine...". It is a network emulator that can be used to deploy a full network topology on your computer. It runs a collection of hosts, switches, and links on a single Linux kernel. For this, it uses lightweight virtualization to make a single system look like a computer network, running the same kernel and user code.

In this lab, we prefer Mininet to other emulation platforms such as GNS3 due to its light-weight and scalability features. To learn more about the advantages and limitations of Mininet over other emulation platforms, take a look at this seminar from SIGCOMM 2014.

2.1 Mininet Tutorial: Command Line Interface

Here is a short tutorial on Mininet. The quickest way to familiarize with Mininet is through its command line interface. The command line interface is used to view the configuration of hosts and switches, and run applications or test on the configured network. It cannot be used to modify the topology, i.e., add or remove nodes or links. To start Mininet, enter:

```
$ sudo mn
```

It starts with the default `minimal` topology that comprises 2 hosts, 1 switch and a software-defined-networking controller (more on this in Lab 4). Alternatively, you could start mininet with one of the standard topologies using the `topo` parameter: `sudo mn --topo=tree`. We will learn about more advanced ways to create and configure networks in Section 2.2. Once Mininet is started, the command prompt changes from `$` to `mininet>`.

At the Mininet prompt, enter `help` to see a list of acceptable commands. Next, we will go through a few important ones.

To see the nodes, enter `nodes`. You will see the following output.

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
```

To see the network connections, enter `net` and to see the information about the nodes, enter `dump`.

Q2/ What are the different links in the network?

[A2] Answer interpreting the following output:

mininet> net

The hosts and switches in Mininet share a common UNIX kernel and file system. As a result, any application or script installed on the Mininet VM is accessible by all the devices. Furthermore, the UNIX command line interface can be used for configuration and running applications. This architecture makes using Mininet very intuitive as we shall see. To execute a command on a particular host or a switch, simply append the UNIX command to the name of the device: `<node/switch> <cmd>`. For instance, to check the list of files in the current directory on host `h1`, use

```
mininet> h1 ls
```

As the hosts share the same filesystem, you can verify that the output for the corresponding command on host `h2` is the same.

Q3/ What command will you use to check the network interfaces of the host h2? How many interfaces does host h2 have? What is/are its/their IP addresses?

[A3] Answer interpreting the following output, usinf ifconfig command:

mininet> h2 ifconfig

Mininet automatically substitutes IP addresses for host names. So, to ping a host `h2` from host `h1`, use:

```
mininet> h1 ping h2
```

Now, in a different terminal, start Wireshark by using the command `sudo wireshark`. In wireshark, choose the interfaces `s1-eth1` and `s1-eth2` and start a capture to see the messages generated by the above ping command.

Q4/ What is the source IP address of the ICMP echo requests sent from h1 to h2?

[A4] The source IP address of the ICMP echo request is the h1 IP address : 10.0.0.1. This is normal as the ping corresponds to packets sent from source h1 and Mininet automatically substitutes IP addresses for host names.

Besides the traditional ping, Mininet also offers the `pingall` command to check connectivity between all hosts. This is particularly handy in checking if a newly configured network is correctly configured.

To exit, use `mininet> exit`. If Mininet crashes for some reason, you can clean up the topology using the following command.

```
$ sudo mn -c
```

For more information, you can use this walk-through.

2.2 MININET TUTORIAL: PYTHON API

While the command line interface is very useful for configuring a few nodes on-the-fly, it is cumbersome for setting up larger networks. For setting up larger networks, Mininet provides a Python-based API that can be used to create and modify a network with hosts and switches, run tests on the switches and collect results, all with using a single script. Next, we shall see how to get started with this API.

NOTE This tutorial will only cover the Python API needed for Mininet. The commands discussed here are sufficient for understanding and modifying a piece of Python code. This level of understanding is sufficient for Lab 1. You will not be able to write a Python code from scratch based on this tutorial. However, we will visit Python in detail again in Lab 3.

Download the `Lab1` folder from Moodle, in this folder you will find scripts folder, copy it to your shared folder on virtual machine. Here is a sample code for a Mininet network with two hosts interconnected by a switch. You can also find this code in the file `firstNetwork.py` in the folder `/media/lca2/shared/Lab1/scripts`.

The code is annotated with comments for your understanding. Comments in Python starts with `#` (for single line comments) or are enclosed in `"""..."""` (for multi-line comments).

```
#!/usr/bin/python

"""
This example shows how to create a Mininet object and add nodes to
it manually.
"""
#Importing Libraries
```

```
from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info

#Function definition: This is called from the main function
def firstNetwork():

    #Create an empty network and add nodes to it.
    net = Mininet()
    info( '*** Adding controller\n' )
    net.addController( 'c0' )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.0.0.1' )
    h2 = net.addHost( 'h2')

    info( '*** Adding switch\n' )
    s12 = net.addSwitch( 's12' )

    info( '*** Creating links\n' )
    net.addLink( h1, s12 )
    net.addLink( h2, s12 )

    info( '*** Starting network\n')
    net.start()

    #This is used to run commands on the hosts

    info( '*** Starting xterm on hosts\n' )
    h1.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h1 &')
    h2.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h2 &')


    info( '*** Running the command line interface\n' )
    CLI( net )

    info( '*** Closing the terminals on the hosts\n' )
    h1.cmd("killall xterm")
    h2.cmd("killall xterm")

    info( '*** Stopping network' )
    net.stop()

#main Function: This is called when the Python file is run
if __name__ == '__main__':
    setLogLevel( 'info' )
    firstNetwork()
```

Note the two ways of adding a host. For adding `h1`, we specify the name of the host and the ip address in the parameters. For adding `h2`, we specify only the name of the host. When the IP address is not specified, Mininet gives it a default IP Address. In this lab, we will use the later approach to create a host. Once we create a host, we will configure it through standard Linux commands through the terminal.

To start Mininet with the custom topology, nagivate to the directory in which `firstNetwork.py` is present and in the command line enter:

```
$ sudo python3 firstNetwork.py
```

Besides the output on your terminal, you should see two new terminals, one each for the hosts `h1` and `h2`, respectively. From these terminals, you can access `h1` and `h2` just like normal UNIX hosts. For instance, in the terminal for `h2`, enter `sudo wireshark` to open wireshark.

Q5/ List down the names of the interfaces that are available for capture.

[A5] The interfaces available for capture are:
- h2-eth0 for host h2 and h1-eth0 for host h1 (Main interface with network)
- loopback: lo (loopback interface, mainly used for testing)

To close all the terminals and the network, in Mininet, enter:

```
mininet> exit
```

More information on the Python API can be found here.

3 Basic connectivity within a LAN

In this section you will learn how to configure machines in order to achieve connectivity in the IP layer within a Local Area Network (LAN). You will also use *Wireshark* to observe which traffic is sent (or not) in some scenarios. This will help you to find what is configured correctly and what has to be configured additionally in order to achieve the connectivity.

3.1 Wiring and addressing scheme

In Mininet, create a configuration with four hosts PC1, PC2, PC3 and PC4 and three switches s14, s24 and s34 as shown in Figure 1. For this purpose, use the file `lanTopology.py` in the folder `/media/lca2/shared/Lab1/scripts`. Parts of the topology file have already been written for you. Fill in the rest **without specifying the ip addresses of any of the interfaces** (you will do this separately through commands) and start the network. Once completed, you should see four terminal windows labeled PC1, PC2, PC3 and PC4. We will use these windows for the remainder of the lab.

3.1.1 IPv4 addressing

For IPv4 addressing, we will use the private IPv4 address space `10.10.0.0/16`.

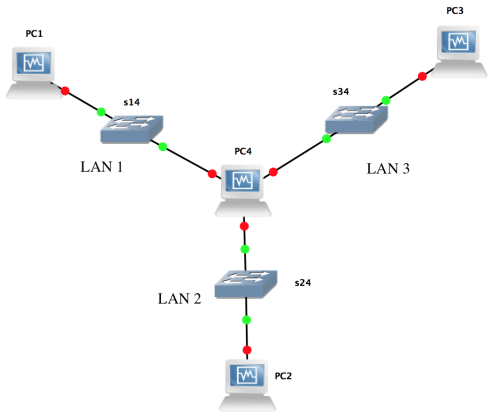

Figure 1: Basic configuration

- For the local network connecting PC4 to PC1 (LAN1), we will use the IPv4 network address prefix `10.10.10.0/24`. The host identifier (the fourth byte) should be `1` to indicate the PC1 and `4` for PC4.
- For the local network connecting PC4 to PC2 (LAN2), we will use the IPv4 network address prefix `10.10.20.0/24`. The host identifier (the fourth byte) should be `2` to indicate the PC2 and and `4` for PC4.
- For the local network connecting PC4 to PC3 (LAN3), we will use the IPv4 network address prefix `10.10.30.0/24`. The host identifier (the fourth byte) should be `3` to indicate the PC3 and and `4` for PC4.

Q6/ How many IPv4 networks does PC4 belong to? Name the type of the network(s).

[A6] PC4 belongs to three local IPv4 networks (LAN). Here is a part of the response to net command in mininet CLI, showing the three links of host PC4 with the other three:
PC4 PC4-eth0:s14-eth2 PC4-eth1:s24-eth2 PC4-eth2:s34-eth2

Q7/ Write down the IPv4 addresses you will use for the interfaces according to this addressing scheme.

[A7.a] PC1-eth0:	10.10.10.1	[A7.b] PC2-eth0:	10.10.20.2
[A7.c] PC3-eth0:	10.10.30.3	[A7.d] PC4-eth0:	10.10.10.4
[A7.e] PC4-eth1:	10.10.20.4	[A7.f] PC4-eth2:	10.10.30.4

3.1.2 IPv6 ADDRESSING

Several IPv6 addresses can be obtained by each interface. In the virtual environment we use private addresses, called Unique Local Addresses (ULAs). Such addresses can be used only inside a private domain and are ignored in the public internet. We will use the prefix `fd24:ec43:12ca::/48`, which is registered to EPFL for the SmartGrid project.

- For LAN 1, we will use the IPv6 subnet `fd24:ec43:12ca:c001:10::/80` and addresses `fd24:ec43:12ca:c001:10::X`, with `X` equal to `1` for PC1 and to `4` for PC4.

- For LAN 2, we will use the IPv6 subnet `fd24:ec43:12ca:c001:20::/80` and the addresses `fd24:ec43:12ca:c001:20::X`, with `X` equal to `2` for PC2 and `4` for PC4.
- For LAN 3, we will use the IPv6 subnet `fd24:ec43:12ca:c001:30::/80` and the addresses `fd24:ec43:12ca:c001:30::X`, with `X` equal to `3` for PC3 and `4` for PC4.

Q8/ Write down the IPv6 addresses you will use for the interfaces according to this addressing scheme.

[A8.a] PC1-eth0: fd24:ec43:12ca:c001:10::1
[A8.b] PC2-eth0: fd24:ec43:12ca:c001:20::2
[A8.c] PC3-eth0: fd24:ec43:12ca:c001:30::3
[A8.d] PC4-eth0: fd24:ec43:12ca:c001:10::4
[A8.e] PC4-eth1: fd24:ec43:12ca:c001:20::4
[A8.f] PC4-eth2: fd24:ec43:12ca:c001:30::4

In addition to the Unique Local Addresses, every IPv6 interface also has a link-local address due to the IPv6 Stateless Address Autoconfiguration (SLAAC). Link local addresses are allocated automatically and do not require any configuration; they can be used only for communication in the same LAN.

Link-local addresses are all in the `fe80::/64` subnet. With this addressing scheme, all the network cards in the world are in the same subnet (you can see that the routing table contains by default this subnet)! For this reason, two machines on the same link can communicate directly without the need to configure routing (the host part is unique because it is derived from the MAC address, which is supposed to be unique).

To determine the MAC address for the `PC1-eth0` interface, look at the `link/ether` field after issuing in a terminal the following command:

```
$ ip link show PC1-eth0
```

Q9/ Write down the MAC address of the interfaces.

[A9.a] PC1-eth0: 8a:65:59:33:fd:62
[A9.b] PC2-eth0: ce:0a:92:6a:04:0e
[A9.c] PC3-eth0: ee:94:ff:8a:14:b9
[A9.d] PC4-eth0: 0e:24:19:9e:f9:53
[A9.e] PC4-eth1: 7a:52:ea:e2:12:3a
[A9.f] PC4-eth2: 9a:e1:46:7a:d6:8f

The SLAAC IPv6 link-local address is formed as follows. The address prefix is `fe80::/64`. For the host part, we use a 64-bit interface identifier in modified EUI-64 format. It is derived from the interface's MAC address by inverting 7th bit and inserting `ff:fe` in the middle. An illustration is given in Figure 2 adapted from the Wikipedia page dedicated to IPv6 addresses.

Q10/ Compute the IPv6 link-local addresses for the interfaces below.

[A10.a] PC3: FE80:0000:0000:0000:EC94:FFFF:FE8A:14B9
[A10.b] PC4, LAN3(eth2): FE80:0000:0000:0000:98E1:46FF:FE7A:D68F

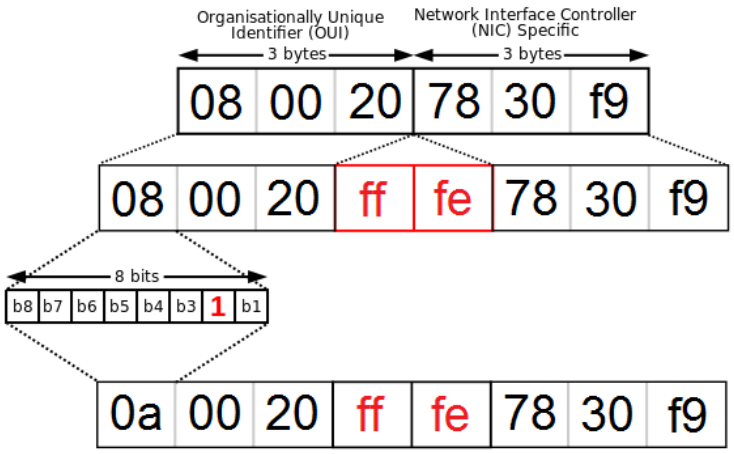

Figure 2: Formation of EUI-64 starting from MAC address of an interface.

3.2 Are you connected? Use wireshark to see what happens.

In order to have access to the commands that manipulate the network interfaces, you need to be logged in as superuser (root).

To see the network interfaces of a PC, type in its terminal:

```
$ ip link
```

Bring up the interfaces of PC4 by running in PC4's terminal:

```
# ip link set PC4-eth0 up
# ip link set PC4-eth1 up
# ip link set PC4-eth2 up
```

Similarly, bring up the interfaces of PC1, PC2 and PC3.

As seen in Lab0, the `Wireshark` program is used to capture the incomming and outgoing traffic on a network interface. You will now inspect what happens at packet level.

Start a `Wireshark` capture on all interfaces of PC1 and PC4. Since the Wireshark process blocks the command line, you need to run it in detach mode such that command line will not be frozen. For that use:

```
# wireshark &
```

We will first test what happens when pinging unassigned addresses.

To test IPv4 connectivity run a *ping* to the unconfigured router from a terminal on `PC1`:

```
$ ping 10.10.10.4
```

Q11/ Explain what happens.

[A11] Ping try to get the Mac addresses because it is a local network using switches. But the ipv4 address 10.10.10.4 is not assigned yet and is not linked to any interface and thus there is no no MAC address response frome any interface.

Test IPv6 connectivity by running a *ping* to the link-local IPv6 address of `PC4-eth0` interface (the one that is connected to PC1) from a terminal on PC1. (**Note that MAC addresses of interfaces are changed every time when you start Mininet. Make sure that you use proper MAC addresses in order to get the link-local IPv6 address.**)

```
$ ping6 <link_local_address_of_PC4_eth0> -I PC1-eth0
```

Q12/ Describe the traffic you observe with Wireshark.

[A12] Ping works.
Packets are captured only in PC1-eth0 and PC4-eth0 interfaces.
We see also neighbor solicitation/advertisement packets ??

Now ping the link-local IPv6 address of `PC4-eth2` interface (the one that is in LAN 3) from a terminal on PC1.

```
$ ping6 <link_local_address_of_PC4_eth2> -I PC1-eth0
```

Q13/ Describe and explain the differences (if any) between the output from the two ping commands.

[A13] Destination unreachable, address unreachable.
Aquí vemos sólo neighbor solicitation pero no neighbor advertisement.

3.3 Configuring and testing of the LAN

In the previous question you observed that there exists only IPv6 connectivity within the LAN. This is a consequence of IPv6 Stateless Address Autoconfiguration. However, these addresses are not globally routable. To obtain full IPv6 connectivity and IPv4 connectivity some additional steps are required.

3.3.1 LAN configuration

The first step in order to achieve full connectivity from your machine is to configure its network interface and to assign an IP address to the interface. This is true for both IPv4 and IPv6 as link-local IPv6 addresses are not routable on the Internet.

To visualize the IPv4 and IPv6 routing tables of the machine, on PC1 type:

```
$ ip route
$ ip -6 route
```

Q14/ Write down the routing table entries.

[A14] mininet> PC1 ip route
10.0.0.0/8 dev PC1-eth0 proto kernel scope link src 10.0.0.1
The network 10.0.0.0/8 is available on PC1-eth0 with a scope of link, which means that the

To configure your network interfaces[1] and assign an IPv4 and an IPv6 address to `PC1-eth0` interface, open a `root` terminal (`su`) and type:

```
# ip addr add 10.10.10.1/24 dev PC1-eth0
# ip -6 addr add fd24:ec43:12ca:c001:10::1/80 dev PC1-eth0
```

Make sure to delete previously existing IP addresses. You may check for those and delete them with commands similar to the following:

```
# ip addr show
# ip addr del <existing_ip_address> dev PC1-eth0
```

Q15/ What are the routing table entries after assigning the new ip addresses?

[A15] mininet> PC1 ip route
10.10.10.0/24 dev PC1-eth0 proto kernel scope link src 10.10.10.1
mininet> PC1 ip -6 route

Q16/ For the IPv4 addresses assigned to PC1-eth0, identify the host and subnetwork portions. Why is it important for the PC to know this information?

[A16] The command « PC1 ip adds show » in mining CLI gives us, among other informations, the following line under the PC1-eth0 interface paragraph:
inet 10.10.10.1/24 scope global PC1-eth0.

From PC1, run another `ping` to PC4's IPv4 interface:

[1] Tip: To avoid typing this again the next time you boot the PCs save the commands in a script file that you can execute later on. Make use of the `shared` folder and the copy-paste functionality.

```
$ ping 10.10.10.4
```

Q17/ What do you observe on Wireshark? Explain

[A17] #describe wireshark# As for question 11, we are unable to reach PCA IPv4 10.10.10.4 because this IP address is not set and is not related to any interface. So it is not possible to get MAC address for the host with IP address given (even if it is in our subnet) because no

Configure the interfaces of PC4 and the one of PC2 and PC3 according to the addressing scheme.

To verify the configuration, type:

```
$ ip addr show
```

Try again pinging PC4 from PC1. It should work now. Do the same for PC2 and PC3 to verify connectivity to PC4.

You can see the mapping between IPv4 addresses and MAC addresses. Take a look at the ARP table of the PC1 workstation:

```
$ ip neigh
```

At this point you should have both IPv4 and IPv6 connectivity within each LAN.

3.4 ROUTING PACKETS

Try to reach any of the `PC4-eth1` or `PC4-eth2` interfaces of PC4 from PC1 (i.e., anyone that is not directly connected to PC1):

For IPv4:

```
$ ping 10.10.20.4
$ ping 10.10.30.4
```

For IPv6:

```
$ ping fd24:ec43:12ca:c001:20::4
$ ping fd24:ec43:12ca:c001:30::4
```

Q18/ Does it work? Why?

[A18] Ping does not work. « Network unreachable ». We don't even see packets because it's detecting that those directions are out of our network and we don't have specified to host PC1 where to send the packets when trying to reach other networks which IP are not in the subnet

On PC1, add an IPv4 default route (i.e., "configure default gateway") and do the same for IPv6:

```
# ip route add default via 10.10.10.4
# ip -6 route add default via fd24:ec43:12ca:c001:10::4
```

Take the `eth2` interface of PC4 and test again if it is reachable from PC1 (both in IPv4 and IPv6).

In PC4 start a `Wireshark` capture of `PC4-eth2`interface to observe the traffic of the link between the machines belonging to LAN3.

From PC1, try to ping PC3, and observe the traffic on PC4:

```
$ ping 10.10.30.3
```

Frustratingly, it does not work. The ICMP messages do not get sent out by PC4. By default, PC4 does not forward IP traffic from one LAN to another. You need in addition to enable IP forwarding on PC4, which allows PC4 to work as router, by typing the following command (if you use copy-paste from this document, most likely the underscore "_" character will not be copied properly):

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Retry to ping PC3 from PC1. It still does not work, but now you know enough to fix it. *Hint:* Check again the traffic on the two links with Wireshark and see which packets do not get sent.

Q19/ Which command(s) you need to fix the problem?. On which PC did you apply them?

[A19] We can see with wiresark that the ICMP ping request packets reach PC3. The problem is that host PC3 has no default gateway configured and thus it does not know where to send the ping ICMP replies when the destination IP is not in its subnet 10.10.30/24. We can solve this like for

Now, configure your network so as to be able to ping PC1, PC2 and PC3 between each other both in IPv4 and IPv6. For that, you will need to enable IPv6 forwarding using:

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

Q20/ Perform the following pings simultaneously: PC2 from PC1, PC3 from PC2, and PC1 from PC3. Comment on the round-trip times that you observe.

[A20] See the results at the end.
For each host couple, the first ping is always slower than the second, which is slower than the following. This is because source host has to trace its route to the destination before sending

Q21/ Do an iperf test between PC1 and PC2 (PC1 run as server) using TCP and UDP. Report the performance (bandwidth values) you obtain with TCP and UDP. Do the port numbers for server and client are changing if you repeat experiment several times?

[A21] Answer by interpreting Q21.png image (output of terminals).

4 CREATING A MAN-IN-THE-MIDDLE ATTACK WITH IP TABLES

In this section you will have an introduction to `iptables`, which will be useful in the next labs. We will work in IPv4 only, and we will use the same working configuration of section 3.4. For your convenience, this configuration can be created directly by running the python script `lanConfig.py`. We will use a man-in-the-middle (MITM) attack example to illustrate iptables, so let's re-label PCs accordingly to their new function (check fig. 3): PC2 is Alice, the naive internet user who is trying to access her bank account; PC1 plays the role of the bank server; PC4 is Alice's home router, which will be hacked to our malicious purposes; and finally PC3 is the attacker's (your) private server where you will redirect Alice's bank transactions and steal her money.

Let's assume that a successful `ping` between two devices is equivalent to a successful money transfer between them. Your goal (as a hacker) is to make Alice and her bank believe they have a successful and point-to-point connection (blue line in fig. 3) while in reality communication goes all the way to your malicious server (PC3) whenever there is a transaction between Alice and the bank (red line in fig. 3). The rule is we can only make configuration changes in Alice's home router and in our own malicious server (PC4 and PC3 respectively).

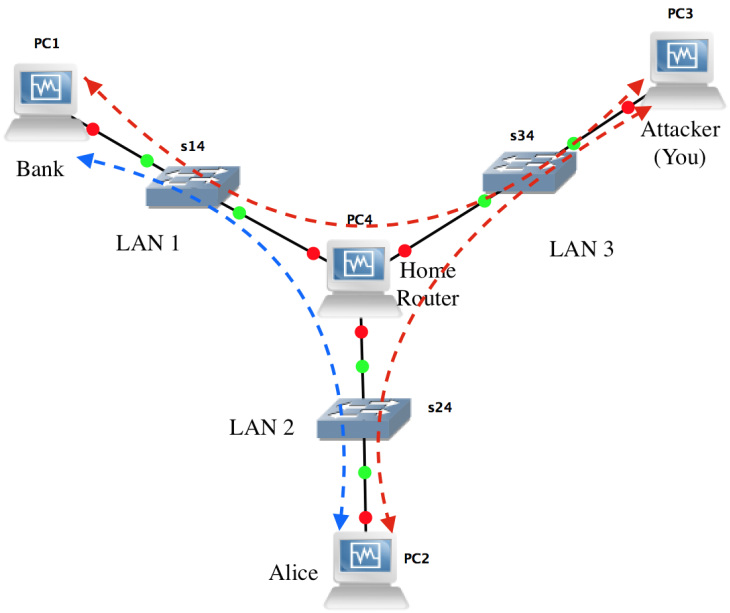

Figure 3: Basic configuration with modified labels

4.1 SETTING UP THE CONFIGURATION

In order for our attack to work properly, we will disable *Reverse path filtering* functionality on Alice's home router. Reverse path filtering (RPF) is a security mechanism where whenever the machine (PC4 in our case) receives a packet, it will first check whether the source IP address of the received packet is reachable through the interface it came in. If it is reachable it will accept the packet; if it is not it will drop it. For more information regarding RPF, here is a link explained by Sarath Pillai (`http://www.slashroot.in/linux-kernel-rpfilter-settings-reverse-path-filtering`):

To disable RPF for IPv4 in all Alice's home router's interfaces, run the script `mitmConfig.sh` script in the folder `/media/lca2/shared/Lab1/scripts` on PC4 terminal.

Finally, we will need to enable IPv4 forwarding on the malicious server (PC3). Do this with the command learnt from previous sections.

4.2 IP TABLES

The Linux kernel contains a packet filter framework called `netfilter` which enables a Linux machine to masquerade source or destination IP addresses. The command used to do this is called `iptables -t nat`, and it manages the table that contains rules regarding address masquerading. This table has two important types of rules:

(i) `PREROUTING`: responsible for packets that just arrived at the network interface, implying rules *before* any routing decision has been made.
(ii) `POSTROUTING`: responsible for packets with a recipient *outside* the linux machine, implying rules before the packet leaves through the network interface (after routing rules).

In this section of the lab, we will learn how to use both rules in order to carry out our MITM attack.

Start a wireshark capture of interface `PC3-eth0` in PC3. Do a ping from Alice's PC (PC2) to the malicious server (PC3). Execute the following command on Alice's home router (PC4):

```
# iptables -t nat -A POSTROUTING -o PC4-eth2 -j MASQUERADE
```

In this command,

- `iptables -t nat` is the command that modifies the masquerade table of the `netfilter`.
- `-A POSTROUTING` says to append a rule to the `POSTROUTING` rules (`-A` stands for `Append`).
- `-o PC4-eth2` states that this rule is valid for packets that leave on interface `PC4-eth2` (`-o` stands for `output`).
- `-j MASQUERADE` states the action that should take place is to "masquerade", i.e., replace source ip address.

Q22/ Do a ping again from Alice to the malicious server and check in wireshark the differences between packets. Write-down any difference in source and destination IP addresses.

[A22]

If you want to remove any configuration from the `iptables` or if you want to flush the mapping or "masquerading" table type the following commands:

```
# iptables -F
# iptables -t nat -F
```

Verify that iptables is disabled by doing another ping from Alice to the malicious server.

Execute the following command on Alice's home router (PC4):

```
# iptables -t nat -A PREROUTING -d 10.10.10.1 -j DNAT --to-destination
10.10.30.3
```

Q23/ Do one more ping from Alice to the bank and check in wireshark the source and destination IP addresses. Based on your observations, what are the use cases for the "*-j MASQUERADE*" and "*-j DNAT*" configuration options?

[A23]

Flush again the iptables with the commands provided in this section.

4.3 Configuring the MITM attack

Let's hack Alice's home router. First let's send the traffic targeted to the bank to go to the malicious server. Since Alice may have other internet activity with high bandwidth consumption (e.g. online gaming), and since Alice may have classmates working with her, we want to select only the traffic we are interested in (*ICMP* packets), and coming only from Alice's PC IP address (`10.10.20.2`). Keep doing the wireshark capture on the malicious server.

Q24/ Write down the command(s) required to this end. If you need help, check the `iptables` howto for masquerading (NAT):
(`http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO-6.html`)

[A24]

Check with wireshark that you have successfully redirected the Bank's traffic from Alice's home router to your private (and malicious) server. Now we need to configure such server (PC3) in order to receive the traffic from Alice's home router, masquerade it, and send it to the bank. To this end, we need to replace the destination IP address of the packet with the IP address of the bank server `10.10.10.1`, and modify the source IP address with your server's IP address `10.10.30.3`.

Start a new wireshark capture on malicious server's `PC3-eth0` interface and on Bank's `PC1-eth0` interface

Q25/ Propose the iptables command(s) that needs to be added to the malicious server in order to complete this task. (**Make sure that IP forwarding on PC3 is enabled.**)

[A25]

On the malicious server (PC3) you should see packets from `10.10.20.2` to `10.10.30.3` (and viceversa), and from `10.10.30.3` to `10.10.10.1` (and viceversa).

Additionally, on Bank's server (PC1) you should see packets coming from `10.10.30.3`. Since this is something that the bank could detect as malicious (e.g. it has an access-control list allowing connections only from its customers' IP addresses), propose the iptables command that you need to configure on Alice's home router (PC4), which is required to masquerade the malicious server with Alice's IP address `10.10.20.2`.

Q26/ Write down the command(s) here

[A26]

That's it, you have successfully mounted a man-in-the-middle attack. However, as you can see, in the current attack, the malicious server bounces the traffic back to the bank and the host PC3 does not receive it.

Q27/ Now, let us assume that there is a program at PC3 that modifies the traffic before bouncing it back to the bank. What should you change in current configuration on PC3 iptables in order to let the traffic come to the program that runs on PC3 host?

[A27]

To conclude the lab just answer the final questions:

Q28/ Can this attack be mounted by just tampering the ip routing table of Alice's home router (PC4)?

[A28]

Q29/ In our lab environment, is there any way (other than checking Alice's empty bank account) that Alice could notice that she is under attack?

[A29]

5 MININET: ANALYZING LATENCY VERSUS NUMBER OF NODES (BONUS)

The following is a bonus section, and is completely optional. It will give you a chance to apply some of what you learned in this lab about Mininet, and it will allow you to gain a better understanding of how Mininet works, in addition to gaining more experience using it, which will be helpful for future labs.

We will be testing how the performance of Mininet scales with the number of nodes. Specifically, as the number of hosts, routers, and switches increases, how is the latency of the communication between these nodes affected?

In order to answer this question, you will need to create several scenarios, each with a Mininet topology in which the number of nodes, and the number of links between them, vary. Of course, this depends on your VM setup, and how much resources (memory, CPU) you assigned to the VM.

One example scenario is a star topology: a router in the center with N interfaces, each interface connected to a switch that is connected to a host. In such a scenario, each host is in a different LAN, and each pair of hosts is separated by the router. How do communication latencies (ping or otherwise) change as the number of hosts N increases from 2 to 5 to 10 to 100?

We ask you to analyze up to two scenarios: either one scenario other than the star topology, or two scenarios including the star topology.

What to submit A PDF file containing a small description of each scenario you analyzed, along with the results (plots, tables) you obtained, and a snippet of the source code used to generate the topology. Put this PDF file along with the PDF file of this report in the same zip folder `Lab3_Name1_Name2`, and submit this folder.

Grading The bonus part will be graded separately from the lab, and you will receive a grade between 0-10. This grade will be part of your research exercise grade.