# Network Traffic Analysis Via Wireshark

**Enrico RUSSO**
Università di Genova



*https://cybersecnatlab.it*

# License & Disclaimer



## License Information

This presentation is licensed under

the Creative Commons BY-NC License


To view a copy of the license, visit:

http://creativecommons.org/licenses/by-nc/3.0/legalcode

## Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

# Goal



- Learn how to perform network traffic analysis via Wireshark

# Prerequisites



- Lecture:
  - *NS_0.1 – Network Fundamentals*

# Outline



- Wireshark: GUI elements
- Wireshark: working with packets
- Wireshark: follow streams and save artifacts
- Wireshark: packet details pane, dissectors
- Using Wireshark in the command line: Tshark

# Outline



- Wireshark: GUI elements
- Wireshark: working with packets
- Wireshark: follow streams and save artifacts
- Wireshark: packet details pane, dissectors
- Using Wireshark in the command line: Tshark

# Wireshark



- Wireshark is a tool to capture data from a network (sniffer) and to analyse them
  - Analysis can be performed in real-time or on previously-recorded traffic files, through, e.g., *packet* capture or PCAP
  - Packets represent *generic* chunks of data and, depending on the considered level, can be interpreted as frames, datagram, or segment
- Available for UNIX and Windows: https://www.wireshark.org/

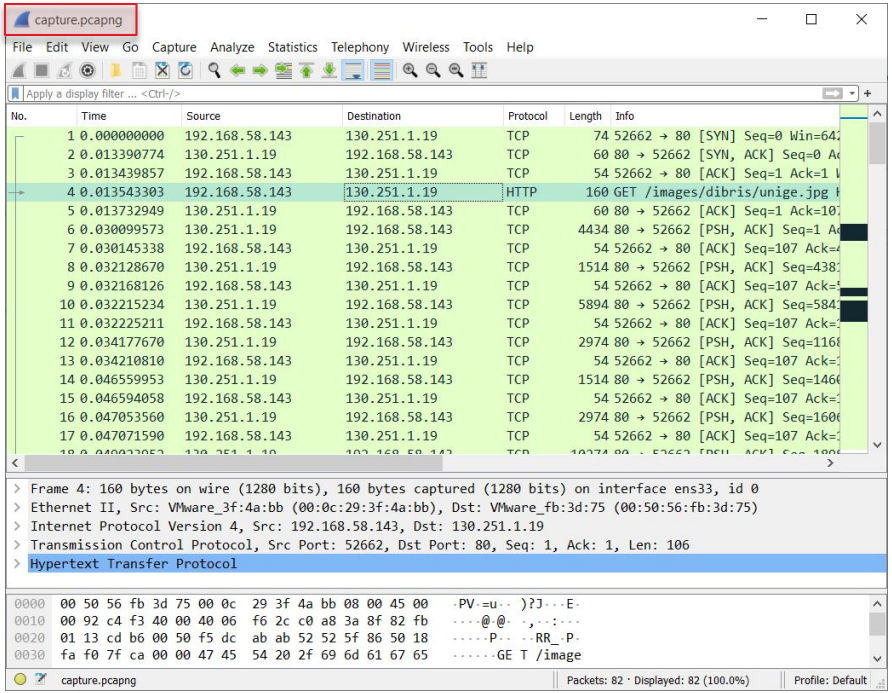
# Networking CTF challenges



- In some CTF challenges, we are given a PCAP file
- Typically, solving these challenges requires analyzing the capture to find the flag by
  - answering questions related to network traffic
  - carving file from packet streams
- Wireshark is a useful tool for these types of challenges

# Wireshark GUI

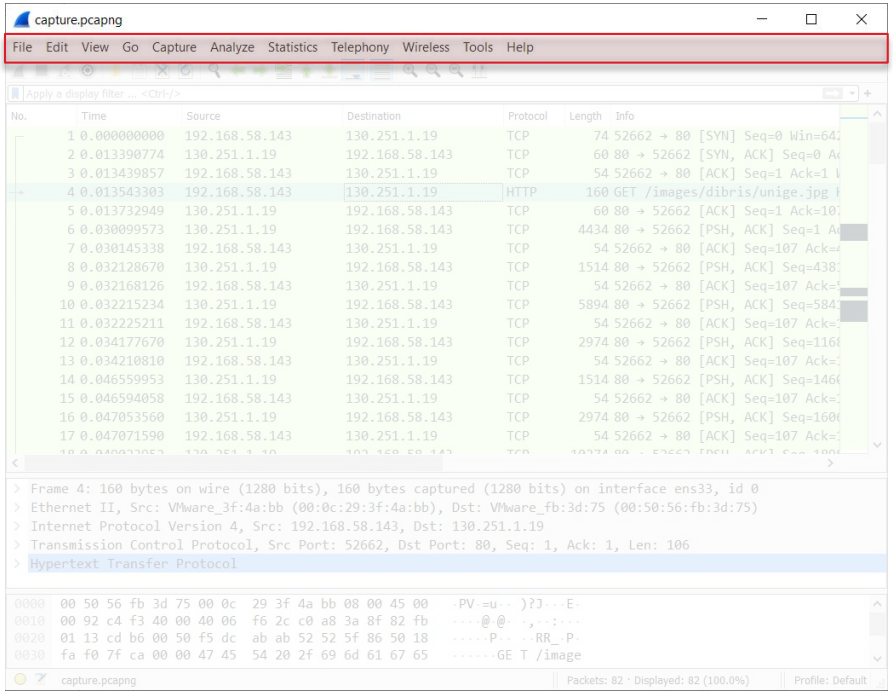





- Wireshark provides a Graphical User Interface (GUI)
- We detail its main elements as it appears after opening an existing PCAP file
  - From the File menu of the Start screen, use the command Open (CTRL-o) and select the PCAP file (e.g., capture.pcapng) to analyze

# Wireshark GUI: menu

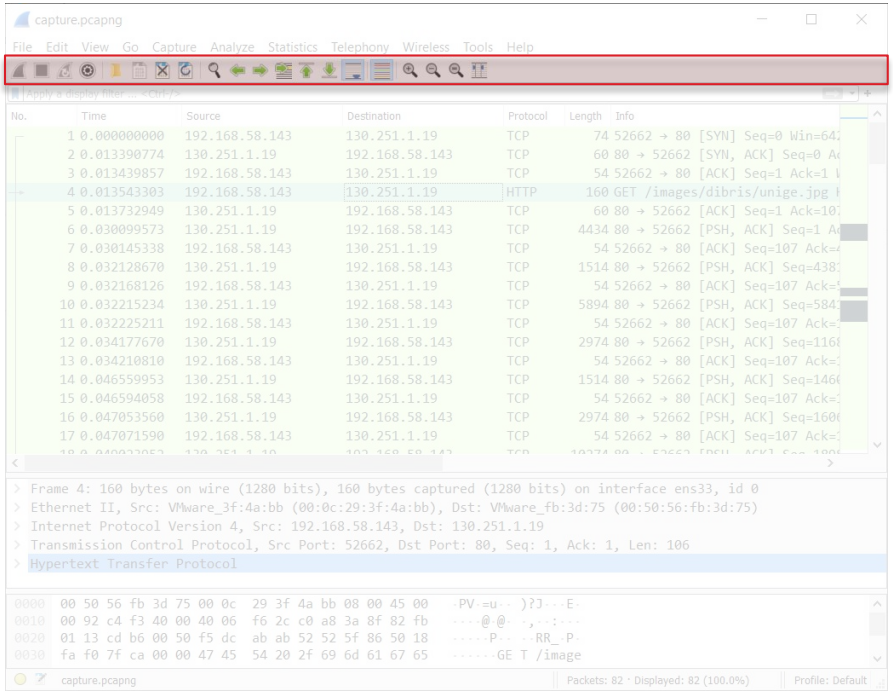





- The **menu** is used to start actions
- Of interest to us are
  - *File*: open and merge capture files, save, print, or export capture
  - *Edit*: find a packet, time reference or mark packets, handle configuration profiles
  - *View*: controls the display of packets (e.g., colorization, name resolution, or fonts)
  - *Go*: items to go to a specific packet
  - *Analyze*: manipulate display filters, enable or disable the dissection of protocols, follow a stream (see next)
  - *Statistics*: display various statistic windows, including a summary of the packets that have been captured, or display protocol hierarchy statistic.

# Wireshark GUI: main toolbar

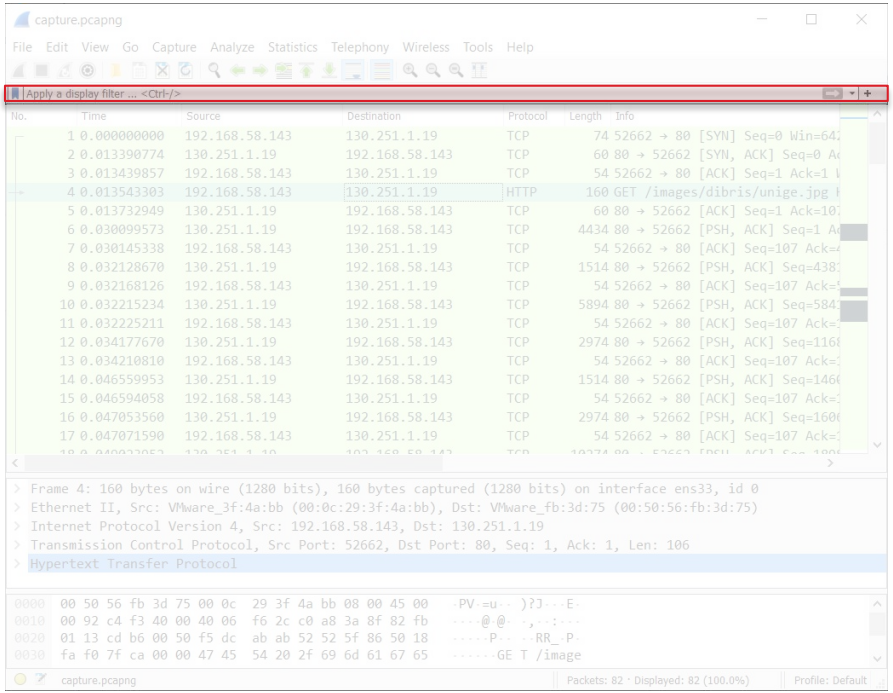





- The **main toolbar** provides quick access to frequently used items from the menu
- Items in the toolbar will be enabled or disabled (greyed out) similar to their corresponding menu items

# Wireshark GUI: filter toolbar

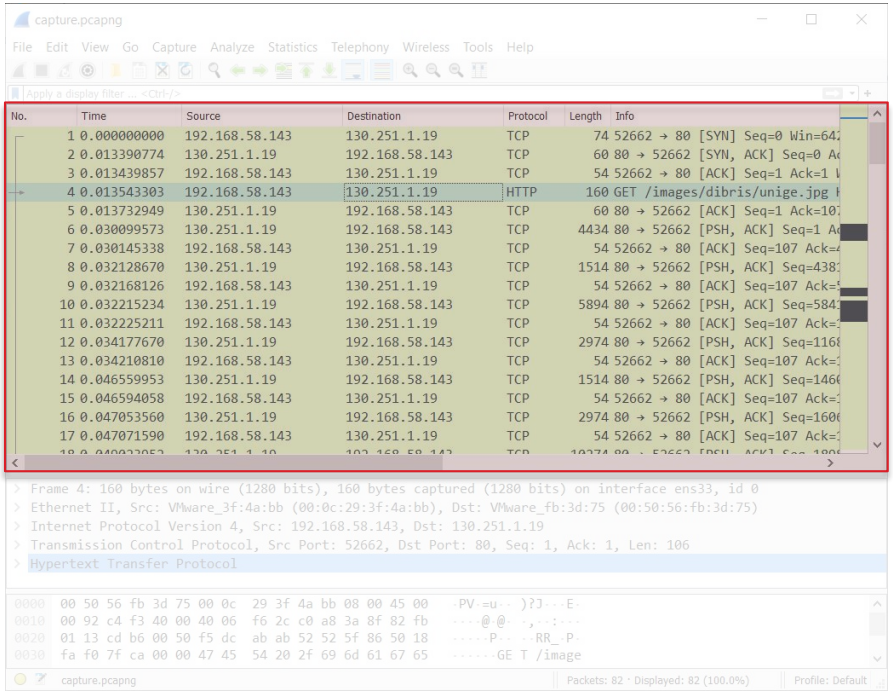





- The **filter toolbar** lets you quickly edit and apply display filters
  - Manage or select saved filters
  - Reset the current display filter and clear the edit area
  - Apply the current value in the edit area as the new display filter
  - Select from a list of recently applied filters
  - Add a new filter button (shortcuts that apply a display filter)

# Wireshark GUI: packet list

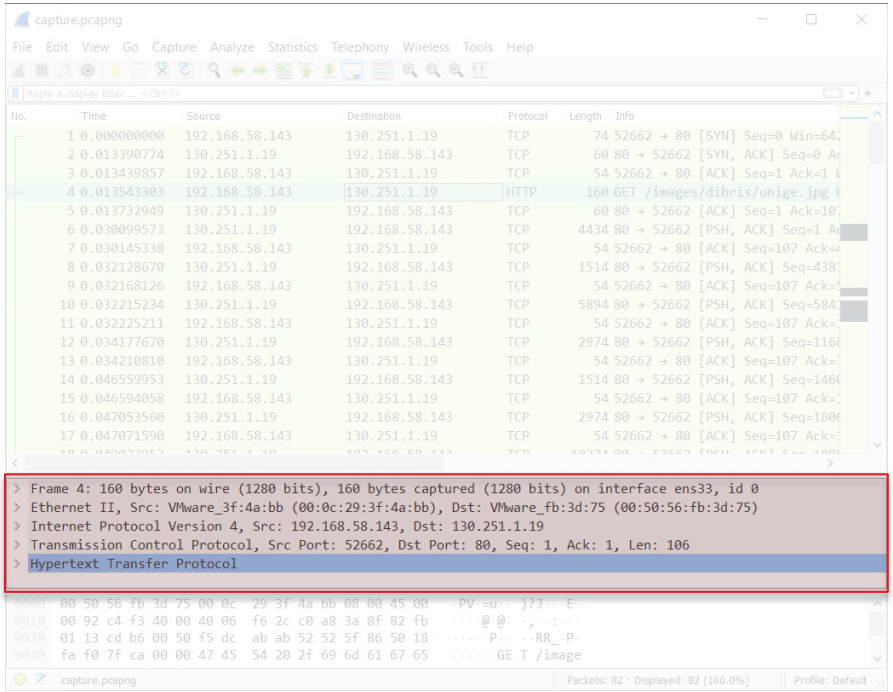





- The **packet list** pane displays a summary of each captured packet
- Each line in the packet list corresponds to one packet in the capture file (selecting a line in this pane displays more details in the *packet details* and *packet bytes* panes)
- Columns provide an overview of the packet
- You can click the column headings to sort by that value

# Wireshark GUI: packet details

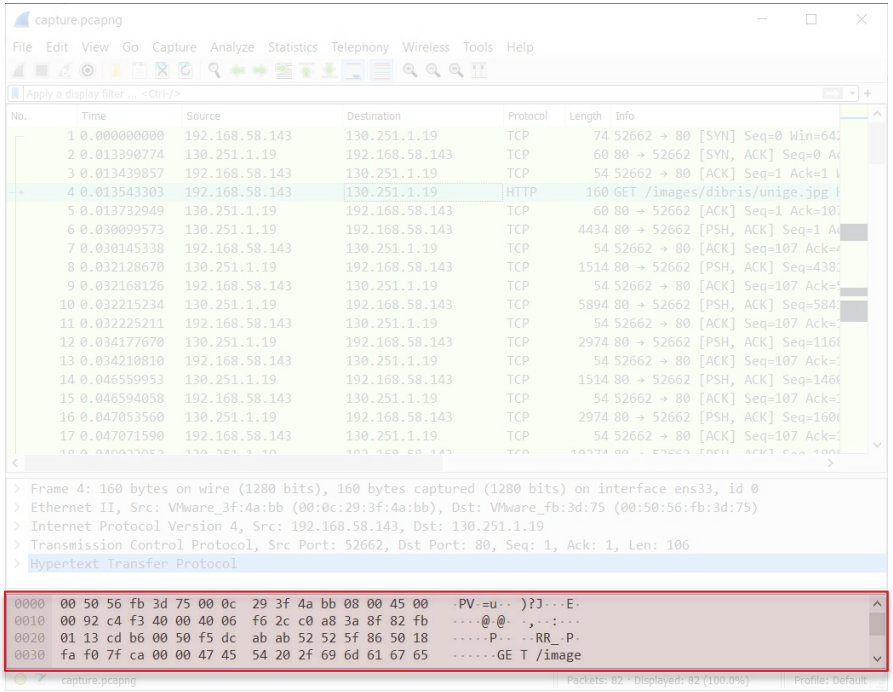





- The **packet details** pane shows the current packet (selected in the packet list pane) in a more detailed form
- In particular, it shows the protocols and fields of the packet in a tree, which can be expanded and collapsed

# Wireshark GUI: packet bytes

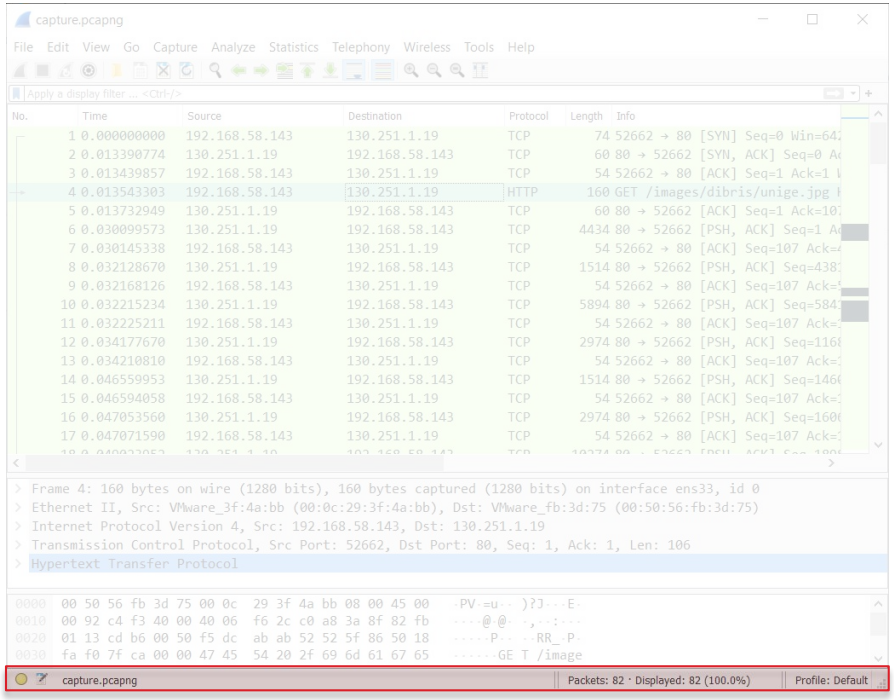





- The **packet bytes** pane shows the data of the current packet (selected in the packet list pane) in a hexdump style
- Each line contains
  - the data offset
  - sixteen hexadecimal bytes
  - sixteen ASCII bytes (Non-printable bytes are replaced with a period “.”)

# Wireshark GUI: statusbar






- The **statusbar** displays informational messages
  - The colorized bullet open the Expert Information dialog (list of anomalies and other items of interest found in a capture file)
  - The edit icon lets you add a comment to the capture file
  - The left side shows the file name or protocols fields information
  - The middle side shows the current number of packets in the file
  - The right side show the current profile

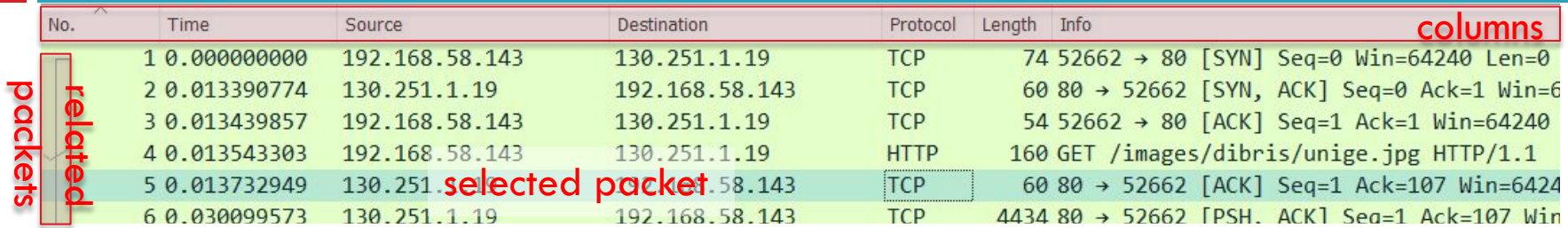
# Outline



- Wireshark: GUI elements
- Wireshark: working with packets
- Wireshark: follow streams and save artifacts
- Wireshark: packet details pane, dissectors
- Using Wireshark in the command line: Tshark

# The packet list pane

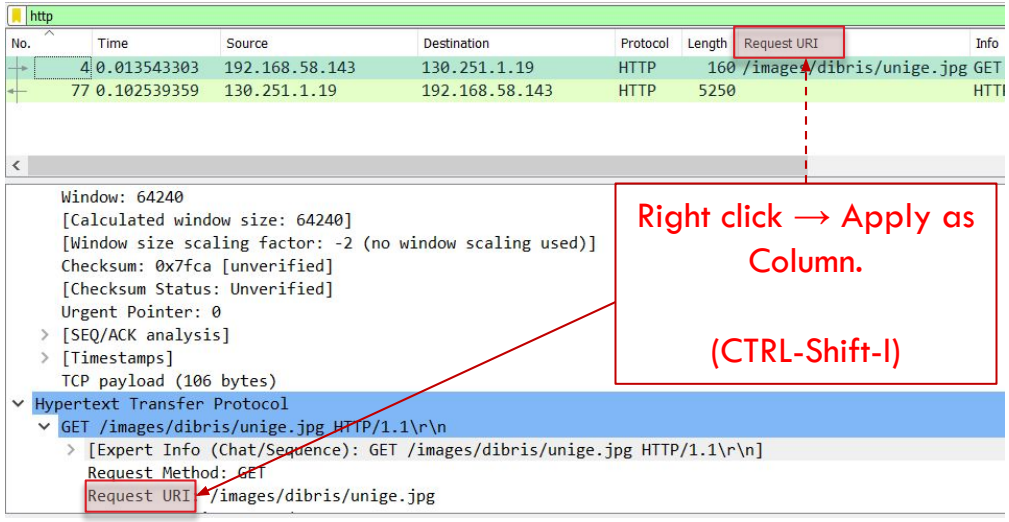





1. **No.** The number of the packet in the capture file. This number won't change, even if a display filter is used
2. **Time** The timestamp of the packet (change display format with *View → Time Display Format*)
3. **Source** The address where this packet is coming from
4. **Destination** The address where this packet is going to
5. **Protocol** The protocol name
6. **Length** The length of each packet
7. **Info** Additional information about the packet content

# Adding columns (example)



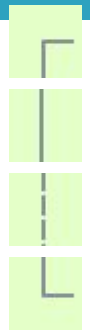



- Add a new column showing the request URI of HTTP packets
  - Select an HTTP packet
  - Expand the HTTP protocols fields in the packet details pane
  - Right click on the Request URI field and select Apply as Column

# Related packets symbols



- First packet in a conversation
- Part of the selected conversation
- Not part of the selected conversation
- Last packet in a conversation

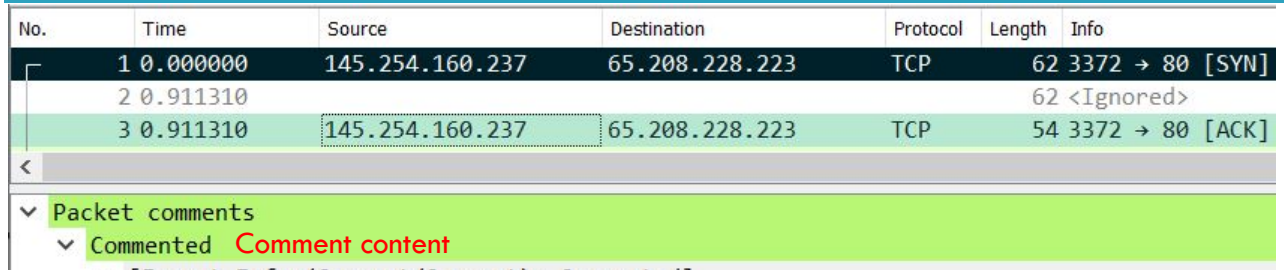



- Request
- Response

- The selected packet acknowledges this packet
- The selected packet is a duplicate acknowledgement of this packet
- The selected packet is related to this packet in some other way (e.g., as part of reassembly)

# Mark, ignore and comment



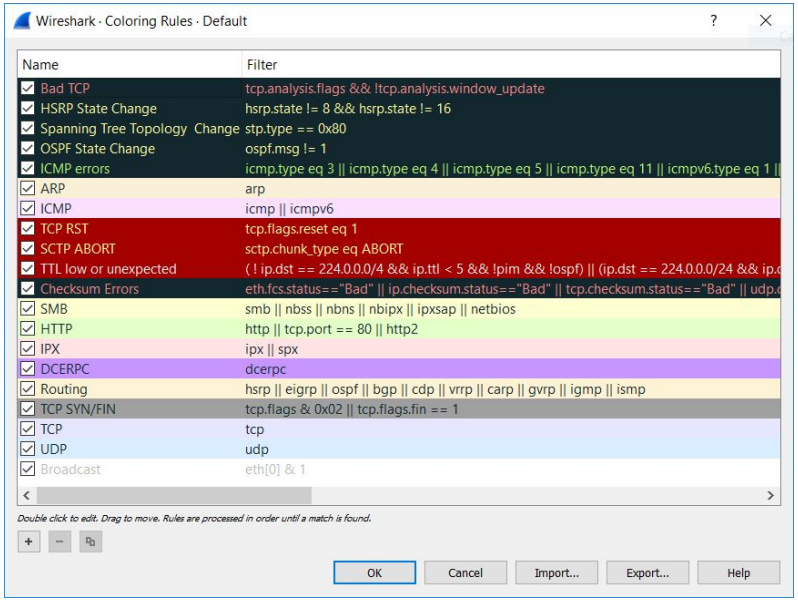



- mark packets of particular interest:
  - CTRL-M
  - jump forward and backward between marked packets: press SHIFT-CTRL-N and SHIFT-CTRL-B respectively
- ignore packets:
  - CTRL-D
- comment packets:
  - CTRL-ALT-C

# Coloring rules



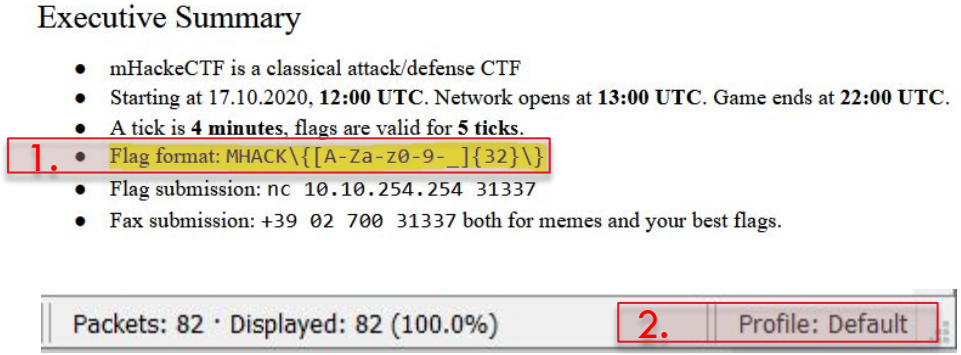
- Wireshark supports coloring rules for packets
- View → Coloring Rules...



# Coloring rules: A/D CTF example

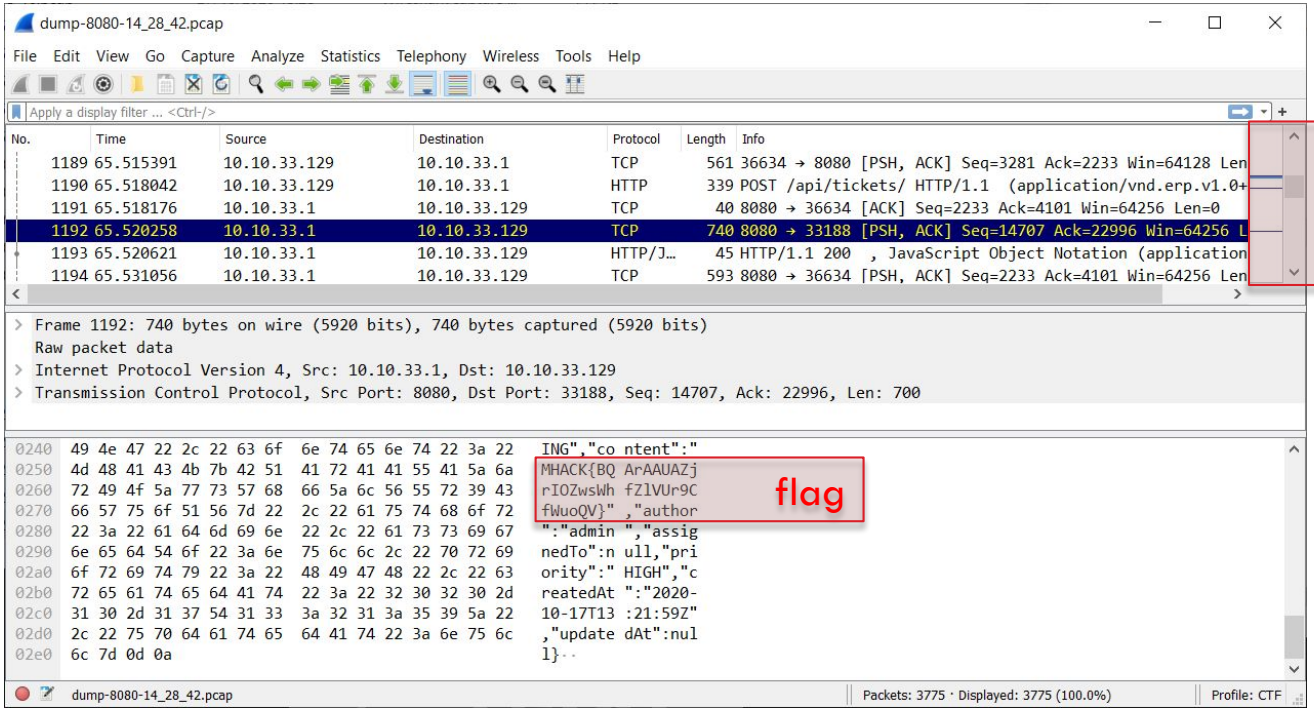


1. Get the flag format from CTF rules
2. Right click on the Profile label of the Status Bar → New (e.g, CTF)
3. View → Coloring Rules… and disable all existing rules.
4. Add a new rule for highlighting flags



# Coloring rules: A/D CTF example





# Display filters: filtering packets



- Wireshark provides a display filter language that enables you to precisely control which packets are displayed
- They can be used to check for
  - the presence of a protocol or field
  - the value of a field
  - compare two fields to each other
- These comparisons can be combined with logical operators and parentheses into complex expressions

# Building filter expressions



1. Help → Manual Pages → Wireshark Filters
2. Expression builder: right click on the toolbar → Display Filter Expression...
3. Select a protocols field in the packet details and use context menu entries:
   - Apply as Filter: filter the packet list with the selected key/value as the filter expression
   - Prepare a Filter: use the selected field key/value in the filter expression (filtering is not applied)

# Outline

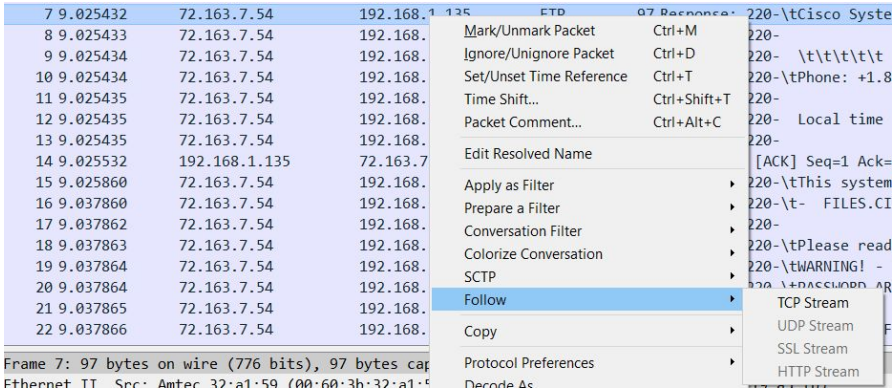


- Wireshark: GUI elements
- Wireshark: working with packets
- Wireshark: follow streams and save artifacts
- Wireshark: packet details pane, dissectors
- Using Wireshark in the command line: Tshark

# Follow streams



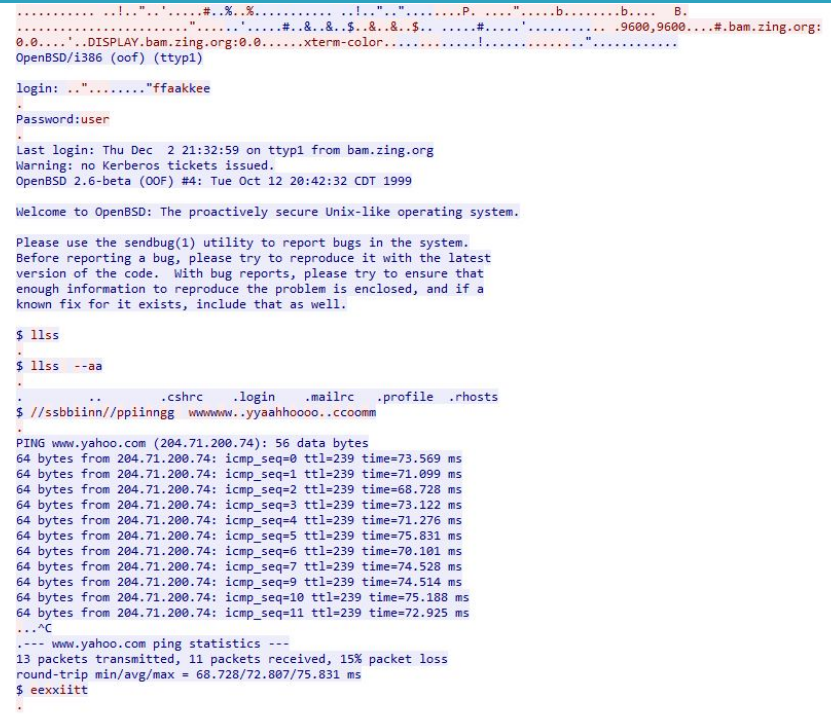
- **Follow stream** provides a different view on network traffic
- Instead of individual packets, one can see data flowing between client and server
- It can be enabled using the context menu in the packet list: a *display filter* which selects all the packets in the current stream is applied



# Follow streams: example



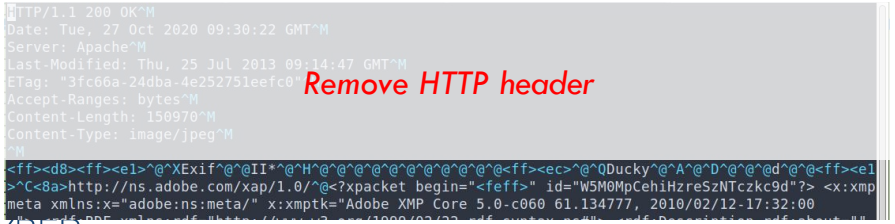
- Telnet is a type of client-server protocol that can be used to open a command line on a remote host
- *Blue* is the data from the server to the client (e.g., the *login:* prompt)
- *Red* is the data from the client to the server (e.g., the user password is sent by the client and is not echoed by the server)
- Non-printable characters are replaced by dots.

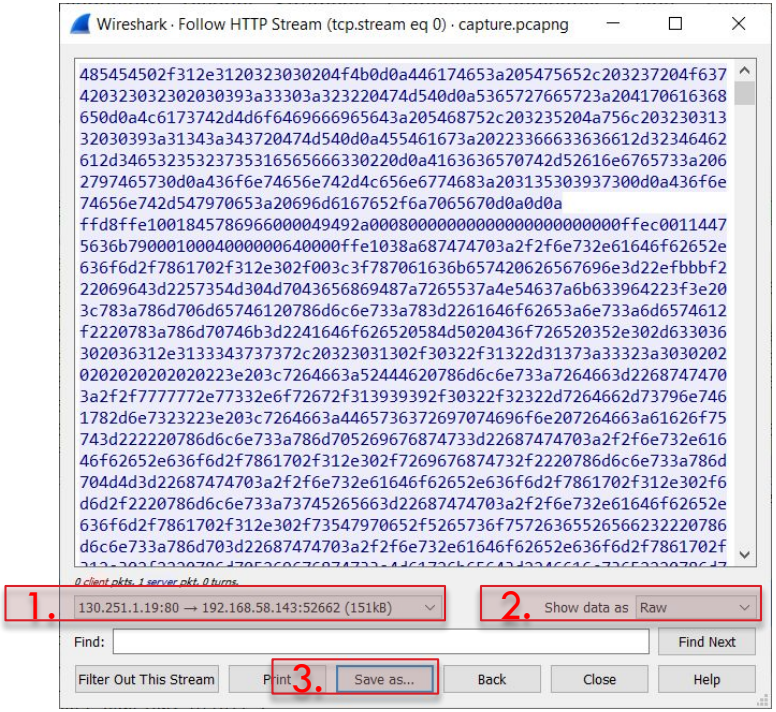


# Carve files from streams: example 1



- Extract and save a JPEG file downloaded using HTTP
  1. Select *server → client* packets (Blue)
  2. Show data as Raw
  3. Save as...
  4. Remove the HTTP header from the saved file



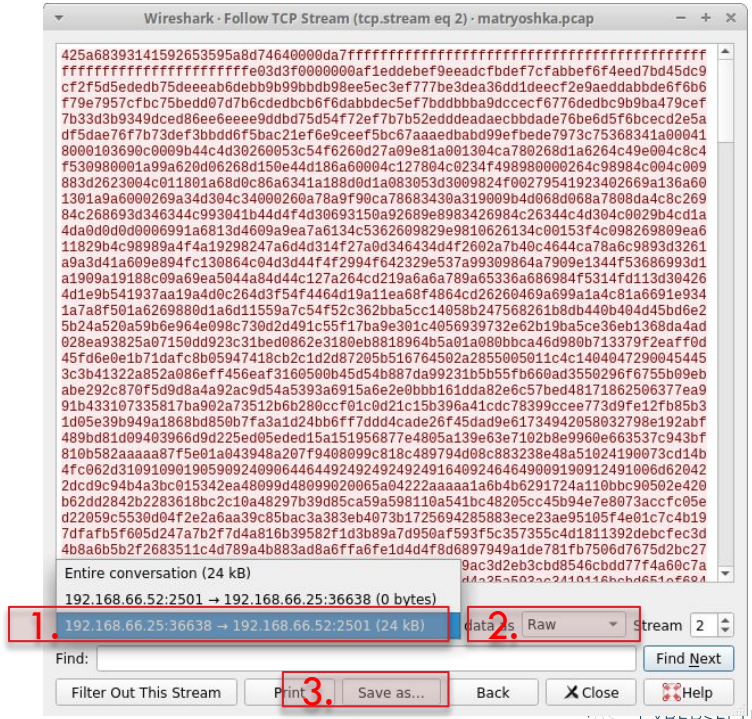



# Carve files from streams: example 2



- Extract a file from a generic stream (unknown protocol)
  1. Switch between client to server or server to client conversation
  2. Show data as Raw
  3. Save as... (e.g., */tmp/bin1*)
  4. Use the linux *file* utility to determine the file type

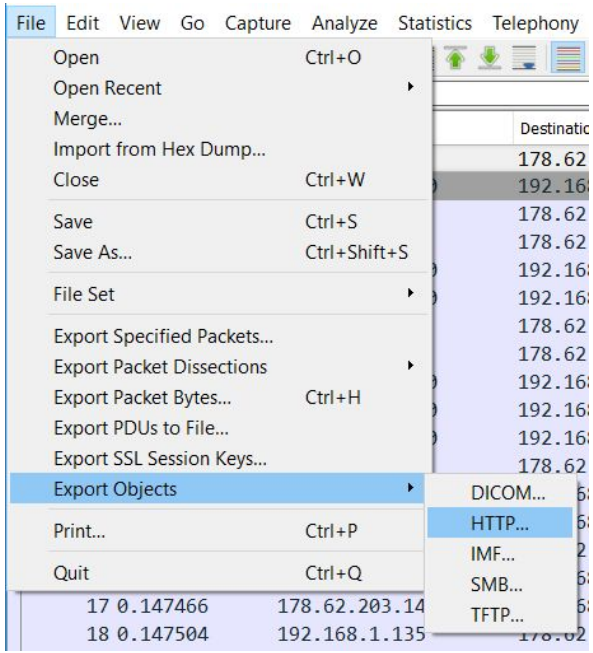
```
/tmp  file bin1
bin1: bzip2 compressed data, block size = 900k
```



# Export objects



- File → Export Objects.
- This feature scans through (some) protocol streams and takes reassembled objects (e.g., HTML docs, images, executables)
- They can be saved to disk



# Outline

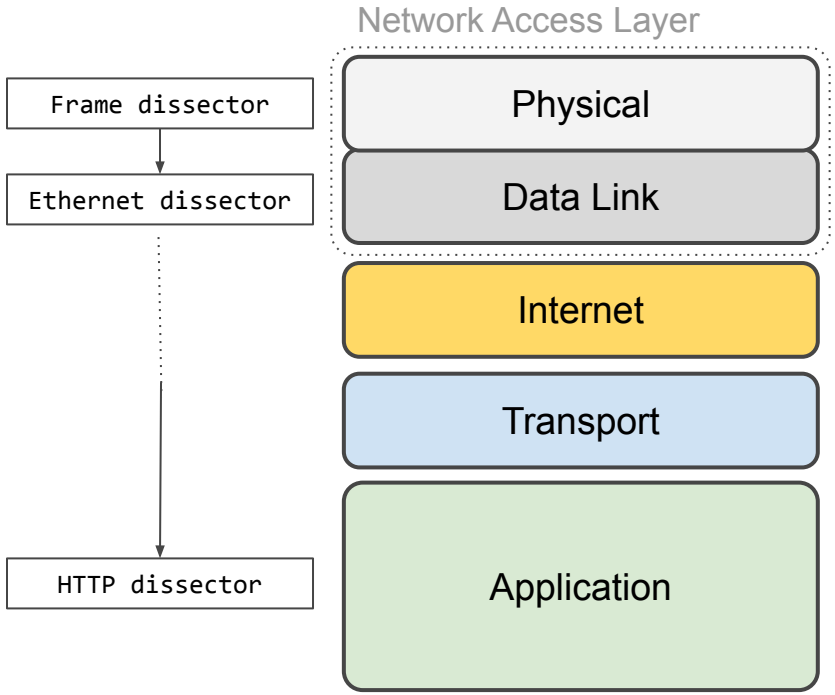


- Wireshark: GUI elements
- Wireshark: working with packets
- Wireshark: follow streams and save artifacts
- Wireshark: packet details pane, dissectors
- Using Wireshark in the command line: Tshark

# Dissectors



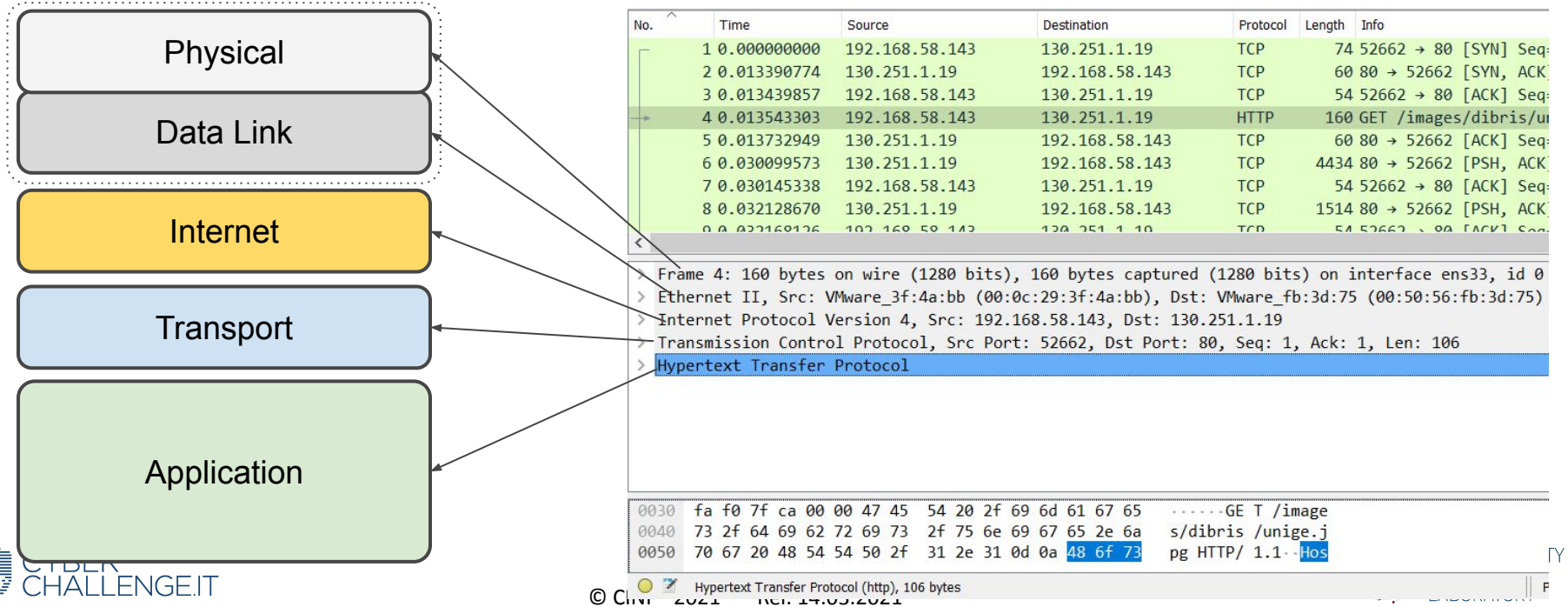
- Dissectors are what parse a protocol and decode it for presenting on the interface
- Each protocol has its own dissector, so dissecting a complete packet will typically involve several dissectors
- Find the right dissector to start decoding the packet data
  - Known conventions (e.g., Ethernet type 0x800 means "IP on top of Ethernet" )
  - Heuristics (e.g., TCP ports)



# Packet details pane: dissectors



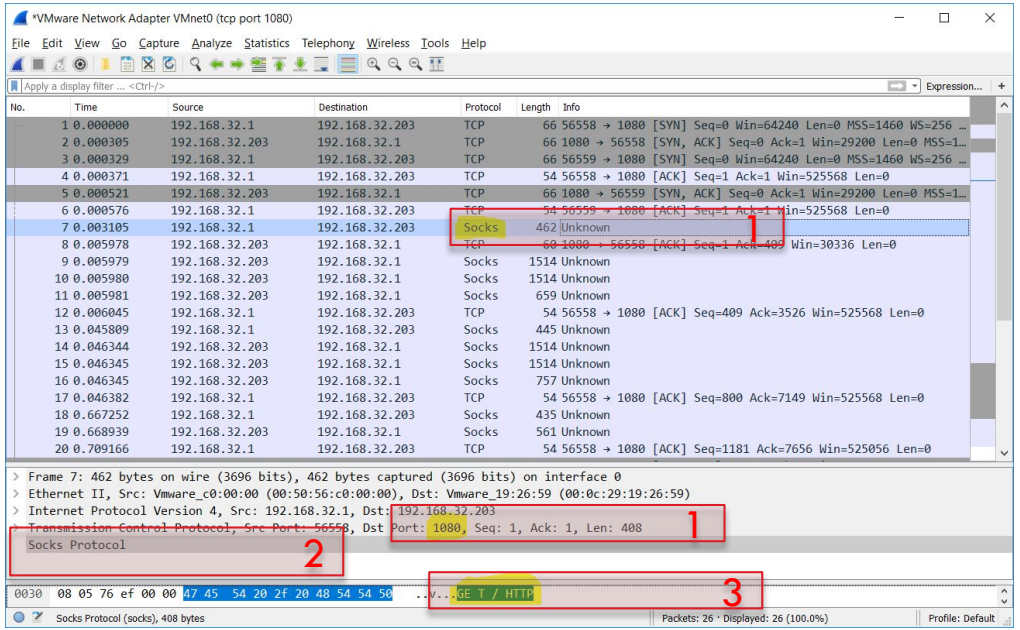
- The packet details pane shows outputs from the applied dissectors



# Change dissection rules (example)



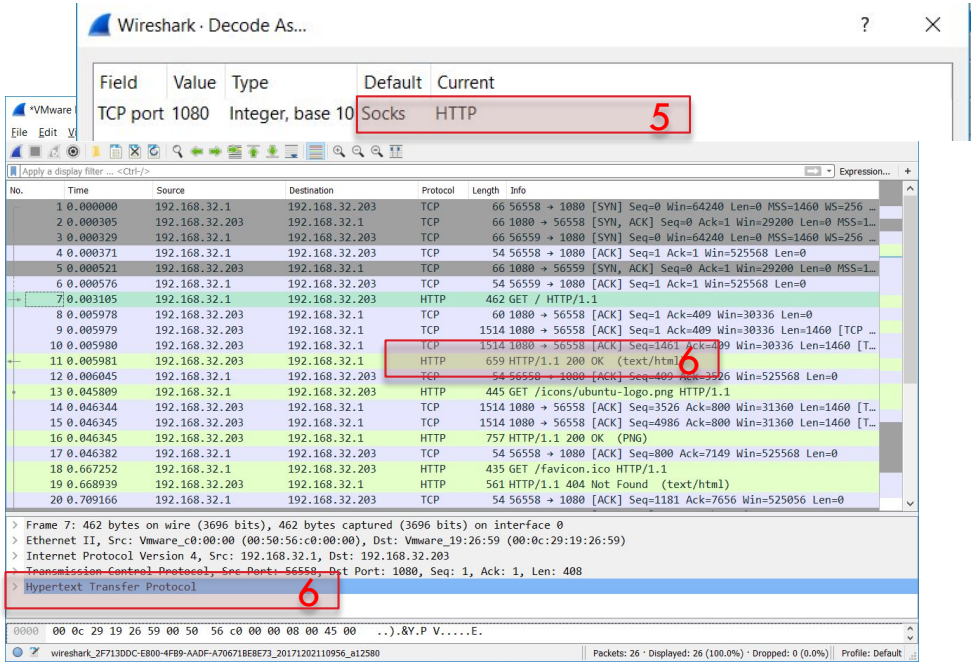
1. Wireshark applies a Socks dissector, as the well-known port for Socks traffic is 1080/tcp
2. The dissector is not able to decode the data correctly (fields are empty in the packet details pane)
3. Raw data contain a request of a GET / HTTP request string.



# Change dissection rules (example)



4. Right click on (one of) the interested packet → Decode As…
5. Change the Current value (Socks) with the right dissector (HTTP)
6. Now protocol fields can be expanded in the packet details pane and visualized on the columns



# Outline



- Wireshark: GUI elements
- Wireshark: working with packets
- Wireshark: follow streams and save artifacts
- Wireshark: packet details pane, dissectors
- Using Wireshark in the command line: Tshark

# Tshark



- TShark is a terminal oriented version of Wireshark
- Designed for capturing and displaying packets
- It supports the same options as Wireshark
  - *tshark –h*: print version and options
  - *man tshark*: linux manual
  - *online*: https://www.wireshark.org/docs/wsug_html_chunked/AppToolstshark.html

# Tshark: examples



- Read PCAP files
  - tshark –r <filename>
- Detail output for specific protocols (available protocols: tshark –G protocols)
  - tshark –O <protocol1>,<protocol2> -r <filename>
- Filter output with a display filter (yank switch)
  - tshark -Y <display_filter_expression> -r <filename>
- Display specific protocols fields (available fields: tshark –G fields)
  - tshark –r <filename>  -T fields –e <field1> … -e <fieldn>
- Convert the hexadecimal payload into a binary files (data carving)
  - tshark [… filtered data payload …] | xxd -r –p > <filename>

# Network Traffic Analysis Via Wireshark

**Enrico RUSSO**
Università di Genova



*https://cybersecnatlab.it*