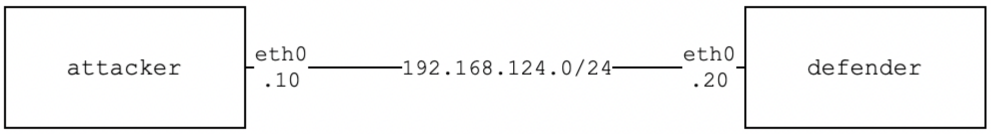
Network Security
Assignment: Data Link Layer

MAC-address spoofing [40 points]

Given the two Docker containers in the image below, the defender container tried to protect itself from the attacker container by only allowing traffic from the MAC-address: `28:B7:D5:50:61:D1`.



Your goal is to evade this defensive measure by spoofing the MAC address of the attacker container's eth0 interface. Before starting with the assignment, make sure to unzip the `mac_spoofing.zip` file and navigate to the resulting directory. Roll the environment out using the following Docker commands.

```
docker-compose build
docker-compose up -d
```

To connect to the attacker container, execute the command below.

```
docker exec -ti attacker bash
```

After finishing the assignment, you can destroy the environment using the command below. First, make sure to copy your developed files outside the container. Otherwise, your assignment will be lost.

```
docker-compose down
```

Goal

Your goal is to produce a Python3 script called `spoof_mac.py`, which accepts two arguments. The first argument should be the name of the interface you are going to spoof. The second argument contains the MAC-address in colon-hexadecimal notation.

```
python3 spoof\_mac.py eth0 28:B7:D5:50:61:D1
```

Apart from spoofing the MAC address, you should also generate four ICMP echoes towards IP address 192.168.124.20. The result of these ICMP echoes should be returned as the output of the script. We expect the following output.

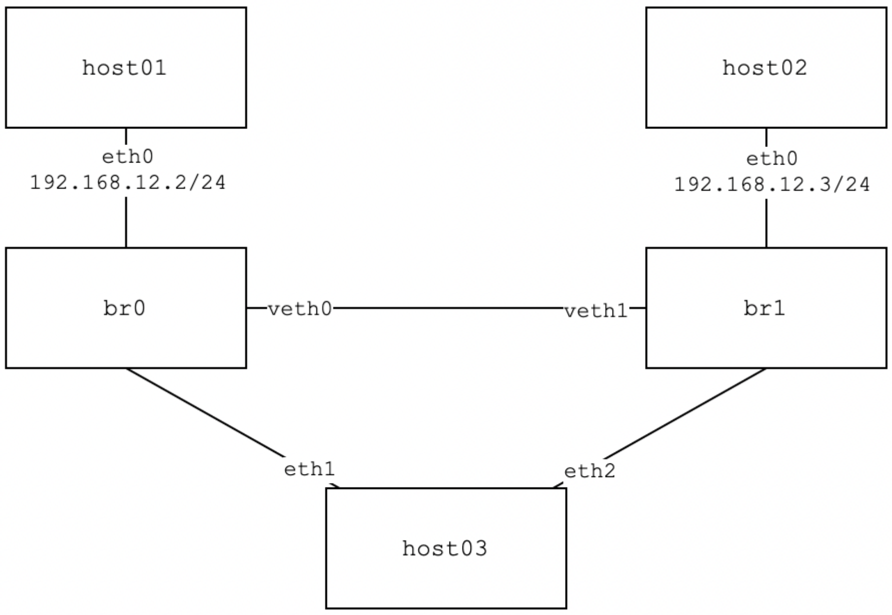
```
Got ICMP-reply to ICMP-echo - 1
Got ICMP-reply to ICMP-echo - 2
Got ICMP-reply to ICMP-echo - 3
Got ICMP-reply to ICMP-echo - 4
```

Submission Instructions

Your script should run on the attacker container. After completing the assignment, you should only submit the resulting script. A `requirements.txt` file should be provided if you use any modules from the Python Package Index (pip).

STP manipulation [60 points]

Consider the topology in the image below.



Both br0 and br1 have STP enabled. Fortunately, host03 is connected to both bridges. Furthermore, the bridge ports to which host03 connects aren't configured to reject *Bridge Protocol Data Units* (BPDUs). Use host03 to hijack the role of the STP root bridge. Use the lowest possible priority value to make sure you win the STP root bridge election process from br0 and br1. This assignment aims to make sure that any traffic between host01 and host02 is bridged over host03.

As with the previous assignment, this assignment uses *Linux Containers* (LXC). Again, we recommend you use `Ubuntu 20.04` as the host *Operating System* (OS) for this assignment. If you don't run this OS, you can choose to run a virtualized instance. To set up the environment, make the `setup_environment.sh` script executable and execute the script.

```
chmod +x setup_environment.sh
./setup_environment.sh
```

To resume the environment after a reboot of the host OS, use the `resume_environment.sh` script.

```
chmod +x resume_environment.sh
./resume_environment.sh
```

To enter a container in LXC, you should execute the following command. `HOST` is used in the command below as a placeholder for the name of the container.

```
sudo lxc-attach HOST bash
```

After finishing the assignment, you can destroy the environment using the command below. First, make sure to copy your developed files outside the container. Otherwise, your assignment will be lost.

```
chmod +x destroy_environment.sh
./destroy_environment.sh
```

Goal

Develop a Python3 script called `run_stp.py`. This script should accept the following arguments: the name of the first interface to use as a bridge, the name of the second interface to use, a bridge ID, and a bridge priority. The invocation of this script is shown below.

```
python3 run_stp.py eth1 eth2 CA:F2:5D:EB:D6:9C 1
```

The name of the bridge you create should be equal to `br0`.

After executing the attack, you should be able to intercept traffic between host01 and host02. To generate traffic between the two hosts, run the following command.

```
sudo lxc-attach host01 ping 192.168.12.3
```

Inside host03, you should be able to inspect the traffic between host01 and host02.

Getting started

Take a look at the following web page to get acquainted with bridges in Linux:

https://wiki.archlinux.org/title/Network_bridge

You receive bonus points if you develop the whole solution to this assignment in Python. This means that STP is fully running on Python code. However, it's also sufficient to make your script call underlying Linux commands in order to configure and activate STP.

Note that LXC doesn't work well when there are other container solutions present on the same Operating System (such as Docker).

Submission Instructions

Your script should run on the host01 container. After completing the assignment, you should only submit the resulting script. A `requirements.txt` file should be provided if you use any modules from the Python Package Index (pip).

Make sure your assignment conforms to the in- and output described in the Goal subsection. Your submission is partly graded through an automated system. Any deviations from the described in- and output can affect your grade.