

# Lab 1 - Session 2

---

Daniel Perdices

May 13, 2024



# What are we going to do today?

Recap of Wireshark

Recap of Python

Advanced Python I

Self-assessment

Material

# Recap of Wireshark

---

# How to start Wireshark?

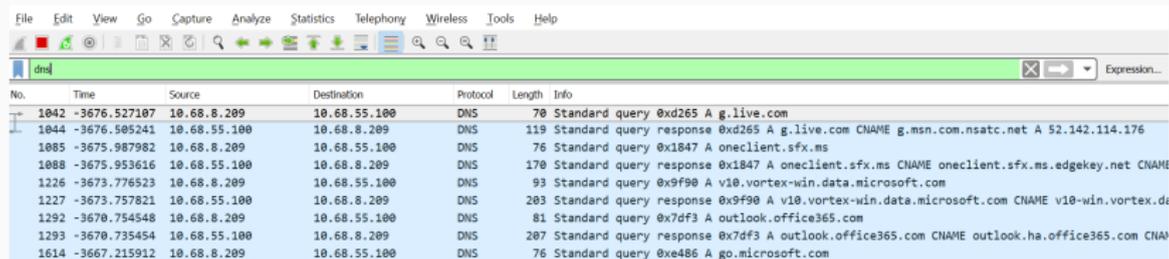
Run in a terminal

```
$ sudo wireshark # or wireshark-gtk
```

- Permission error when starting live capture if you open it without sudo.
- Remember: <TAB> is your BFF in bash.

# Visualization filter

The filters on the top part of the screen



The screenshot shows the Wireshark interface with a green visualization filter bar at the top of the packet list. The filter bar contains the text "[dns]". Below the filter bar, the packet list is displayed with columns for No., Time, Source, Destination, Protocol, Length, and Info. The packets shown are all DNS queries and responses.

No.	Time	Source	Destination	Protocol	Length	Info
1042	-3676.527107	10.68.8.209	10.68.55.100	DNS	70	Standard query 0xd265 A g.live.com
1044	-3676.505241	10.68.55.100	10.68.8.209	DNS	119	Standard query response 0xd265 A g.live.com CNAME g.msn.com nsatc.net A 52.142.114.176
1085	-3675.987982	10.68.8.209	10.68.55.100	DNS	76	Standard query 0x1847 A oneclient.sfx.ms
1088	-3675.953616	10.68.55.100	10.68.8.209	DNS	170	Standard query response 0x1847 A oneclient.sfx.ms CNAME oneclient.sfx.ms.edgekey.net CNAME
1226	-3673.776523	10.68.8.209	10.68.55.100	DNS	93	Standard query 0x9f90 A v10.vortex-win.data.microsoft.com
1227	-3673.757821	10.68.55.100	10.68.8.209	DNS	203	Standard query response 0x9f90 A v10.vortex-win.data.microsoft.com CNAME v10-win.vortex.d
1292	-3670.754548	10.68.8.209	10.68.55.100	DNS	81	Standard query 0x7df3 A outlook.office365.com
1293	-3670.735454	10.68.55.100	10.68.8.209	DNS	207	Standard query response 0x7df3 A outlook.office365.com CNAME outlook.ha.office365.com CNAME
1614	-3667.215912	10.68.8.209	10.68.55.100	DNS	76	Standard query 0xe486 A go.microsoft.com

**Figure 1:** Visualization filter (green bar on top of packet list)

## Characteristics

- Flexible
- They can be undone
- Only inside Wireshark (or tools from same devs)
- Slow

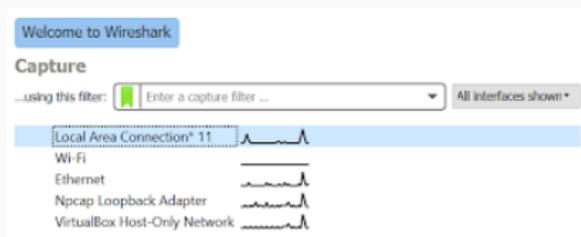
# Visualization filter

## Examples:

- All ARP traffic
- All packets with TCP source port  $< 10000$
- First ten packets
- All traffic going to or coming from address 10.0.0.1

# Capture filter

The filters when you configure a network interface



**Figure 2:** Capture filter

## Characteristics

- Less flexible.
- You cannot “uncapture” what it has not been captured.
- Format is widely accepted. See BPF (Berkeley Packet Filter)
- Fast, they compile to assembly code running in the kernel

## Examples:

- All ARP traffic
- All packets with TCP source port = 22
- TCP and UDP traffic
- All traffic going to or coming from address 10.0.0.1

# Recap of Python

---

## Basic syntax

```
def gradeInSpanish(grade, distinction=False,
                   recommendation=False):
    if grade < 5:
        anot_grade = "Suspenso"
    elif grade < 7:
        anot_grade = "Aprobado"
    elif grade < 9:
        anot_grade = "Notable"
    elif distinction and recommendation:
        anot_grade = "M. Honor"
    else:
        anot_grade = "Sobresaliente"
    return anot_grade
```

```
for i in range(10+1):  
    print(gradeInSpanish(i))
```

- What's the first value of *i*? And the last one?
- Transform this code to save the result into a list instead of printing it.

# Basic types

```
a = 1
a1 = 0x01
a2 = 0b01
b = 1.0
c = "1"
d = True
e = None
f = 1+2j # Sorry, I know you already
        # passed analog circuits... :)
```

# Data structures

- Remember: types are not enforced

```
a = [1,2,"3"] # list
```

```
b = {1,2,"3",3} # set (uniqueness)
```

```
c = (1,2,"3") # tuple (unmodifiable)
```

- Dictionaries are extremely useful:

```
d = {1:"foo", 2: "bar"}
```

- What happens if you iterate a, b or c? And d?
- How do you sort an Iterable?

# Advanced Python I

---

## How to convert from integer to bytes?

First option (the one I personally don't like)

```
ethertype = struct.pack("!H", value)
```

Problems:

- Harder to read
- Requires “!” to indicate network order (big endian)
- Requires “B”, “H”, “I”, or “Q” for the size (1, 2, 4, 8 bytes respectively)

Advantage:

- It allows you to do multiple conversions at once:

```
random_bytes = struct.pack("!BH", value1, value2)
```

## How to convert from integer to bytes?

Second option (the one I personally like)

```
# parenthesis are required if it is a literal  
# they are not if it's a variable  
ethertype = (value).to_bytes(2, # size of the number  
                             order="big") #big=network
```

Advantages:

- Easier to read
- Size and order are explicitly declared and human-readable

Problem:

- It does not allow you to do multiple conversions at once:

```
random_bytes = value1.to_bytes(1) + \  
                value2.to_bytes(2, order="big")
```

## How to convert from bytes to integer?

First option (the one I personally don't like)

```
value = struct.unpack("!H", ethertype)
```

Problems:

- Harder to read
- Requires “!” to indicate network order (big endian)
- Requires “B”, “H”, “I”, or “Q” for the size (1, 2, 4, 8 bytes respectively)

Advantage:

- It allows you to do multiple conversions at once:

```
(value1,value2) = struct.unpack("!BH", random_bytes)
```

## How to convert from bytes to integer?

Second option (the one I personally like)

```
value = int.from_bytes(ethertype, #size is len(ethertype)  
                       order="big") #big=network
```

Advantages:

- Easier to read
- Size and order are explicitly declared and human-readable

Problem:

- It does not allow you to do multiple conversions at once:

```
value1 = int.from_bytes(random_bytes[0:1], order="big")  
value2 = int.from_bytes(random_bytes[1:], order="big")
```

## Test: do you really know python?

```
def broadcastEthAddress():  
    # TODO  
    return bytes()
```

## Test: do you really know python?

```
def int2hex():  
    # TODO  
    return ""  
  
def bytes2hex():  
    # TODO  
    return ""
```

# Self-assessment

---

# What you should have done by today

- Basic Wireshark
  - Basic concepts
  - Viz filter vs capture filter
  - Open interfaces, open/save traces
- Basic Python
  - Understand the examples
  - Understand our material

# What you should have done by today

- Linux (either baremetal install or VM)
  - `apt update && apt install libpcap-dev`
- Read Assignment 1
  - What are you being asked for?
  - What is the material provided?
  - How are you going to be evaluated?
  - How do you know that what you are doing is right?
- Run program `practica1.py`

# What you should have done by next week

- Questions about Wireshark
  - Lab 1 test might include questions regarding Wireshark
- Questions about Python
  - Lab 1 test might include questions regarding Python3
- Questions about the assignment
  - Lab 1 test will include questions regarding the assignment and the libpcap
- A first draft of the assignment

# Material

---

## Where are these slides?

`github.com/dperdices/redes1-1391-2022`

- Source code of this slides (in markdown)
- Slides in PDF

If you want a completed version or find any mistakes,

- Fork the repo
- Complete it / Fix it yourself
- Make a PR
- Wait for my approval (or comments)