Kathará

# Kathará lab

two computers

| | |
|---|---|
| **Version** | 1.0 |
| **Author(s)** | L. Ariemma |
| **E-mail** | contact@kathara.org |
| **Web** | http://www.kathara.org/ |
| **Description** | A lab with two directly connected device with the goal of teaching how to change a MAC address and to sniff a Kathará collision domain |

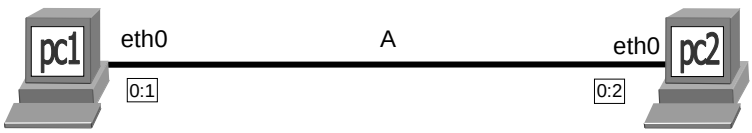




# Copyright notice

- All the pages/slides in this presentation, including but not limited to, images, photos, animations, videos, sounds, music, and text (hereby referred to as "material") are protected by copyright.
- This material, with the exception of some multimedia elements licensed by other organizations, is property of the authors and/or organizations appearing in the first slide.
- This material, or its parts, can be reproduced and used for didactical purposes within universities and schools, provided that this happens for non-profit purposes.
- Information contained in this material cannot be used within network design projects or other products of any kind.
- Any other use is prohibited, unless explicitly authorized by the authors on the basis of an explicit agreement.
- The authors assume no responsibility about this material and provide this material "as is", with no implicit or explicit warranty about the correctness and completeness of its contents, which may be subject to changes.
- This copyright notice must always be redistributed together with the material, or its portions.

# topology








# the lab.conf

```
lab.conf
pc1[0]=A
pc1[image]="kathara/base"
pc1[ipv6]="false"

pc2[0]=A
pc2[image]="kathara/base"
pc2[ipv6]="false"
```

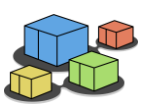
pc1 will use image "kathara/base"

ipv6 networking is disabled

# remarks

- the used image is "kathara/base"
  - contains basic networking tools
- IPv6 is used in specific labs
  - get used to add the flag so it won't create problems when IPv6 is used





# question

two computers on the same LAN; can communicate?

# question

how?



# ethernet frames

- ethernet frames (Layer 2 frames) can be sent on a LAN just knowing the destination MAC address
- Linux has no default utilities for sending such packets
- we'll use the python library `scapy`

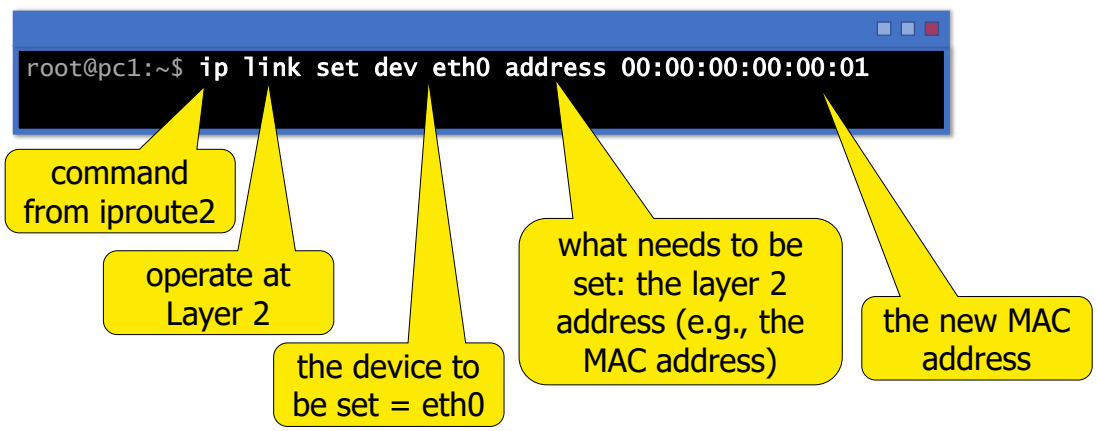
# MAC addresses and Kathará

- Kathará assigns random MAC addresses to network interfaces
  - the obtained MAC addresses are difficult to remember
  - it is possible to change the default MAC addresses with something more readable




# change the default MAC address

- the MAC address can be changed with command



# a small reminder

- the commands executed inside a console can also be inserted into the .startup file of the corresponding machine

**pc1.startup**

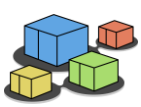
```
ip link set dev eth0 address 00:00:00:00:00:01
```

**pc2.startup**

```
ip link set dev eth0 address 00:00:00:00:00:02
```







# the lab – a second version

**lab.conf**

```
pc1[0]=A
pc1[image]="kathara/base"
pc1[ipv6]="false"

pc2[0]=A
pc2[image]="kathara/base"
pc2[ipv6]="false"
```

**pc1.startup**

```
ip link set dev eth0 address 00:00:00:00:00:01
```

**pc2.startup**

```
ip link set dev eth0 address 00:00:00:00:00:02
```

# let's start the lab



# scapy

- a versatile Python tool for crafting, analyzing, and sniffing network packets; it supports numerous protocols and offers an interactive shell for real-time network manipulation
- comes with an interactive shell
- allows you to craft custom packets with any field values

# entering the scapy shell

- to enter the scapy shell use command

```
root@pc1:~$ scapy
                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.5.0
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | Wanna support scapy? Star us on
       scccccp///pSP///p          p//Y   | GitHub!
      sY/////////y  caa           S//P   |             -- Satoshi Nakamoto
       cayCyayP//Ya              pY/Ya   |
        sY/PsY////YCc           aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs

>>>
```



# exiting the scapy shell

- to quit the scapy shell use command

```
>>> exit()
```

# crafting an ethernet frame

- to craft a layer 2 frame, use command

```
>>> p=Ether(dst='00:00:00:00:00:0B', src='00:00:00:00:00:0A')
```

save this packet into a variable p

create a new instance of the Ether class (used to craft Ethernet frames)

destination MAC address

source MAC address



# sending an ethernet frame

- to send a layer 2 frame, use command

```
>>> sendp(p, iface='eth0')
```

send a packet

the name of the variable holding the packet

the network interface through which the packet will be sent

# BEWARE

- this commands are typed into the `scapy` interactive shell
- those are NOT system commands; this means that can NOT be written into a .startup file
- moreover, it doesn't make sense to automate the sending



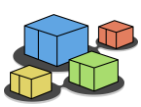
# what after sending?

- the frame has been sent, but a few questions arise:
  - how do we know if it was received?
  - how do we know if the data we set (src and dest MAC addresses) were correctly written?
  - how do we know if Kathará is not a complete scam?

# a packet sniffer

- A packet sniffer, a.k.a. network analyzer, is a tool that captures ("sniffs") network traffic as it transits over a network segment
- It allows users to see the content of packets being transmitted or received over a network
- Packet sniffers are used for network diagnostics, performance analysis, and cybersecurity purposes to detect vulnerabilities or malicious activities





# wireshark

- A widely-used, open-source packet analyzer
- It captures and displays the data traveling into and out of network devices in real-time
- With its powerful filtering and analysis tools, Wireshark is invaluable for network troubleshooting, protocol development, and cybersecurity investigations

# using wireshark in Kathará

- is it very easy to use Wireshark inside a Kathará lab
- there is an official guide available here
  - https://github.com/KatharaFramework/Kathara-Labs/tree/main/tutorials/capture-packets
- we need to add a dedicated device with the sole goal of running wireshark



# add the wireshark device to a lab

- to add a device (called wireshark) running Wireshark to a Kathará lab, add the following lines to the lab.conf file

lab.conf

```
....


wireshark[bridged]=true
wireshark[port]="3000:3000"
wireshark[image]="lscr.io/linuxserver/wireshark"
wireshark[num_terms]=0
```

connect a device called wireshark to the host network

map *port* 3000 of the host to *port* 3000 of the guest

use a dedicated image

do not open any terminals

# restart the lab

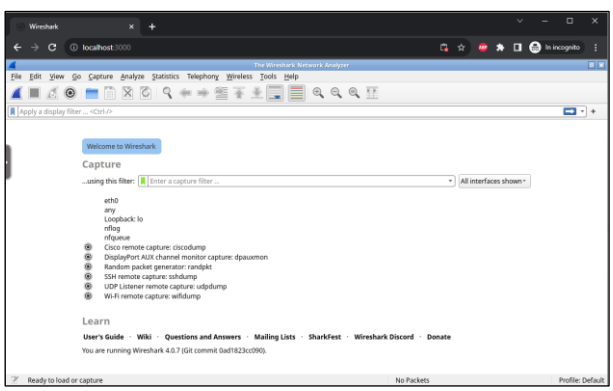
- after the lab.conf is changed, the lab needs to be restarted in order to apply the new configuration

```
user@localhost:kathara-lab_two-computers$ kathara lrestart
```

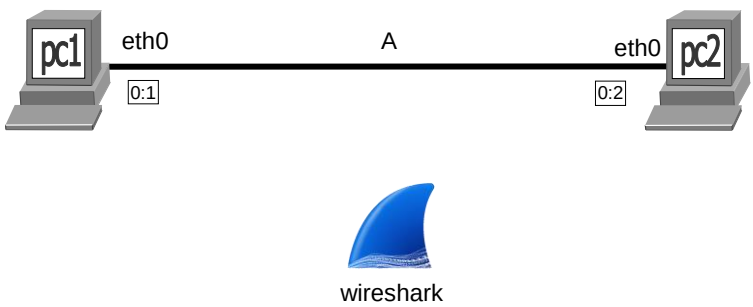


# open Wireshark

- open a browser in your host and go to
  - http://localhost:3000



# topology with Wireshark








# connect wireshark to the CD

- to connect the wireshark device to the collision domain to be sniffed, use this command

```
user@localhost:kathara-lab_two-computers$ kathara lconfig -n wireshark --add A
```

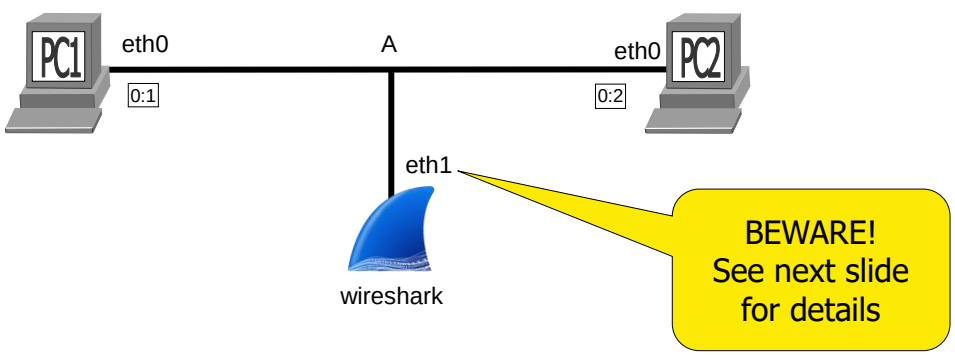
manage the network interfaces of a running lab

specify the name (n) of the device to modify

add a network interface ....

.... connected to collision domain A

# topology with Wireshark – 2nd version





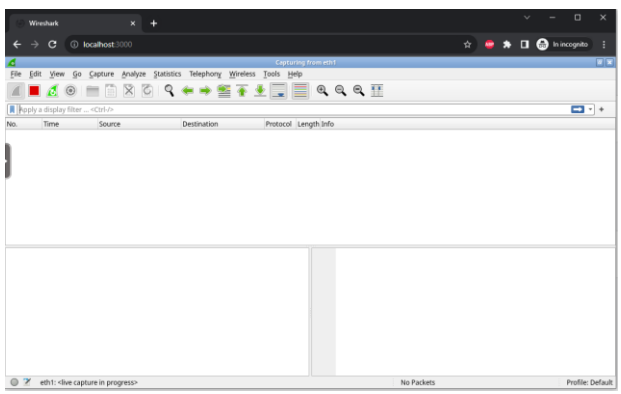



# notice about the wireshark device

- the wireshark device is created connected to the host
  - the eth0 interface is connected to the host
- when it is connected to a collision domain to be sniffed, an interface is added
  - the interface connected to the collision domain will be eth1

# sniffing the first packet – part 1

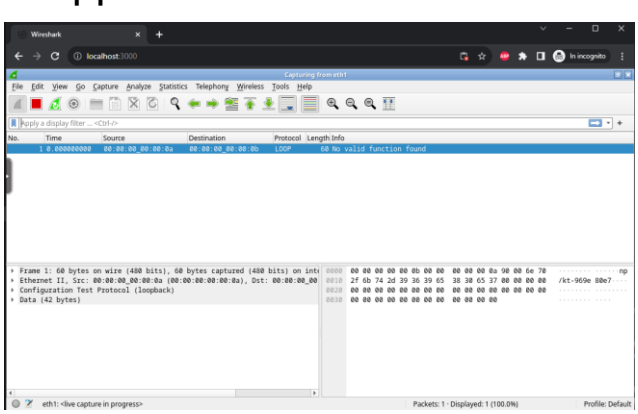
- reload the wireshark web page
- open the eth1 interface by double clicking on it





# sniffing the first packet – part 2

- send a new packet with `scapy`
- a wild packet appeared



# sniffing the packet – part 3

- use the wireshark functions to see the packet contents

```
▸ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)…
▾ Ethernet II, Src: 00:00:00_00:00:0a (00:00:00:00:00:0a), Dst: 00:0…
  ▸ Destination: 00:00:00_00:00:0b (00:00:00:00:00:0b)
  ▸ Source: 00:00:00_00:00:0a (00:00:00:00:00:0a)
    Type: Loopback (0x9000)
▸ Configuration Test Protocol (loopback)
▸ Data (42 bytes)
```



# sniffing the packet – part 4

- give a look also at the hex dump of the packet
- why it ends with a lot of zeros?

```
0000  00 00 00 00 00 0b 00 00  00 00 00 0a 90 00 6e 70
0010  2f 6b 74 2d 39 36 39 65  38 30 65 37 00 00 00 00
0020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0030  00 00 00 00 00 00 00 00  00 00 00 00
```

# exercises

- try to send different ethernet frames and sniff them
  - try to send a frame from pc2 to pc1
  - try to send a frame to a MAC address that's not on the LAN
  - try to add other devices to the LAN