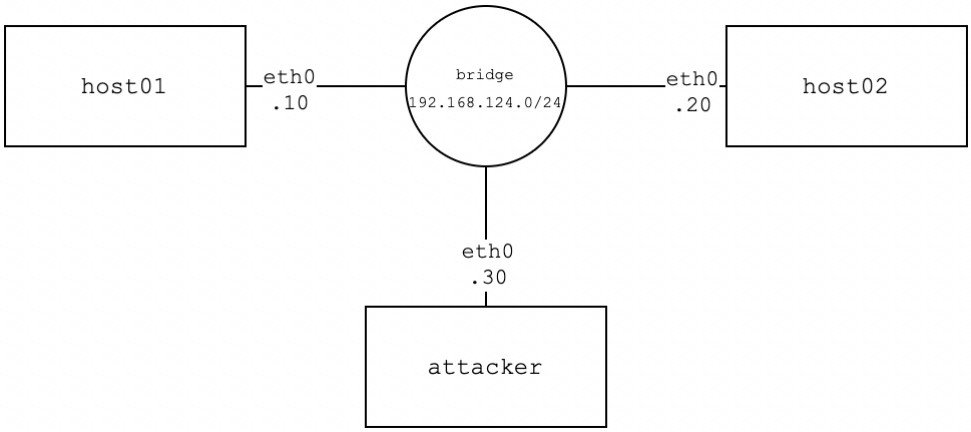
Network Security Q4 Y2024-25

Assignment: Data Link Layer

ARP spoofing [50 points]

In the setup depicted in the image below, two hosts are constantly communicating with each other. We know that the network isn't protected against ARP-related attacks.



In this assignment, you will exploit this lack of security. This will allow you to capture traffic originating from `host01` to `host02` and vice versa. Use the attacker container as the vantage point for executing your attack.

Before starting with the exercise, unzip the provided file and navigate to the resulting directory. Afterward, roll out the environment using the commands below.

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

The hosts above are configured as Docker containers. The *attacker* container has a mount configured, such that you can edit files directly on your computer in an editor of your choice and then access them within the container: files in the `solution` directory are mirrored in the container under `/solution`. You can enter the container by executing the following command.

```
docker exec -ti assignment_2_attacker bash
```

After finishing the assignment, make sure to clean up the environment using the command below.

```
docker compose down
```

Goal

Create a Python3 script that runs on the *attacker* container. The arguments should be the IP addresses of two hosts in the network. Spoof the ARP tables of the two hosts, such that you can intercept packets from the first host to the second host and vice versa. The script should run in an infinite loop, continuously spoofing the ARP tables of the two machines. The command below shows how this script should be executed.

```
python3 spoof_arp.py 192.168.124.10 192.168.124.20
```

You should continuously print the source and destination IP address taken from the captured packets as well as the raw content of the last layer. An example of the expected output is shown below.

```
root@attacker:/solution# python spoof_arp.py 192.168.124.10 192.168.124.20
Received traffic from 192.168.124.20 to 192.168.124.10: b'Hi!'
Received traffic from 192.168.124.10 to 192.168.124.20: b'Hi!'
Received traffic from 192.168.124.10 to 192.168.124.20: b'Hello'
Received traffic from 192.168.124.20 to 192.168.124.10: b'Hello'
Received traffic from 192.168.124.20 to 192.168.124.10: b'Hi!'
[...]
```

Finally, when the user presses Ctrl+C, the script should restore the ARP tables to their original state so that the two machines can communicate with each other normally again.

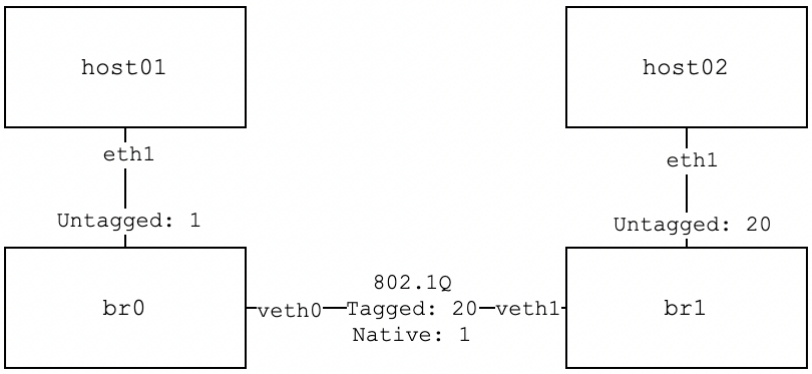
You may not use external libraries or tools different from those installed on the provided *attacker* container.

Submission Instructions

After completing the assignment and verifying that it works correctly, submit your Python script on WebLab.

VLAN hopping [50 points]

In this assignment, we work with the topology shown in the image below. The two hosts are in different VLANs — `host01` (`192.168.120.100`) is in VLAN 1 and `host02` (`192.168.130.100`) is in VLAN 20. Under normal conditions, this prevents them from communicating with each other. ID 1 is the native VLAN.



However, such a setup is vulnerable to a double tag attack, allowing `host01` to send packets to `host02`.

We recommend running this assignment on Linux (or in a VM). To set up the environment, extract files from the provided ZIP and execute the setup script.

```
sudo ./setup_environment.sh
```

The two hosts are configured as Docker containers. As before, the *attacker* container (`host01` in the diagram above) has a mount configured; files in the `solution` directory on your computer can be accessed in the container under `/solution`. You can enter the container by executing the command below.

```
docker exec -it assignment_2_vlan_attacker bash
```

You can use `tcpdump` to view packets received by the other host. This can be done as follows:

```
docker exec -it assignment_2_vlan_host2 tcpdump -vv -n
```

For debugging, you might also find it useful to attach Wireshark or use `tcpdump` on various interfaces in the network in order to observe packets. Some such interfaces are `vlan_veth0` (connecting `host01` to `br0`), `vlan_trunk0` (connecting `br0` to `br1`), and `vlan_veth1` (connecting `host02` to `br1`).

After finishing the assignment, make sure to clean up the environment using the command below.

```
sudo ./destroy_environment.sh
```

Goal

Make a Python3 script that runs on the *attacker* container. The program should accept three parameters: the VLAN ID of the attacker host, the VLAN ID of the target host, and the IP address of the target. Execute the attack and send an ICMP packet (echo request / ping) that bypasses the VLAN restrictions and is correctly received and accepted by the target host.

```
python3 double_tag.py <first VLAN ID> <second VLAN ID> <destination IP>
```

You can verify whether the double tag attack succeeded by inspecting incoming traffic on `host02`:

```
$ docker exec -it assignment_2_vlan_host2 tcpdump -vv -n
[...]
12:23:51.130009 IP (tos 0x0, ttl 64, id 1, offset 0, flags [none], proto ICMP (1),
length 28)
    192.168.120.100 > 192.168.130.100: ICMP echo request, id 0, seq 0, length 8
12:23:51.130031 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.130.1
tell 192.168.130.100, length 28
12:23:52.151512 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.130.1
tell 192.168.130.100, length 28
12:23:53.175393 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.130.1
tell 192.168.130.100, length 28
[...]
```

You may not use external libraries or tools different from those installed on the provided *attacker* container.

Submission Instructions

After completing the assignment and verifying that it works correctly, submit your Python script on WebLab.