Network Traffic Analysis via Wireshark

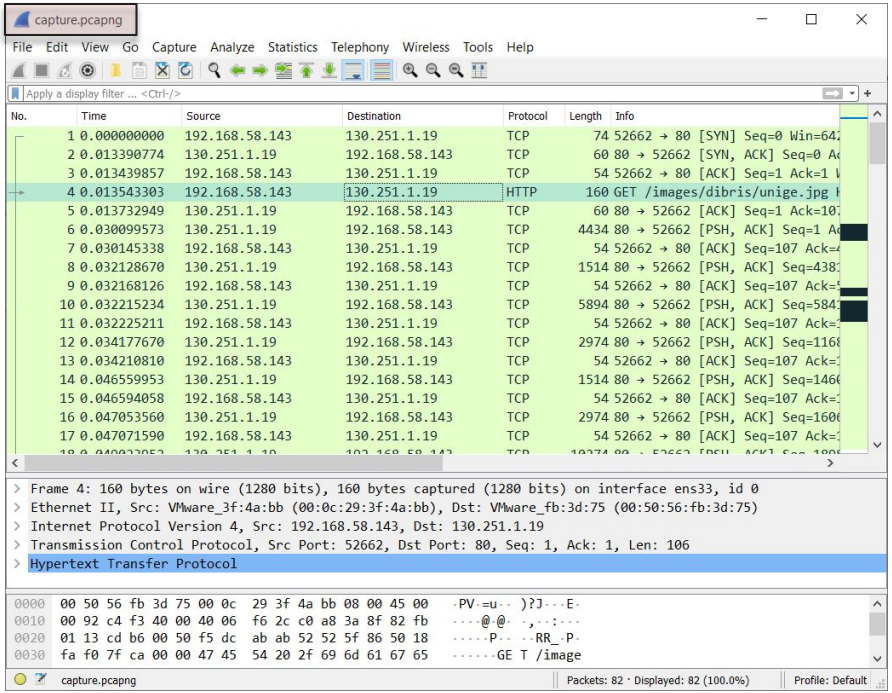
Digital Forensics - ay 2024/2025

Enrico RUSSO <enrico.russo@dibris.unige.it>

Wireshark

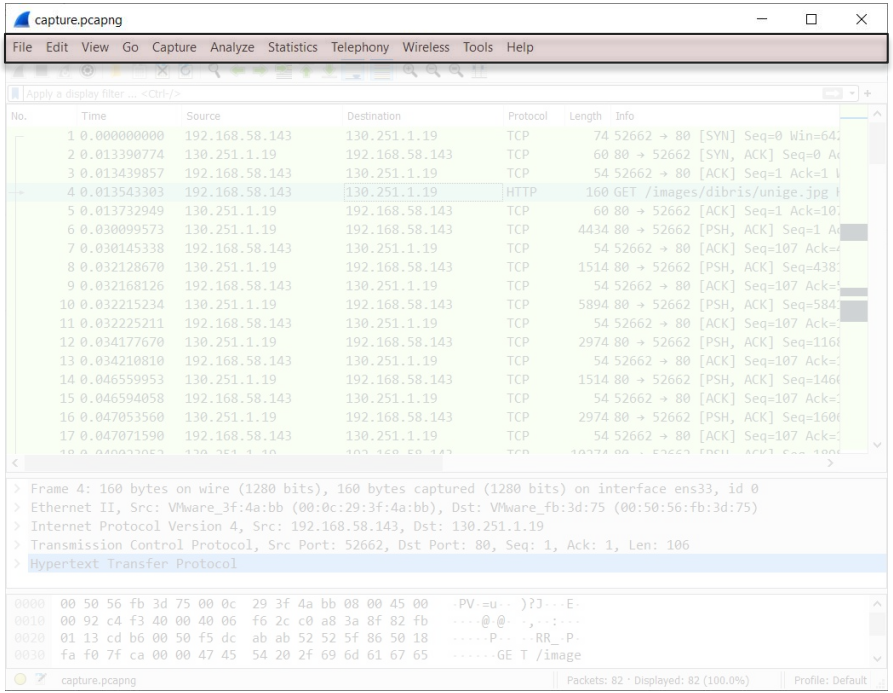
- Wireshark is a tool to capture data from a network (sniffer) and to analyse them
  - Analysis can be performed in real-time or on previously-recorded traffic files, through, e.g., *packet* capture or PCAP
  - Packets represent *generic* chunks of data and, depending on the considered level, can be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

Wireshark GUI



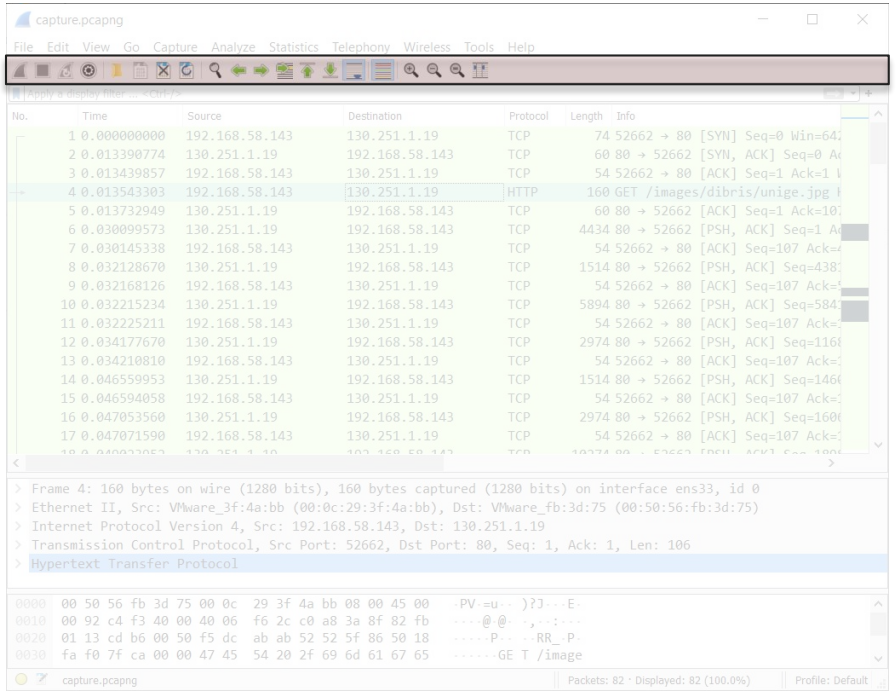
- Wireshark provides a Graphical User Interface (GUI)
- We detail its main elements as it appears after opening an existing PCAP file
  - From the File menu of the Start screen, use the command Open (CTRL-o) and select the PCAP file (e.g., capture.pcapng) to analyze

Wireshark GUI: menu



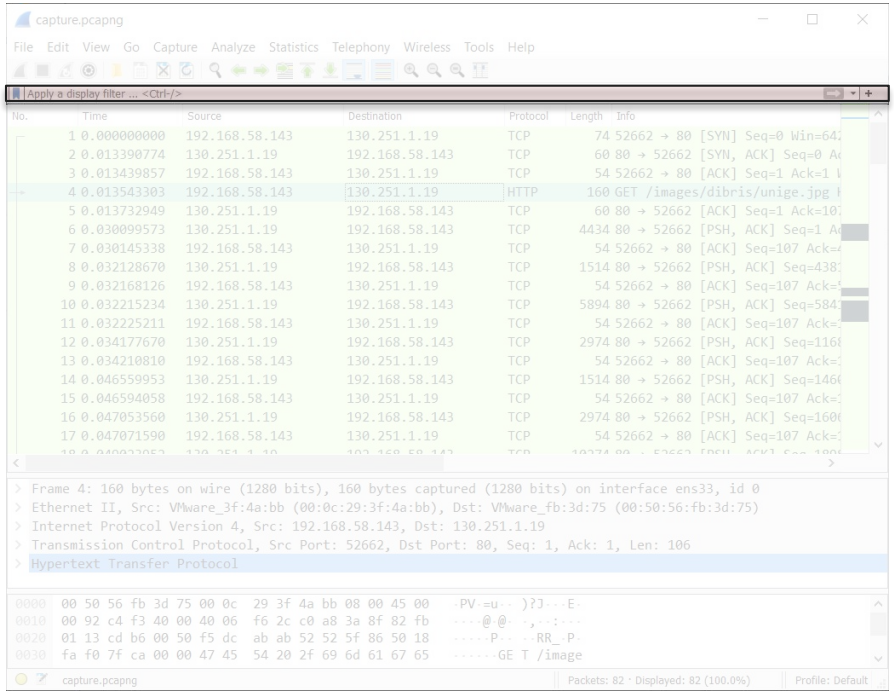
- The **menu** is used to start actions
- Of interest to us are
  - *File*: open and merge capture files, save, print, or export capture
  - *Edit*: find a packet, time reference or mark packets, handle configuration profiles
  - *View*: controls the display of packets (e.g., colorization, name resolution,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



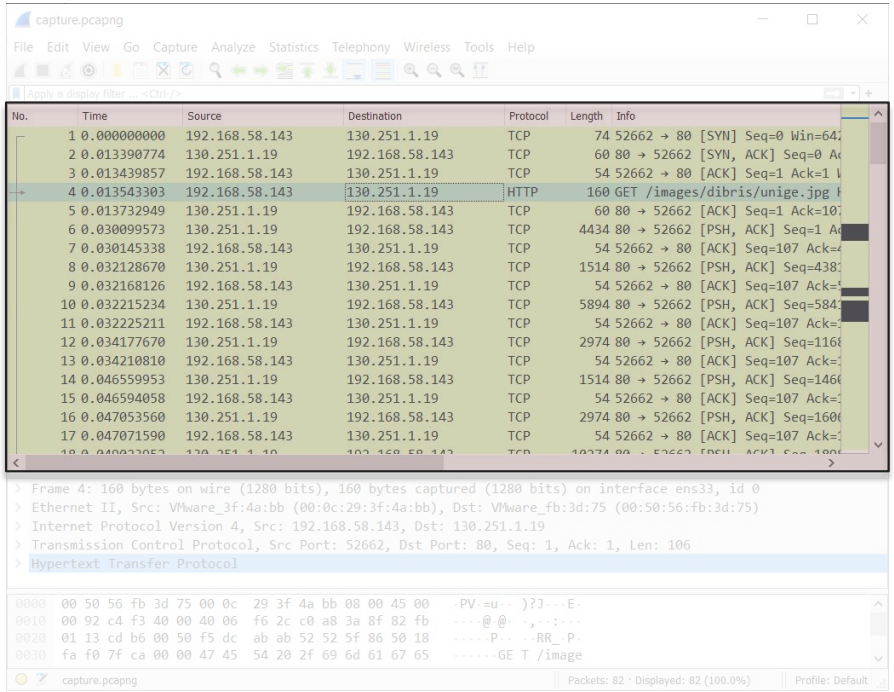
- The **main toolbar** provides quick access to frequently used items from the menu
- Items in the toolbar will be enabled or disabled (greyed out) similar to their corresponding menu items

Wireshark GUI: filter toolbar



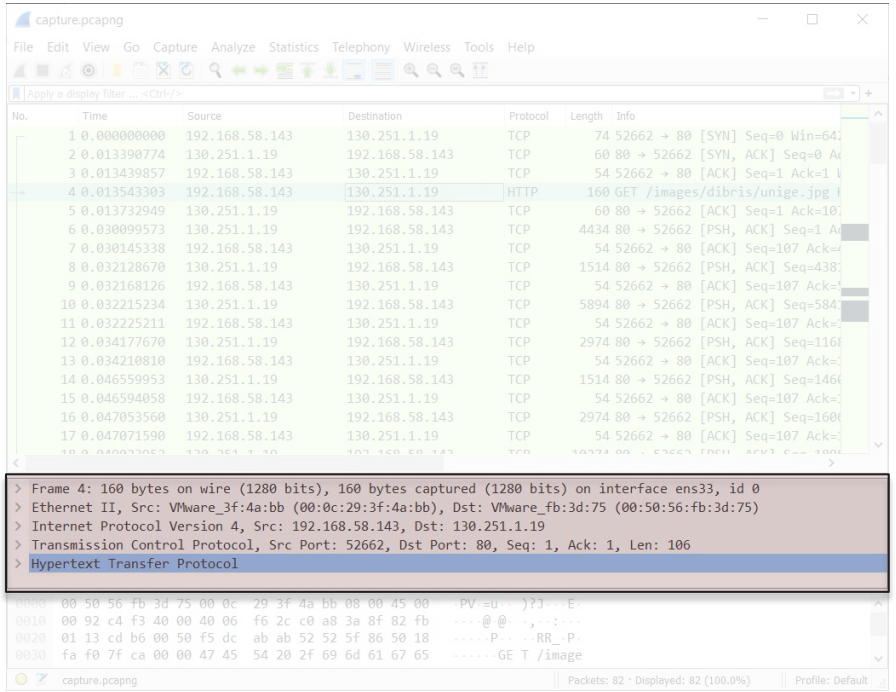
- The **filter toolbar** lets you quickly edit and apply display filters
  - Manage or select saved filters
  - Reset the current display filter and clear the edit area
  - Apply the current value in the edit area as the new display filter
  - Select from a list of recently applied filters
  - Add a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

Wireshark GUI: packet list



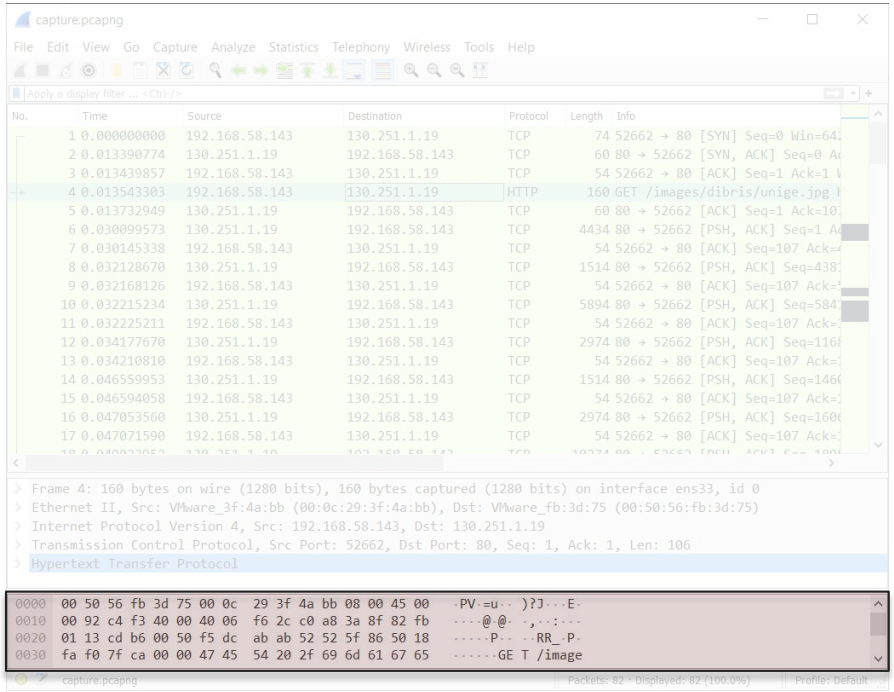
- The **packet list** pane displays a summary of each captured packet
- Each line in the packet list corresponds to one packet in the capture file (selecting a line in this pane displays more details in the *packet details* and *packet bytes* panes)
- Columns provide an overview of the packet
- You can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

Wireshark GUI: packet details



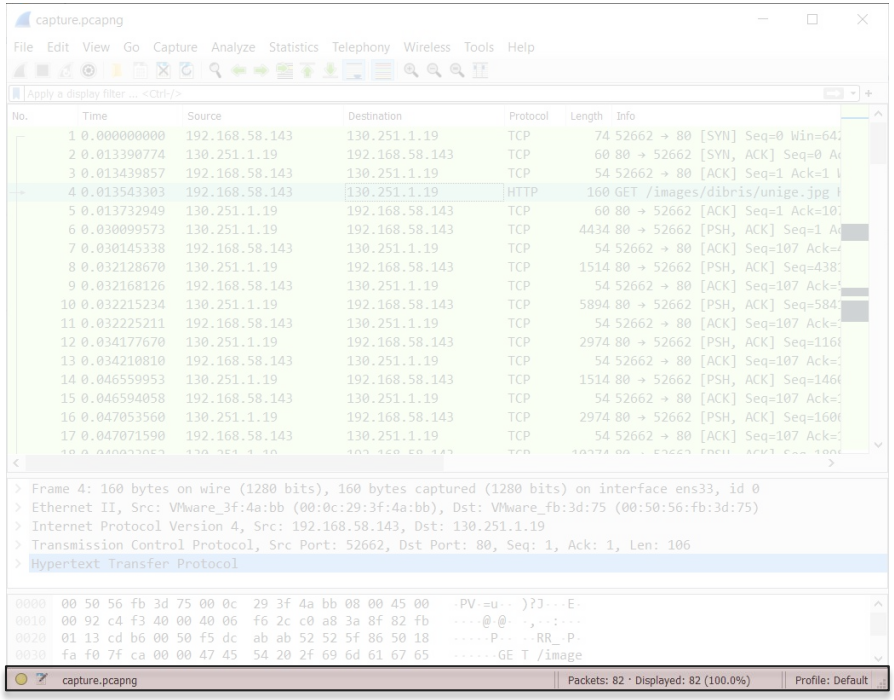
- The **packet details** pane shows the current packet (selected in the packet list pane) in a more detailed form
- In particular, it shows the protocols and fields of the packet in a tree, which can be expanded and collapsed

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

Wireshark: statistical data

Statistics menu provides different statistical data. Of particular interest is the "Conversations" item.

Ethernet · 2 | IPv4 · 68 | IPv6 | TCP · 1076 | UDP

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration
203.20.123.123	198.51.100.100	28,042	237 M	18,205	991 k	9,837	236 M	57.214411	54.8824
44.10.12.77	198.51.100.100	2,011	112 k	1,006	58 k	1,005	54 k	103.963289	11.6744
76.188.201.228	198.51.100.100	30	3.193	18	1.645	12	1.548	14.770688	83.4199
121.15.145.113	198.51.100.100	20	2.227	12	1.195	8	1.032	37.858108	58.1707
20.151.223.124	198.51.100.100	20	2.170	12	1.138	8	1.032	82.109152	15.0391
184.179.109.137	198.51.100.100	20	2.137	12	1.105	8	1.032	89.623414	4.3573
69.19.203.242	198.51.100.100	20	2.134	12	1.102	8	1.032	92.908816	15.8214
157.0.60.196	198.51.100.100	20	2.124	12	1.092	8	1.032	30.448629	6.3112
11.79.141.92	198.51.100.100	10	1.116	6	600	4	516	57.833839	0.0100
60.35.144.252	198.51.100.100	10	1.116	6	600	4	516	26.257560	0.0015
64.127.79.7	198.51.100.100	10	1.116	6	600	4	516	55.771633	0.0006

most active (or outliers)

Wireshark: statistical data

Ethernet · 2 IPv4 · 68 IPv6 TCP · 1076 UDP

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Sta
4.156.230.63	53352	198.51.100.100	80	10	1.104	6	588	4	516	80.07
11.0.196.106	37092	198.51.100.100	80	10	1.088	6	572	4	516	37.81
11.79.141.92	36480	198.51.100.100	80	10	1.116	6	600	4	516	57.83
14.139.104.9	36432	198.51.100.100	80	10	1.055	6	539	4	516	52.56
20.151.223.124	53366	198.51.100.100	80	10	1.066	[illegible]	[illegible]	[illegible]	516	82.10
20.151.223.124	45428	198.51.100.100	80	10	1.104	6	588	4	516	97.13
21.89.215.163	37286	198.51.100.100	80	10	973	6	457	4	516	40.96
37.2.104.204	52182	198.51.100.100	80	10	1.071	6	555	4	516	103.4
39.245.232.178	58290	198.51.100.100	80	10	1.091	6	575	4	516	0.00
44.10.12.77	59063	198.51.100.100	443	2	112	1	58	1	54	115.48
44.10.12.77	59063	198.51.100.100	80	2	108	1	54	1	54	115.48
44.10.12.77	59319	198.51.100.100	445	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	21	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	1025	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	53	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	110	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	22	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	139	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	443	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	995	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	993	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	3389	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	8888	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	113	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	135	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	256	2	112	1	58	1	54	115.5

web server (well-known port)

Wireshark: statistical data

Ethernet · 2 IPv4 · 68 IPv6 TCP · 1076 UDP

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Sta
4.156.230.63	53352	198.51.100.100	80	10	1.104	6	588	4	516	80.07
11.0.196.106	37092	198.51.100.100	80	10	1.088	6	572	4	516	37.81
11.79.141.92	36480	198.51.100.100	80	10	1.116	6	600	4	516	57.83
14.139.104.9	36432	198.51.100.100	80	10	1.055	6	539	4	516	52.56
20.151.223.124	53366	198.51.100.100	80	10	1.06[illegible]	6	550	4	516	82.10
20.151.223.124	45428	198.51.100.100	80	10	1.104	6	588	4	516	97.13
21.89.215.163	37286	198.51.100.100	80	10	973	6	457	4	516	40.96
37.2.104.204	52182	198.51.100.100	80	10	1.071	6	555	4	516	103.4
39.245.232.178	58290	198.51.100.100	80	10	1.091	6	575	4	516	0.00
44.10.12.77	59063	198.51.100.100	443	2	112	1	58	1	54	115.48
44.10.12.77	59063	198.51.100.100	80	2	108	1	54	1	54	115.48
44.10.12.77	59319	198.51.100.100	445	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	21	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	1025	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	53	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	110	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	22	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	139	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	443	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	995	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	993	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	3389	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	8888	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	113	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	135	2	112	1	58	1	54	115.54
44.10.12.77	59319	198.51.100.100	256	2	112	1	58	1	54	115.54

clients (ephemeral ports)

Wireshark: statistical data

Ethernet · 2 IPv4 · 68 IPv6 TCP · 1076 UDP

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Sta
4.156.230.63	53352	198.51.100.100	80	10	1.104	6	588	4	516	80.07
11.0.196.106	37092	198.51.100.100	80	10	1.088	6	572	4	516	37.81
11.79.141.92	36480	198.51.100.100	80	10	1.116	6	600	4	516	57.83
14.139.104.9	36432	198.51.100.100	80	10	1.055	6	539	4	516	52.56
20.151.223.124	53366	198.51.100.100	80	10	1.066	6	550	4	516	82.10
20.151.223.124	45428	198.51.100.100	80	10	1.104	6	588	4	516	97.13
21.89.215.163	37286	198.51.100.100	80	10	973	6	457	4	516	40.96
37.2.104.204	52182	198.51.100.100	80	10	1.071	6	555	4	516	103.4
39.245.232.178	58290	198.51.100.100	80	10	1.091	6	575	4	516	0.00
44.10.12.77	59063	198.51.100.100	443	2	112	1	58	1	54	115.4
44.10.12.77	59063	198.51.100.100	80	2	108	1	54	1	54	115.4
44.10.12.77	59319	198.51.100.100	445	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	21	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	1025	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	53	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	110	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	22	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	139	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	443	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	995	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	993	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	3389	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	8888	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	113	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	135	2	112	1	58	1	54	115.5
44.10.12.77	59319	198.51.100.100	256	2	112	1	58	1	54	115.5

portscan?

Wireshark ex.1: statistical data

For all exercises, use the PCAP available at
https://github.com/enricorusso/DF_Exs/blob/main/netfor/wireshark_tutorial.pcapng

Consider the network with address 198.51.100.0/24 as the one under analysis.

Ex.1: using *only* statistical data, try listing

- servers and exposed services
- most active hosts/outliers
- possible malicious actors

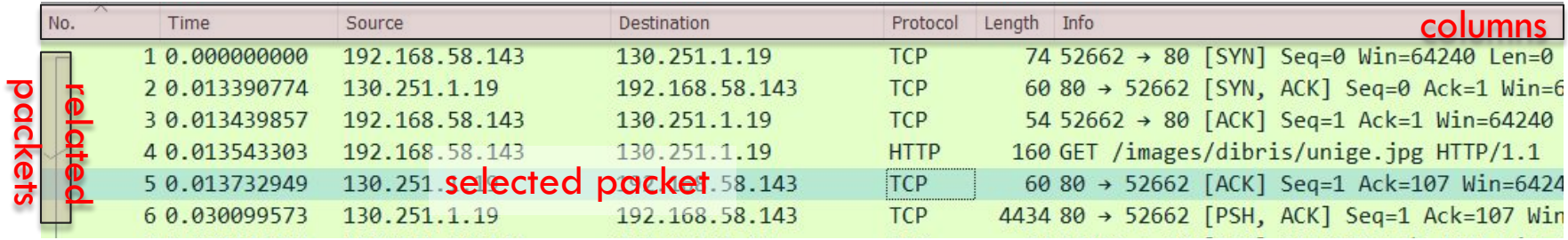
Time

- Each received frame (in this context, the term frame is used to refer to physical layer messages) gets encapsulated with extra header information during the packet capture
  - the time that gets recorded in the header timestamp is provided **by the clock on the machine performing the packet capture**

```
v Frame 1: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface ens38, id 0
    Section number: 1
  > Interface id: 0 (ens38)
    Encapsulation type: Ethernet (1)
    Arrival Time: Dec 21, 2017 21:41:47.216812286 W. Europe Standard Time
```

- By default, Wireshark displays all time stamps in absolute time (seconds) since the beginning of the capture
- Time display format can be changed from View > Time Display Format

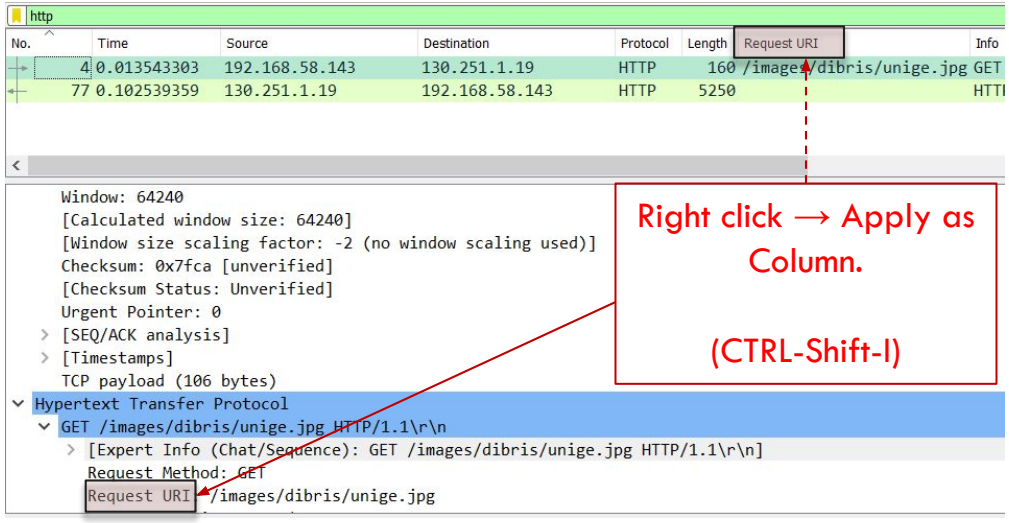
The packet list pane



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	.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)

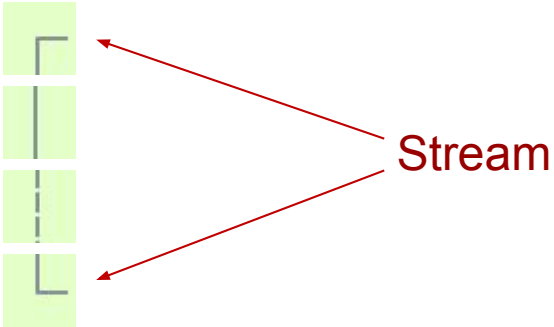


Ex.2: add a column containing the HTTP request method

- 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)

Ex.3: find a stream, a request/response, an acknowledgment

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
[Expert Info (Comment/Comment): Commented]

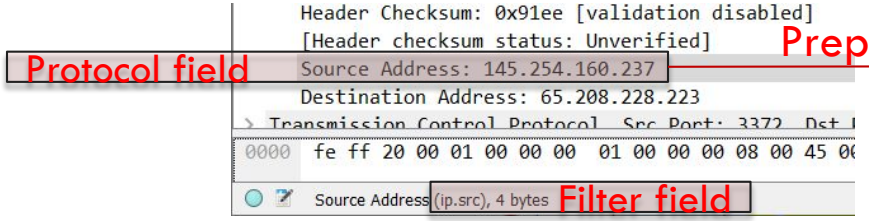
- mark packets of particular interest:
  - CTRL-M
  - jump forward and backward between marked packets: press SHIFT-CTRL-N and SHIFT-CTRL-B respectively
- ignore packets:
  - CTRL-D
- comment packets:
  - CTRL-ALT-C

Display filters: filtering packets

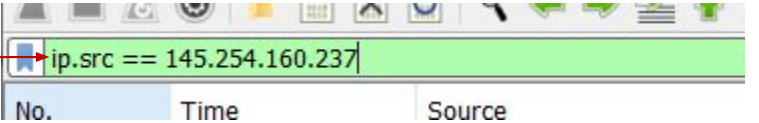
- Wireshark provides a display filter language that enables you to precisely control which packets are displayed
- They can be used to check for
  - the presence of a protocol or field
  - the value of a field
  - compare two fields to each other
- These comparisons can be combined with logical operators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

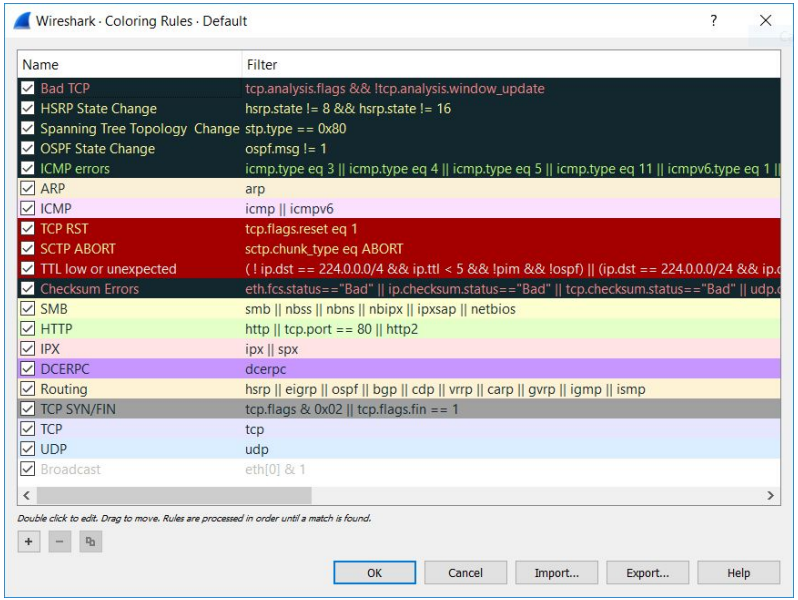


Prepare a Filter



Coloring rules

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

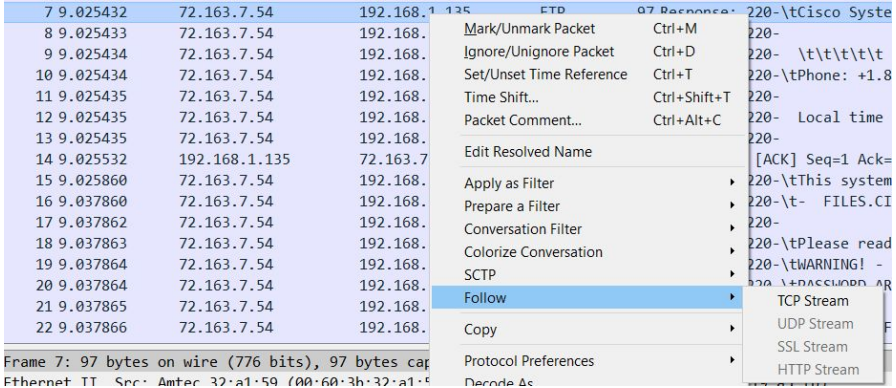


Ex.4

- create a filter that will display all of the POST actions of forms that contain an element with the name "login"
  - what logins and passwords are being used?
- create a filter that displays open ports found during portscan
- filter ICMP traffic and try performing a passive OS fingerprinting (for example, see http://blog.alan-kelly.ie/blog/payload_comparsion/)

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



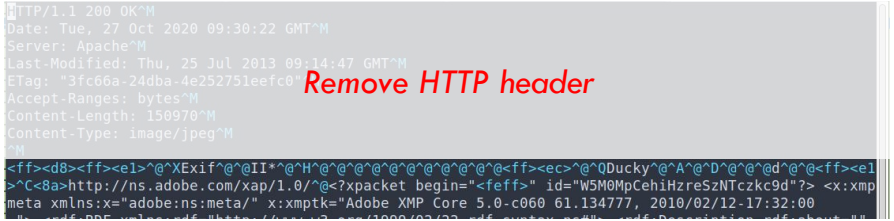
Follow streams: example

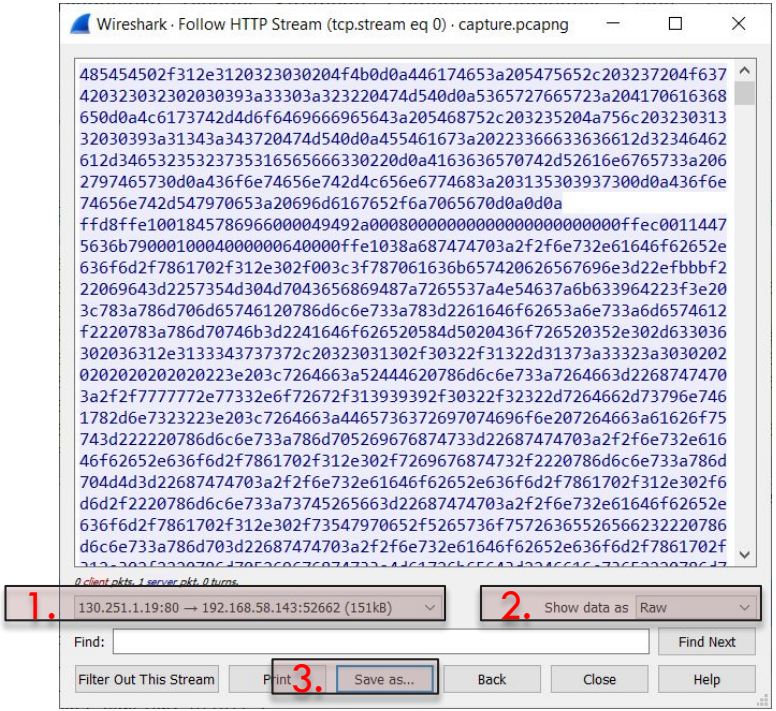
- Telnet is a type of client-server protocol that can be used to open a command line on a remote host
- *Blue* is the data from the server to the client (e.g., the *login:* prompt)
- *Red* is the data from the client to the server (e.g., the user password is sent by the client and is not echoed by the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

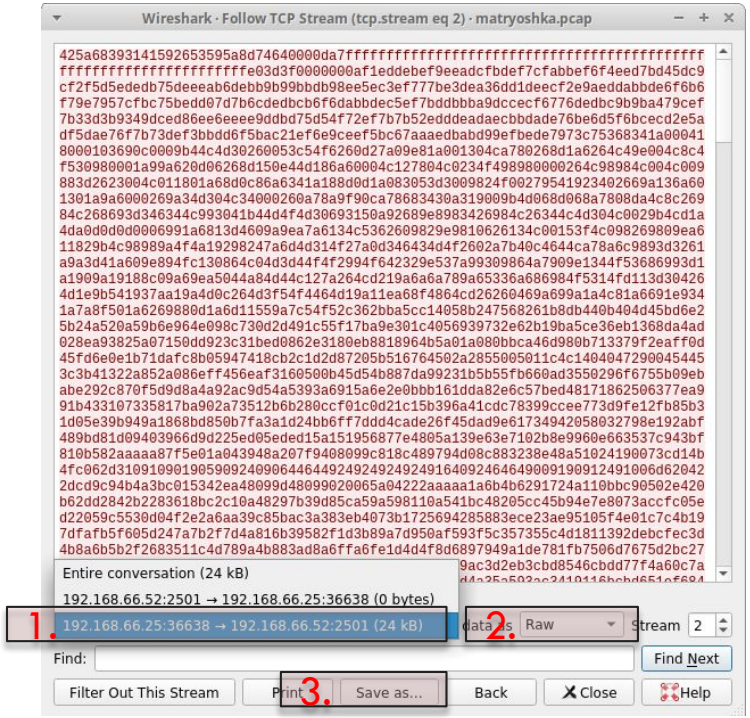




Carve files from streams: example 2

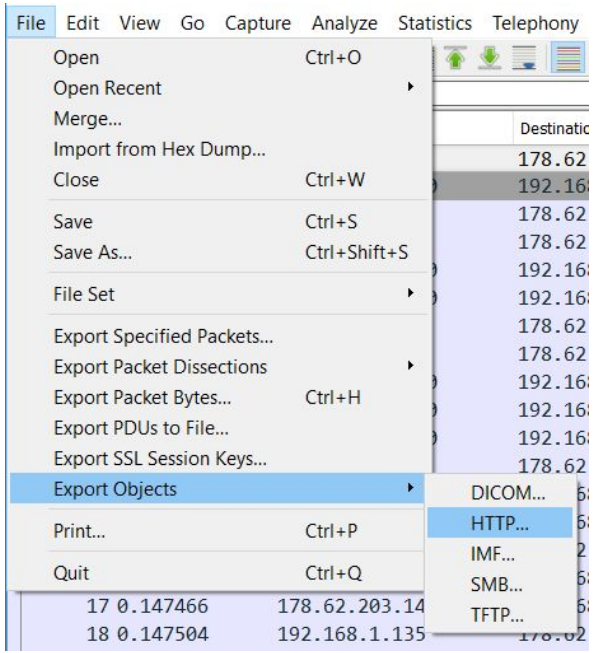
- Extract a file from a generic stream (unknown protocol)
  1. Switch between client to server or server to client conversation
  2. Show data as Raw
  3. Save as... (e.g., */tmp/bin1*)
  4. Use the linux *file* utility to determine the file type

```
/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



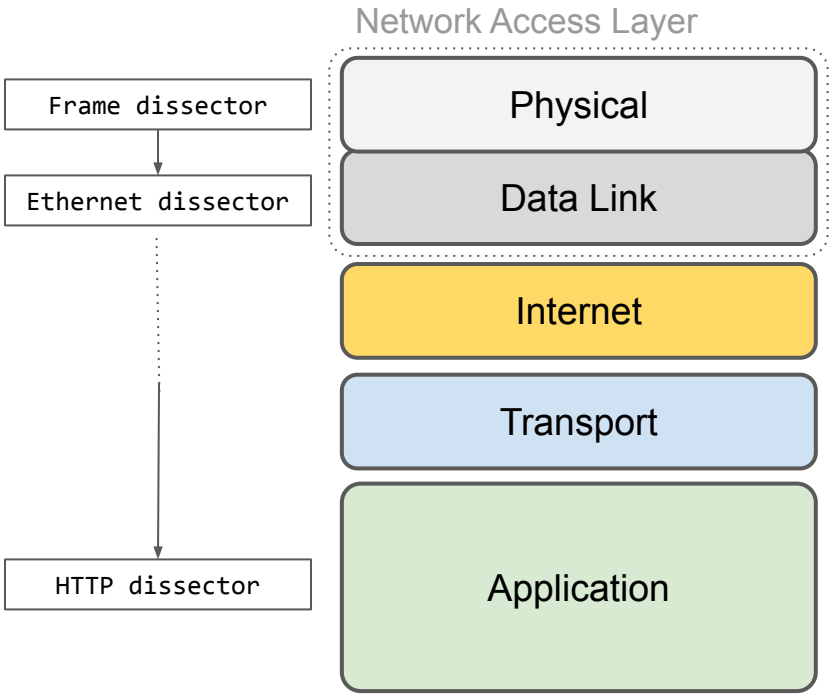
Ex. 5

- export and analyze evidence from HTTP traffic
- carve manually the PDF file
- carve the file transferred using FTP*

*Consider the FTP data port: https://mkichenamourty.wordpress.com/2017/04/05/how-to-calculate-the-ftp-data-port/

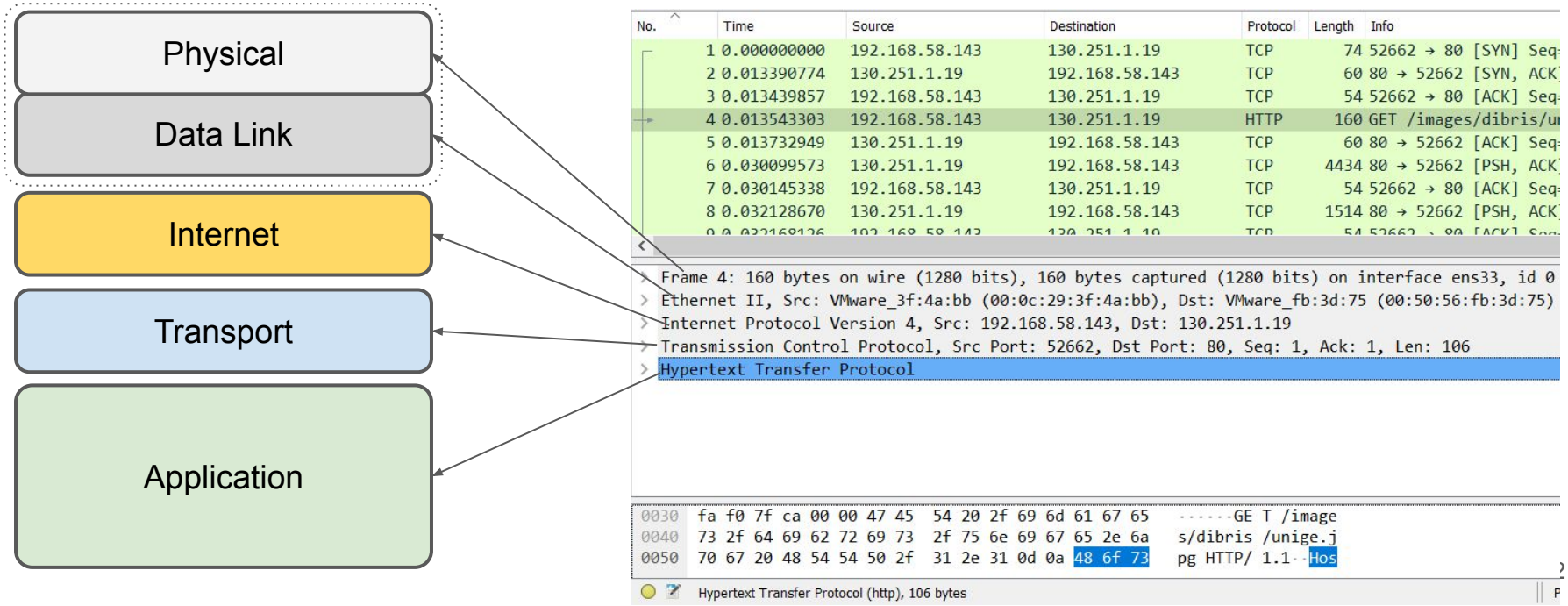
Dissectors

- 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)



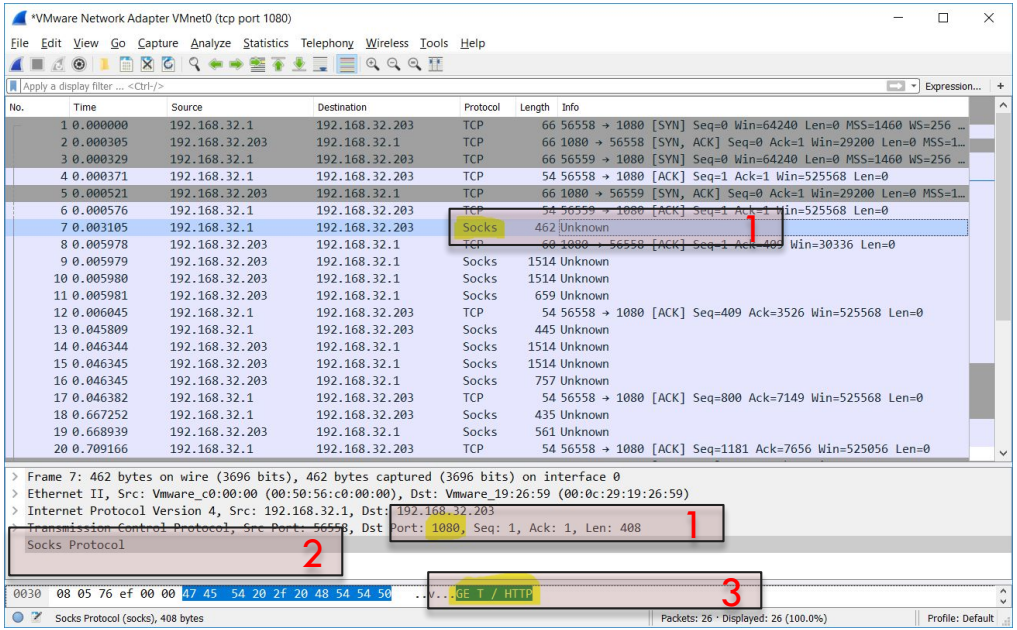
Packet details pane: dissectors

- 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