Network Traffic Analysis Via Wireshark

Enrico RUSSO
Università di Genova

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it

License & Disclaimer

License Information

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

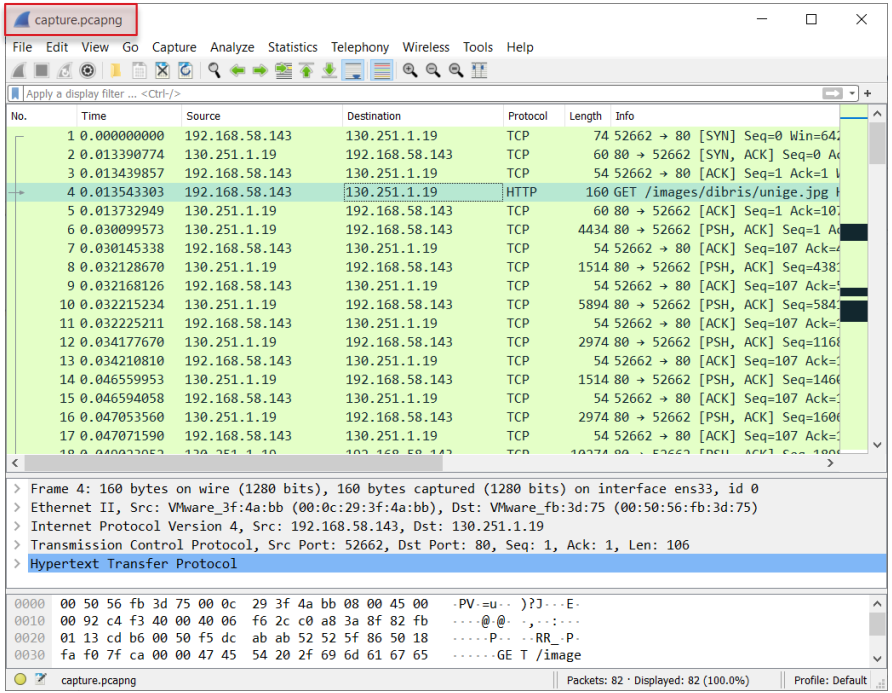
Wireshark

- 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

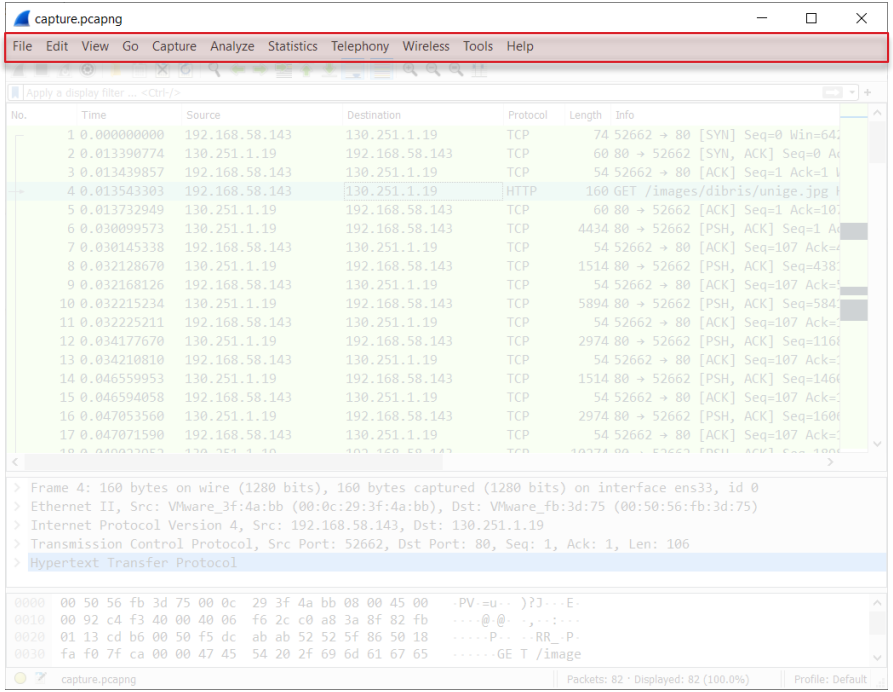
- In some CTF challenges, we are given a PCAP file
- Typically, solving these challenges requires analyzing the capture to find the flag by
  - answering questions related to network traffic
  - carving file from packet streams
- Wireshark is a useful tool for these types of challenges

Wireshark GUI



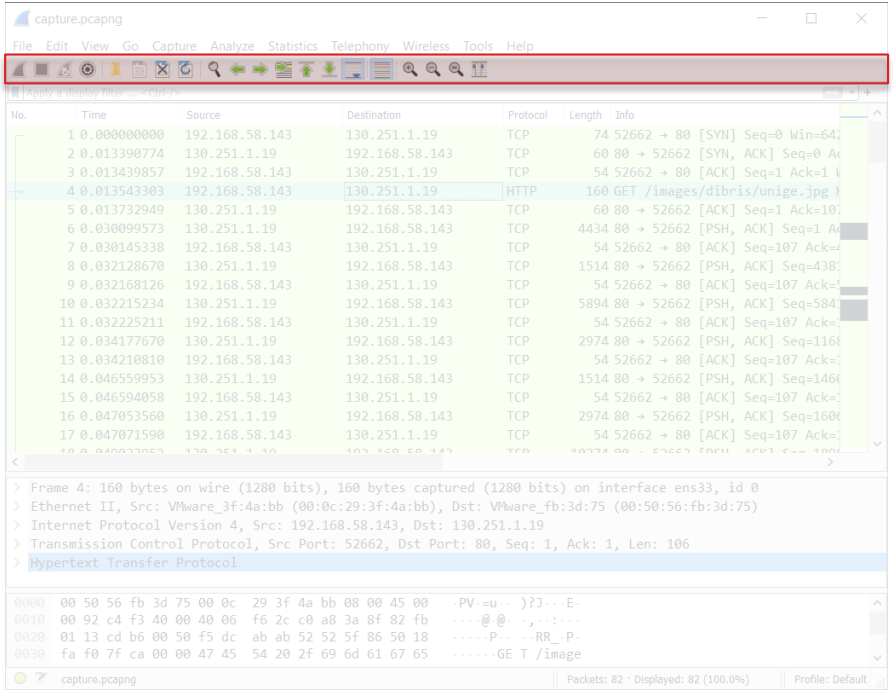
- Wireshark provides a Graphical User Interface (GUI)
- We detail its main elements as it appears after opening an existing PCAP file
  - From the File menu of the Start screen, use the command Open (CTRL-o) and select the PCAP file (e.g., capture.pcapng) to analyze

Wireshark GUI: menu



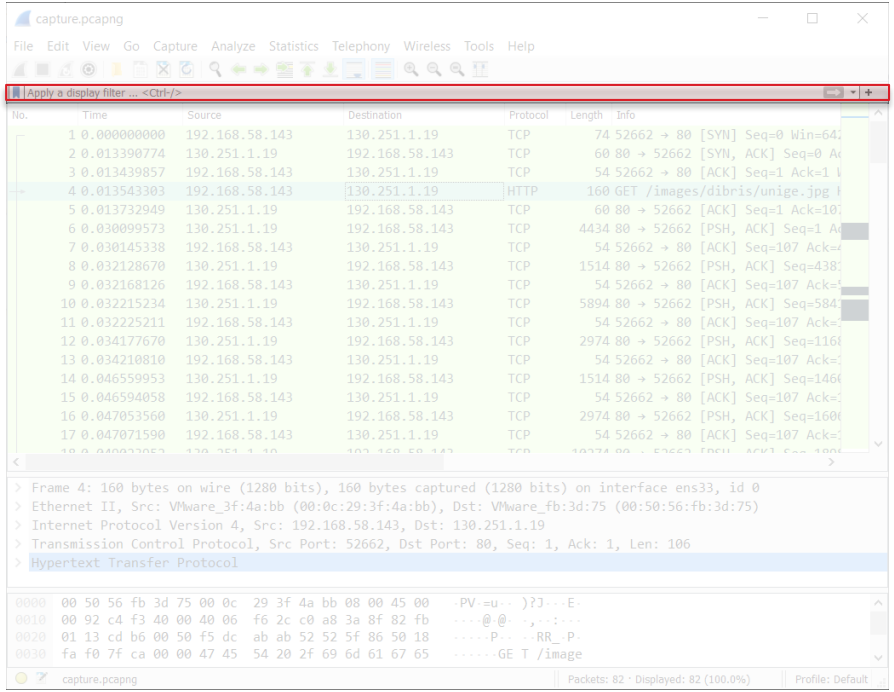
- The **menu** is used to start actions
- Of interest to us are
  - *File*: open and merge capture files, save, print, or export capture
  - *Edit*: find a packet, time reference or mark packets, handle configuration profiles
  - *View*: controls the display of packets (e.g., colorization, name resolution, 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



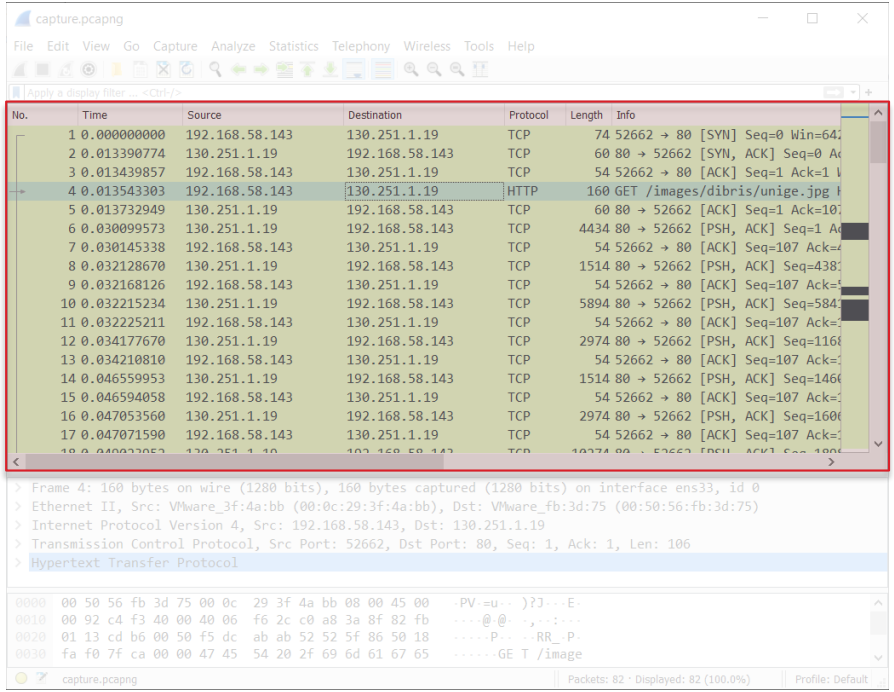
- The **main toolbar** provides quick access to frequently used items from the menu
- Items in the toolbar will be enabled or disabled (greyed out) similar to their corresponding menu items

Wireshark GUI: filter toolbar



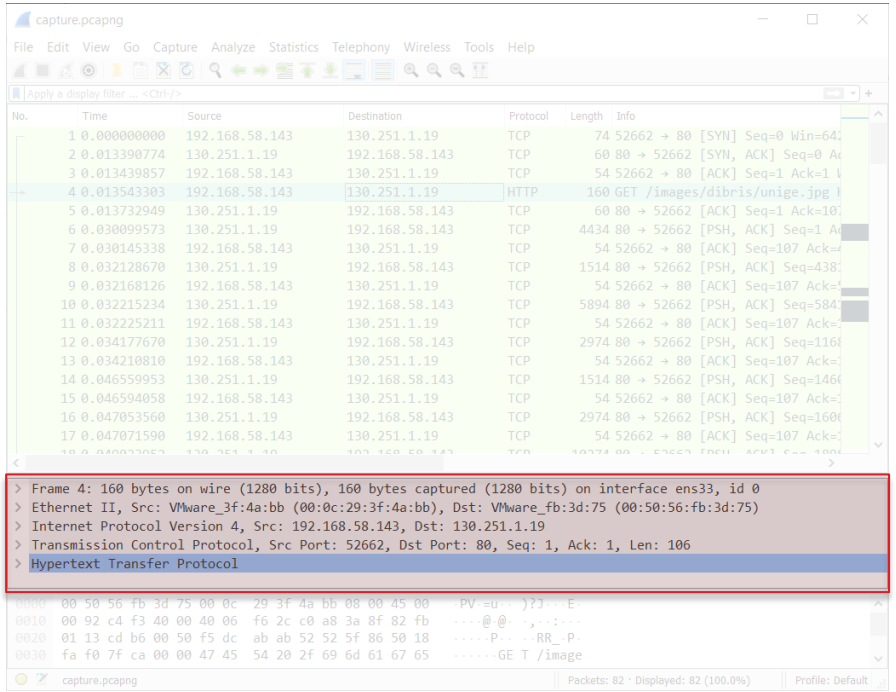
- The **filter toolbar** lets you quickly edit and apply display filters
  - Manage or select saved filters
  - Reset the current display filter and clear the edit area
  - Apply the current value in the edit area as the new display filter
  - Select from a list of recently applied filters
  - Add a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

Wireshark GUI: packet list



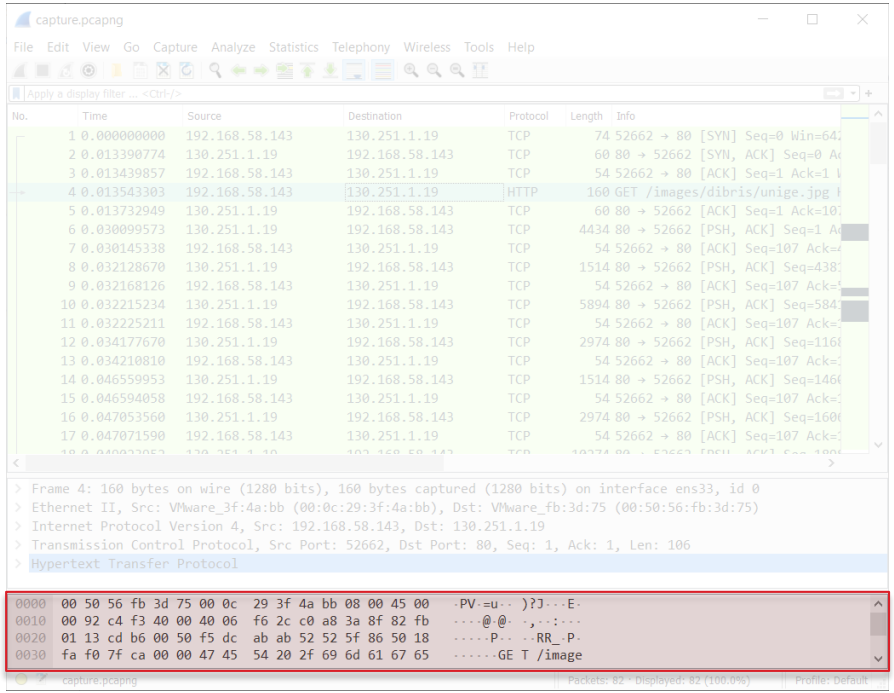
- The **packet list** pane displays a summary of each captured packet
- Each line in the packet list corresponds to one packet in the capture file (selecting a line in this pane displays more details in the *packet details* and *packet bytes* panes)
- Columns provide an overview of the packet
- You can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

Wireshark GUI: packet details



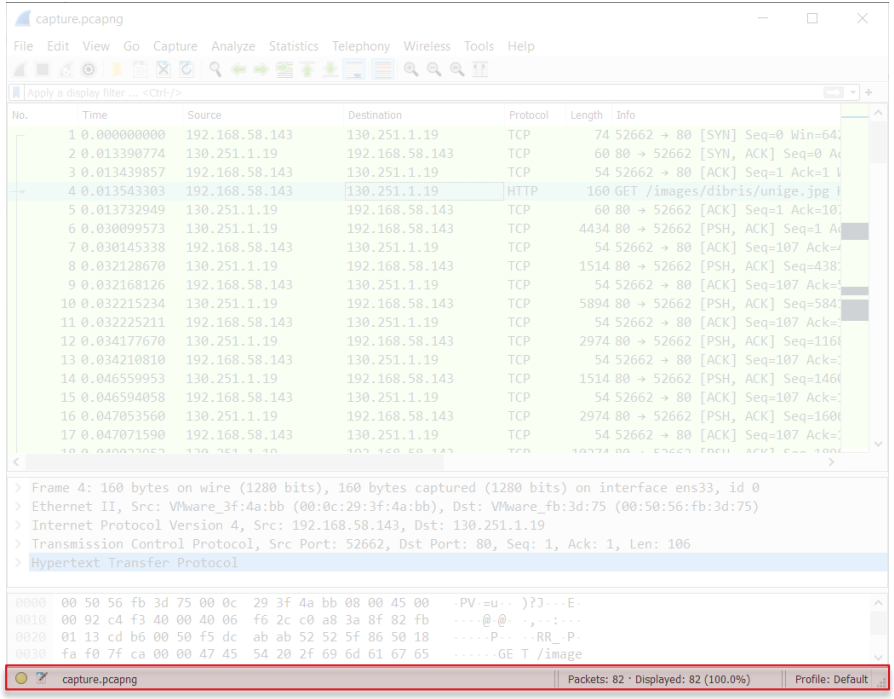
- The **packet details** pane shows the current packet (selected in the packet list pane) in a more detailed form
- In particular, it shows the protocols and fields of the packet in a tree, which can be expanded and collapsed

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

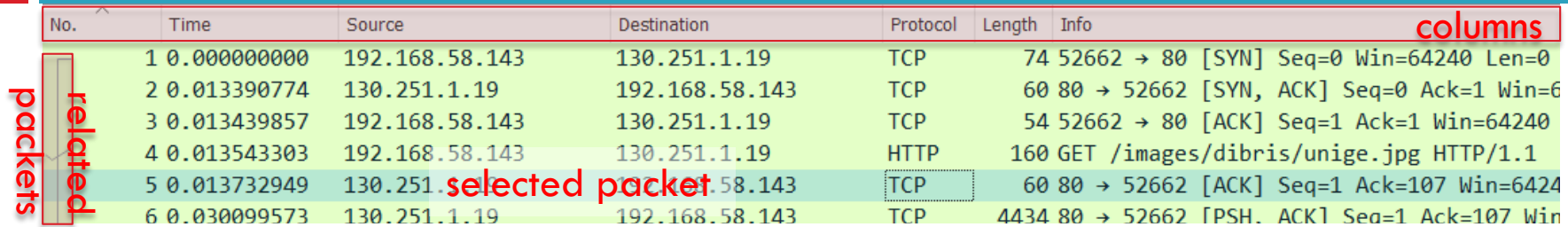


- The **statusbar** displays informational messages
  - The colorized bullet open the Expert Information dialog (list of anomalies and other items of interest found in a capture file)
  - The edit icon lets you add a comment to the capture file
  - The left side shows the file name or protocols fields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

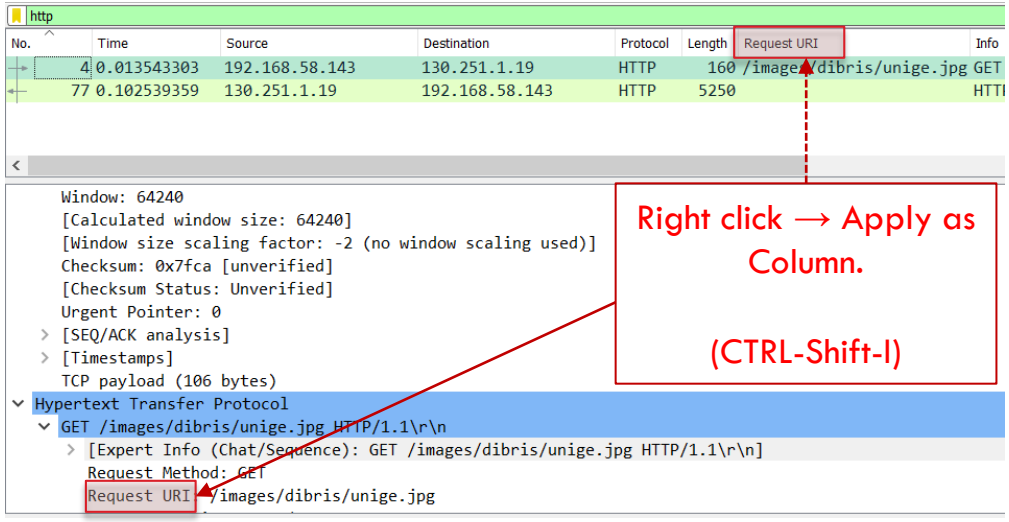
The packet list pane



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.58.143	130.251.1.19	TCP	74	52662 → 80 [SYN] Seq=0 Win=64240 Len=0
2	0.013390774	130.251.1.19	192.168.58.143	TCP	60	80 → 52662 [SYN, ACK] Seq=0 Ack=1 Win=6
3	0.013439857	192.168.58.143	130.251.1.19	TCP	54	52662 → 80 [ACK] Seq=1 Ack=1 Win=64240
4	0.013543303	192.168.58.143	130.251.1.19	HTTP	160	GET /images/dibris/unige.jpg HTTP/1.1
5	0.013732949	130.251.[illegible]	[illegible].58.143	TCP	60	80 → 52662 [ACK] Seq=1 Ack=107 Win=6424
6	0.030099573	130.251.1.19	192.168.58.143	TCP	4434	80 → 52662 [PSH, ACK] Seq=1 Ack=107 Win

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

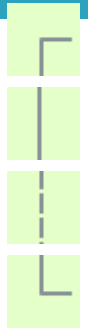
Adding columns (example)



- 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

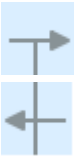
Related packets symbols

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



Stream

- 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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	145.254.160.237	65.208.228.223	TCP	62	3372 → 80 [SYN]
2	0.911310				62	<Ignored>
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK]

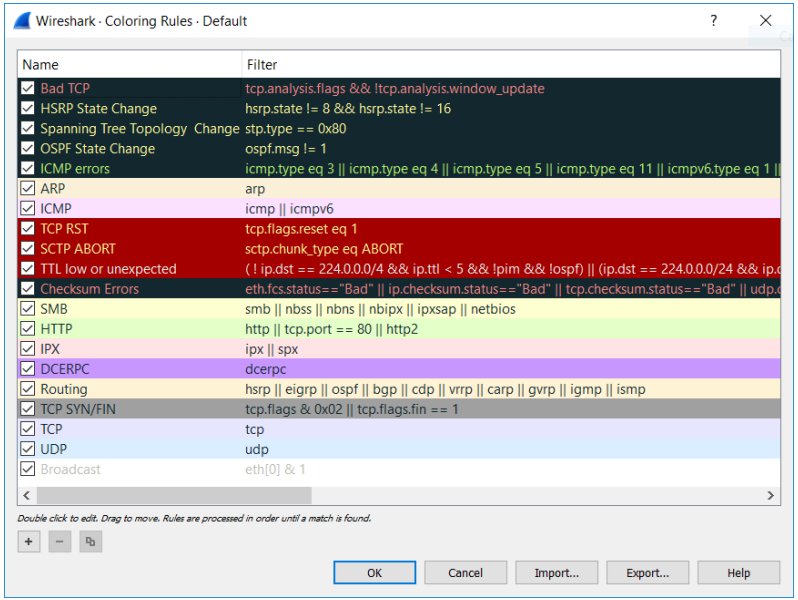
marked
ignored
commented

Packet comments
Commented Comment content

- 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

Executive Summary

- mHackeCTF is a classical attack/defense CTF
- Starting at 17.10.2020, **12:00 UTC**. Network opens at **13:00 UTC**. Game ends at **22:00 UTC**.
- A tick is **4 minutes**, flags are valid for **5 ticks**.
- 1. Flag format: `MHACK\{[A-Za-z0-9-_]{32}\}`
- Flag submission: `nc 10.10.254.254 31337`
- Fax submission: `+39 02 700 31337` both for memes and your best flags.

Packets: 82 · Displayed: 82 (100.0%) 2. Profile: Default

3. Wireshark · Coloring Rules CTF

Name	Filter
4. FLAG	frame matches "MHACK{[A-Za-z0-9-_]{32}}"
Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update && !tcp.analysis.keep_alive && !tcp.analysis.keep_alive_ack
HSRP State Change	hsrp.state != 8 && hsrp.state != 16
Spanning Tree Topology Change	stp.type == 0x80

Coloring rules: A/D CTF example

dump-8080-14_28_42.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1189	65.515391	10.10.33.129	10.10.33.1	TCP	561	36634 → 8080 [PSH, ACK] Seq=3281 Ack=2233 Win=64128 Len
1190	65.518042	10.10.33.129	10.10.33.1	HTTP	339	POST /api/tickets/ HTTP/1.1 (application/vnd.erp.v1.0+
1191	65.518176	10.10.33.1	10.10.33.129	TCP	40	8080 → 36634 [ACK] Seq=2233 Ack=4101 Win=64256 Len=0
1192	65.520258	10.10.33.1	10.10.33.129	TCP	740	8080 → 33188 [PSH, ACK] Seq=14707 Ack=22996 Win=64256 L
1193	65.520621	10.10.33.1	10.10.33.129	HTTP/J…	45	HTTP/1.1 200 , JavaScript Object Notation (application
1194	65.531056	10.10.33.1	10.10.33.129	TCP	593	8080 → 36634 [PSH, ACK] Seq=2233 Ack=4101 Win=64256 Len

Packet with flags (blue lines)

```
> Frame 1192: 740 bytes on wire (5920 bits), 740 bytes captured (5920 bits)
  Raw packet data
> Internet Protocol Version 4, Src: 10.10.33.1, Dst: 10.10.33.129
> Transmission Control Protocol, Src Port: 8080, Dst Port: 33188, Seq: 14707, Ack: 22996, Len: 700
```

```
0240  49 4e 47 22 2c 22 63 6f  6e 74 65 6e 74 22 3a 22   ING","co ntent":"
0250  4d 48 41 43 4b 7b 42 51  41 72 41 41 55 41 5a 6a   MHACK{BQ ArAAUAZj
0260  72 49 4f 5a 77 73 57 68  66 5a 6c 56 55 72 39 43   rIOZwsWh fZlVUr9C
0270  66 57 75 6f 51 56 7d 22  2c 22 61 75 74 68 6f 72   fWuoQV}" ,"author
0280  22 3a 22 61 64 6d 69 6e  22 2c 22 61 73 73 69 67   ":"admin ","assig
0290  6e 65 64 54 6f 22 3a 6e  75 6c 6c 2c 22 70 72 69   nedTo":n ull,"pri
02a0  6f 72 69 74 79 22 3a 22  48 49 47 48 22 2c 22 63   ority":" HIGH","c
02b0  72 65 61 74 65 64 41 74  22 3a 22 32 30 32 30 2d   reatedAt ":"2020-
02c0  31 30 2d 31 37 54 31 33  3a 32 31 3a 35 39 5a 22   10-17T13 :21:59Z"
02d0  2c 22 75 70 64 61 74 65  64 41 74 22 3a 6e 75 6c   ,"update dAt":nul
02e0  6c 7d 0d 0a                                        l}··
```

flag

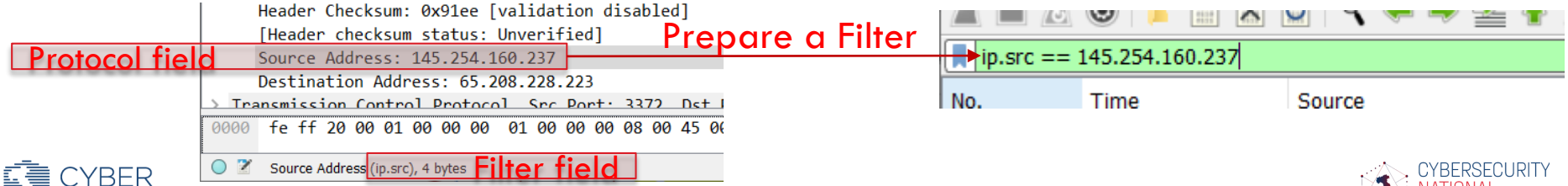
dump-8080-14_28_42.pcap | Packets: 3775 · Displayed: 3775 (100.0%) | Profile: CTF

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)

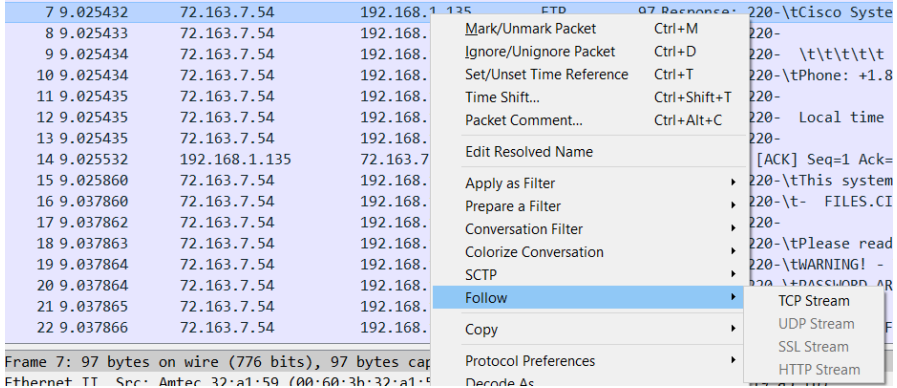


Outline

- 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



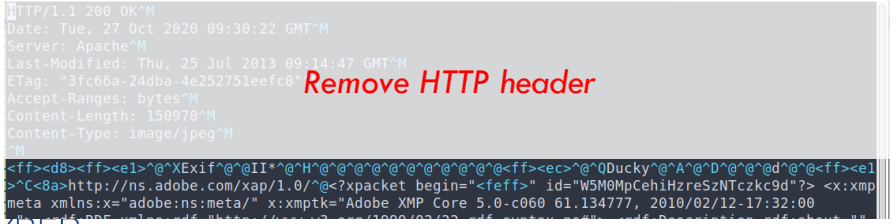
Follow streams: example

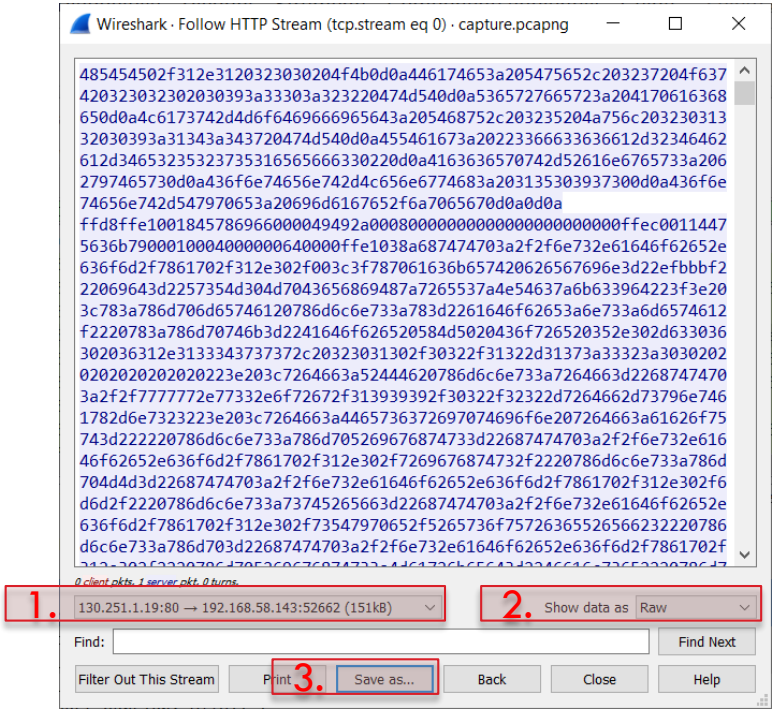
- Telnet is a type of client-server protocol that can be used to open a command line on a remote host
- *Blue* is the data from the server to the client (e.g., the *login:* prompt)
- *Red* is the data from the client to the server (e.g., the user password is sent by the client and is not echoed by the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

```
........... ..!.."..'.....#..%..%........... ..!.."..".........P. ....".....b........b....  B.
......................."......'.....#..&..&..$..&..&..$.. .....#.....'........... .9600,9600....#.bam.zing.org:
0.0....'..DISPLAY.bam.zing.org:0.0......xterm-color.............!..............".............
OpenBSD/i386 (oof) (ttyp1)

login: .."........"ffaakkee
.
Password:user
.
Last login: Thu Dec  2 21:32:59 on ttyp1 from bam.zing.org
Warning: no Kerberos tickets issued.
OpenBSD 2.6-beta (OOF) #4: Tue Oct 12 20:42:32 CDT 1999

Welcome to OpenBSD: The proactively secure Unix-like operating system.

Please use the sendbug(1) utility to report bugs in the system.
Before reporting a bug, please try to reproduce it with the latest
version of the code.  With bug reports, please try to ensure that
enough information to reproduce the problem is enclosed, and if a
known fix for it exists, include that as well.

$ llss
.
$ llss  --aa
.
.         ..         .cshrc    .login    .mailrc   .profile  .rhosts
$ //ssbbiinn//ppiinngg  wwwwww..yyaahhoooo..ccoomm
.
PING www.yahoo.com (204.71.200.74): 56 data bytes
64 bytes from 204.71.200.74: icmp_seq=0 ttl=239 time=73.569 ms
64 bytes from 204.71.200.74: icmp_seq=1 ttl=239 time=71.099 ms
64 bytes from 204.71.200.74: icmp_seq=2 ttl=239 time=68.728 ms
64 bytes from 204.71.200.74: icmp_seq=3 ttl=239 time=73.122 ms
64 bytes from 204.71.200.74: icmp_seq=4 ttl=239 time=71.276 ms
64 bytes from 204.71.200.74: icmp_seq=5 ttl=239 time=75.831 ms
64 bytes from 204.71.200.74: icmp_seq=6 ttl=239 time=70.101 ms
64 bytes from 204.71.200.74: icmp_seq=7 ttl=239 time=74.528 ms
64 bytes from 204.71.200.74: icmp_seq=9 ttl=239 time=74.514 ms
64 bytes from 204.71.200.74: icmp_seq=10 ttl=239 time=75.188 ms
64 bytes from 204.71.200.74: icmp_seq=11 ttl=239 time=72.925 ms
...^C
.--- www.yahoo.com ping statistics ---
13 packets transmitted, 11 packets received, 15% packet loss
round-trip min/avg/max = 68.728/72.807/75.831 ms
$ eexxiitt
.
```


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

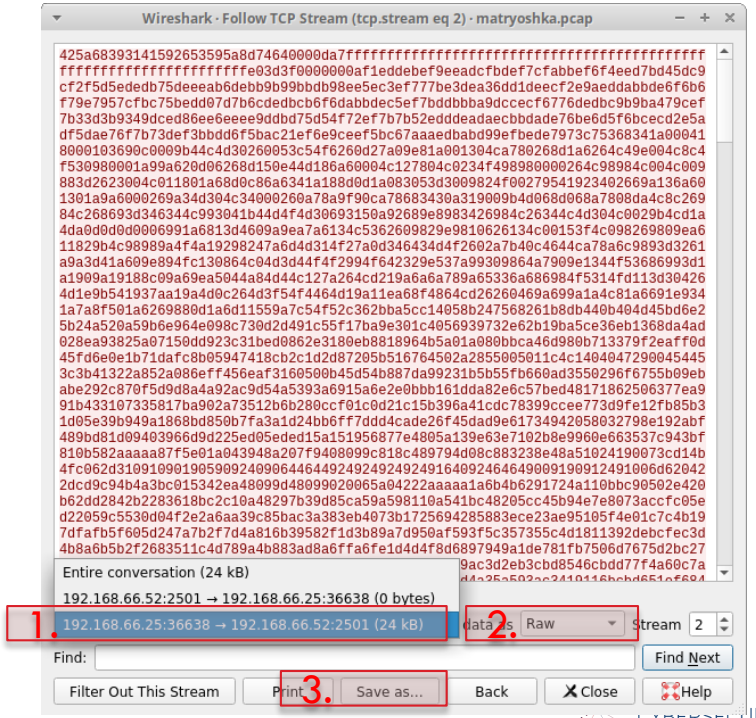




Carve files from streams: example 2

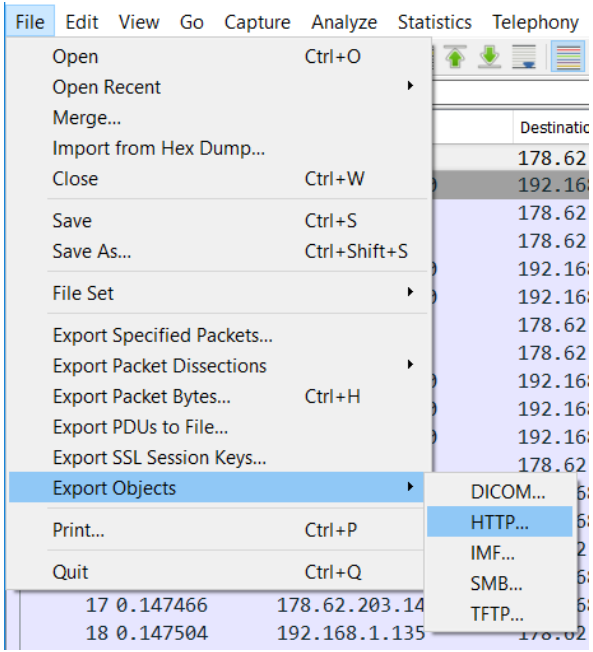
- Extract a file from a generic stream (unknown protocol)
  1. Switch between client to server or server to client conversation
  2. Show data as Raw
  3. Save as... (e.g., */tmp/bin1*)
  4. Use the linux *file* utility to determine the file type

```
/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

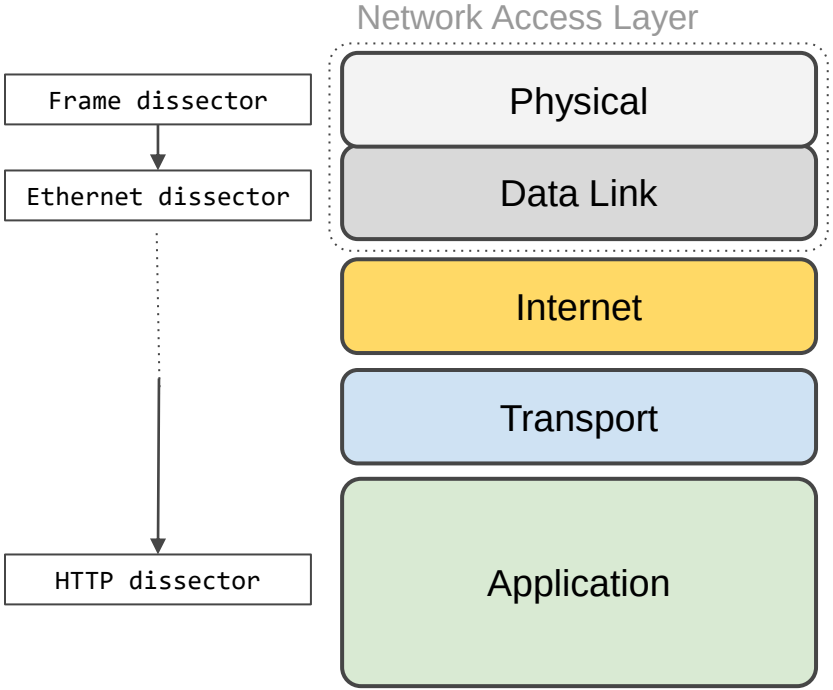


Outline

- 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

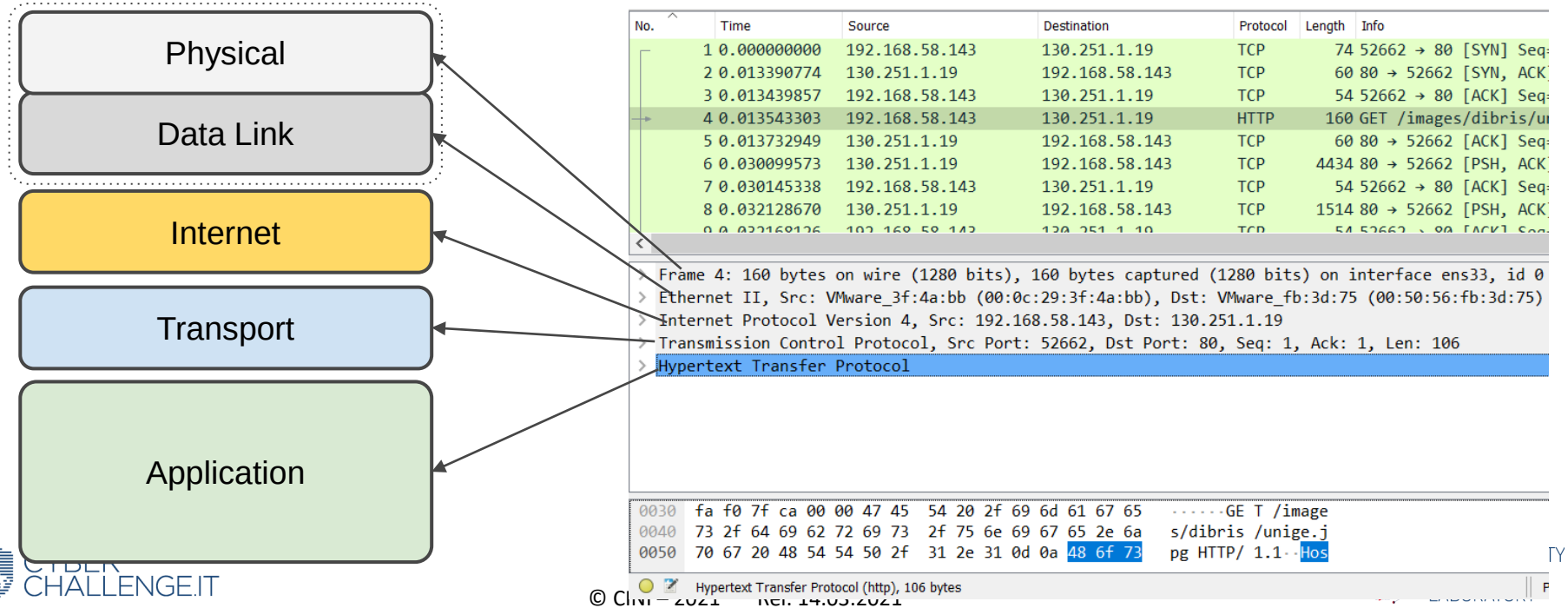
Dissectors

- 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)



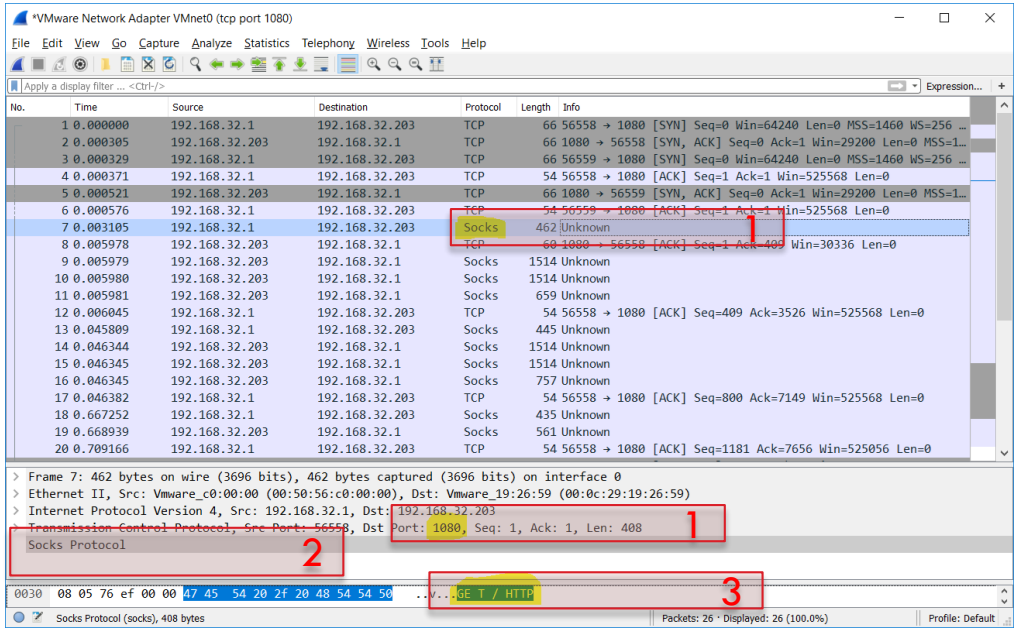
Packet details pane: dissectors

- The packet details pane shows outputs from the applied dissectors



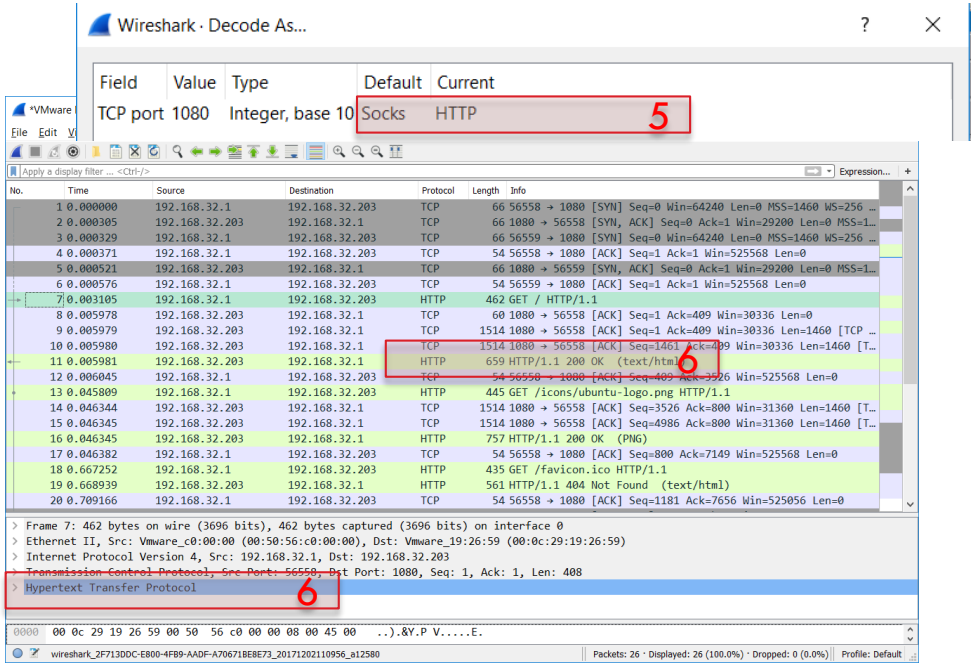
Change dissection rules (example)

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

CYBER CHALLENGE.IT

CYBERSECURITY NATIONAL LABORATORY

https://cybersecnatlab.it