# Wireshark 101

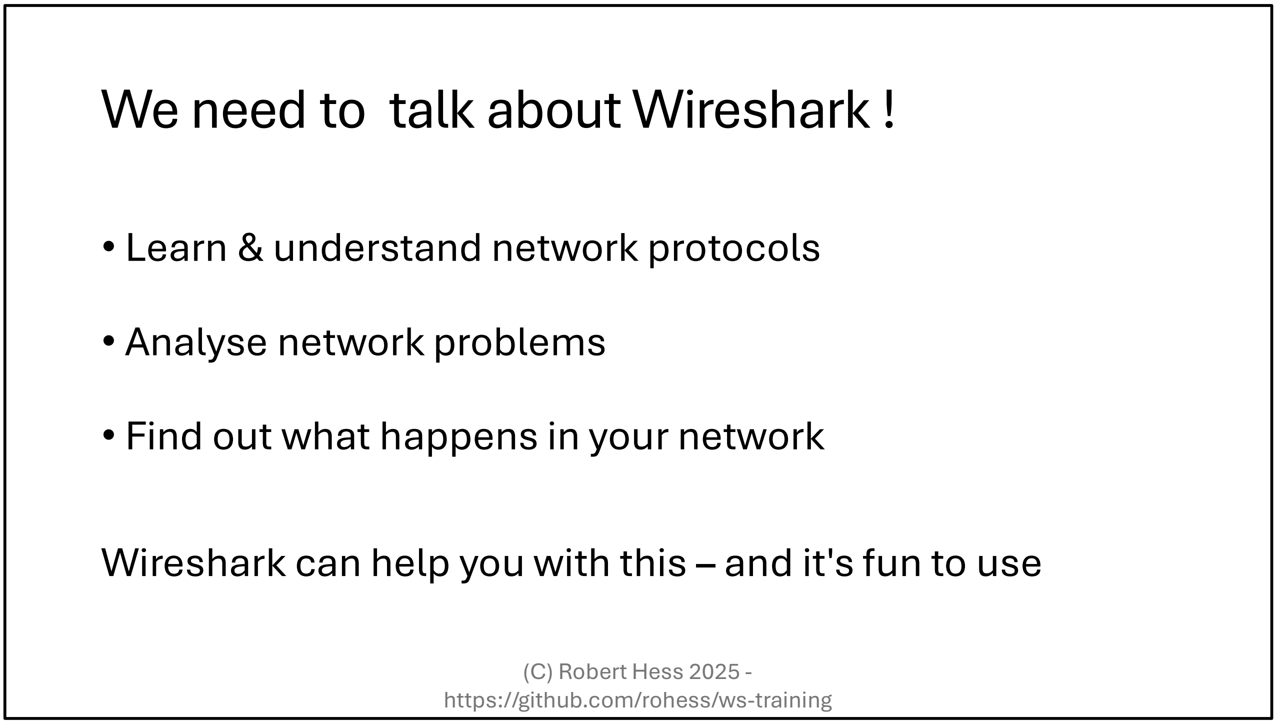
Its all about packets

Robert Hess – GoTo Dresden

robert.hess@goto.com

Material download & instructions:

**https://github.com/rohess/ws-training**

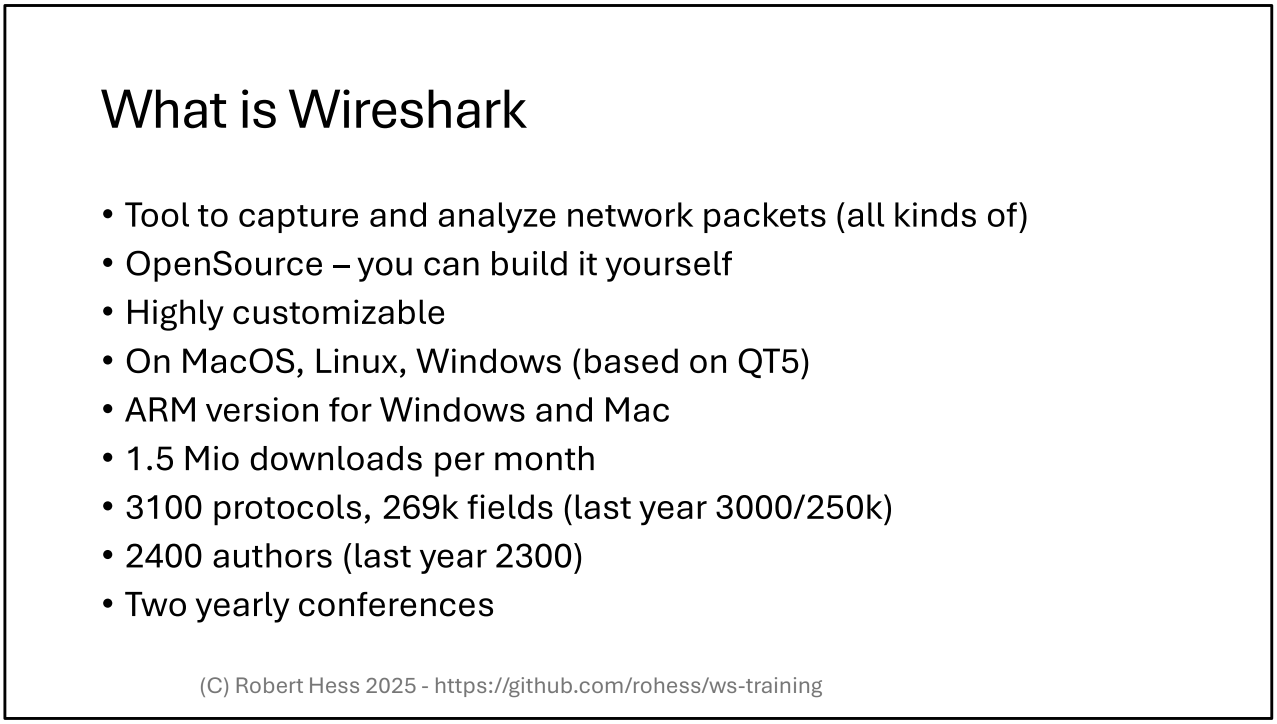


## We need to  talk about Wireshark !

- Learn & understand network protocols
- Analyse network problems
- Find out what happens in your network

Wireshark can help you with this – and it's fun to use

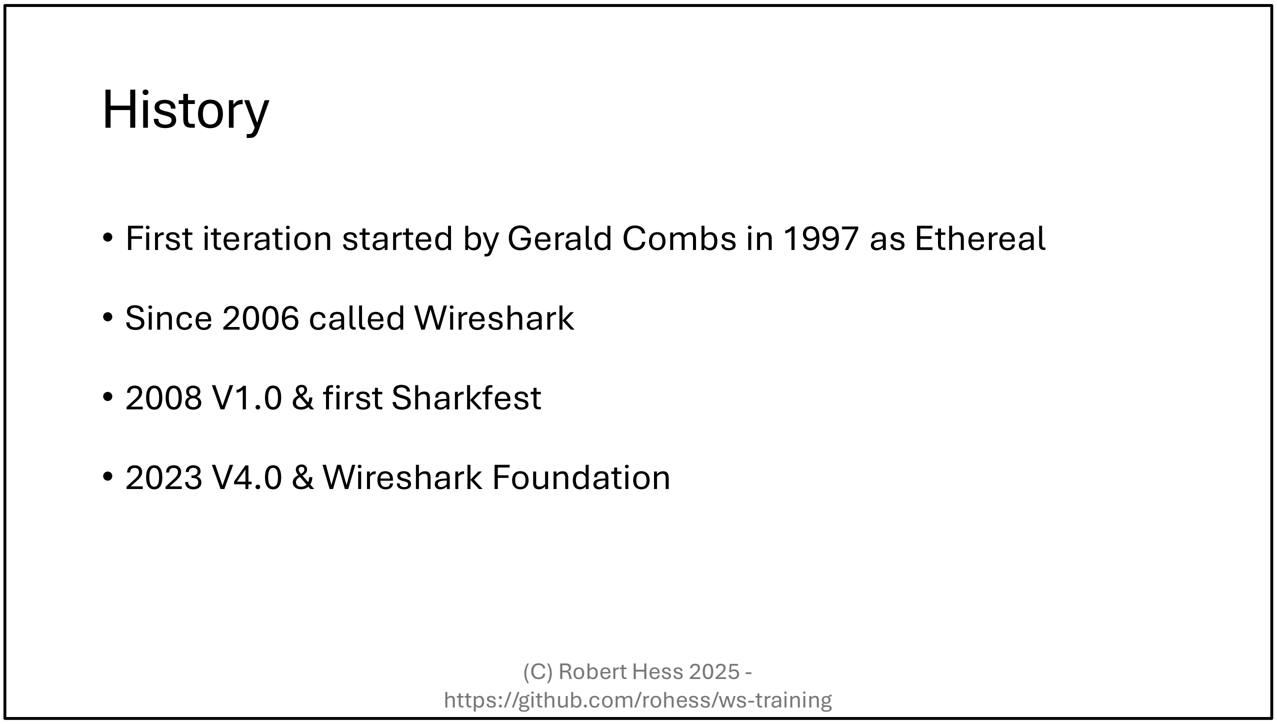
In depth insight into details – if you are like me you need to see things to understand



## What is Wireshark

- Tool to capture and analyze network packets (all kinds of)
- OpenSource – you can build it yourself
- Highly customizable
- On MacOS, Linux, Windows (based on QT5)
- ARM version for Windows and Mac
- 1.5 Mio downloads per month
- 3100 protocols, 269k fields (last year 3000/250k)
- 2400 authors (last year 2300)
- Two yearly conferences

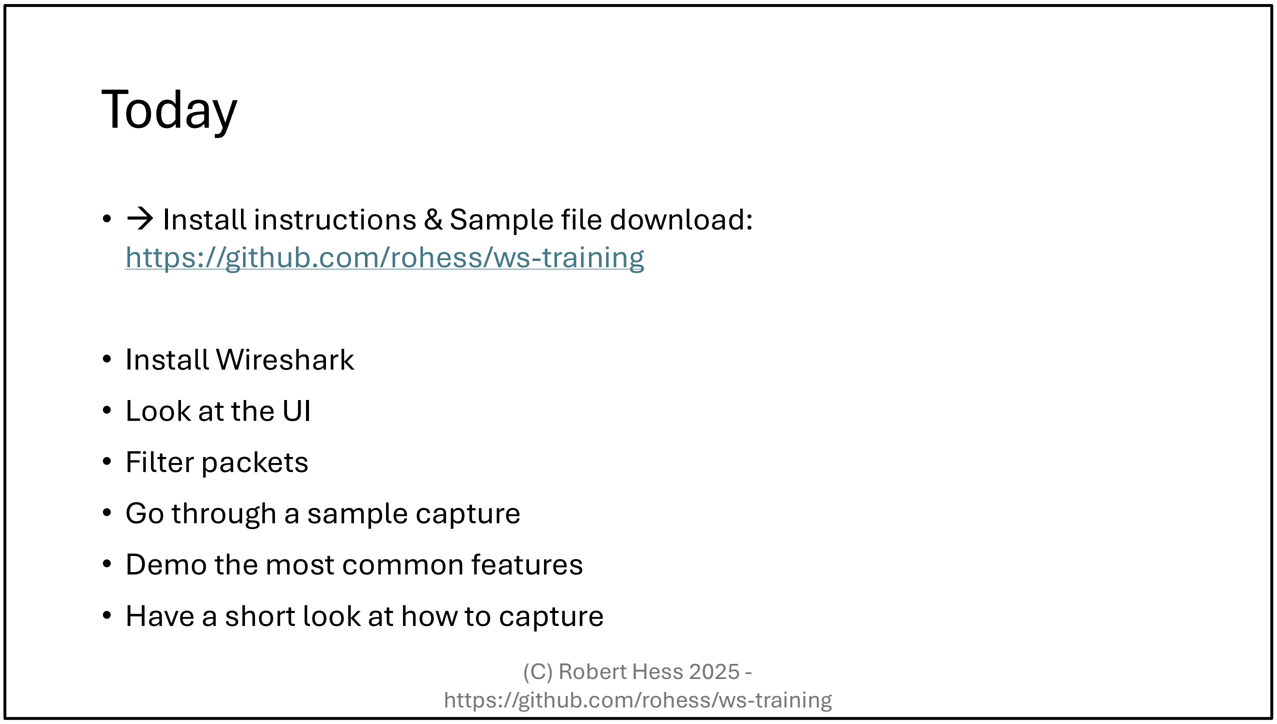
Also Wireless, USB etc. And meanwhile also Linux sys calls - StratoShark



# History

- First iteration started by Gerald Combs in 1997 as Ethereal
- Since 2006 called Wireshark
- 2008 V1.0 & first Sharkfest
- 2023 V4.0 & Wireshark Foundation

I attended 3 Sharkfests so far
WebRTC Masterclass



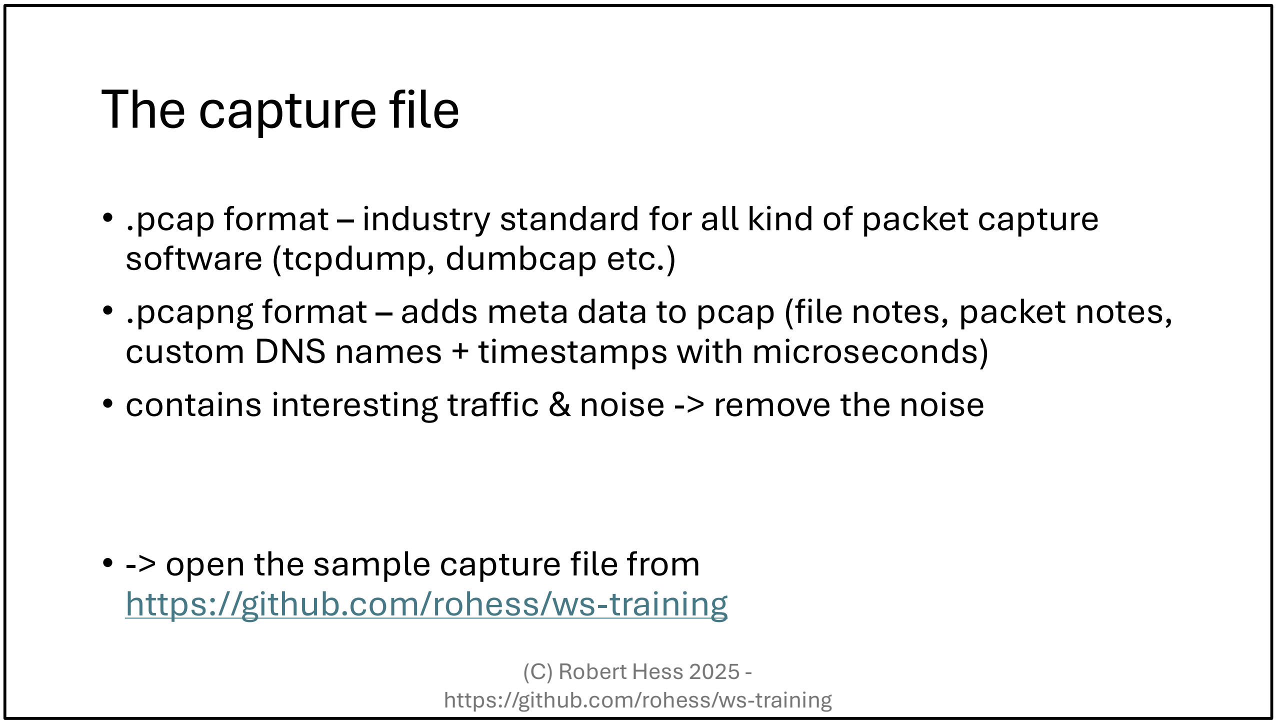
# Today

- → Install instructions & Sample file download: https://github.com/rohess/ws-training
- Install Wireshark
- Look at the UI
- Filter packets
- Go through a sample capture
- Demo the most common features
- Have a short look at how to capture

Who knows about

# Installation

- Recent version of Wireshark, at least 4.0 – current is 4.6.2 (stable) 4.7 (dev)
- On MacOS and Windows – just download the installer and install → https://www.wireshark.org/download.html
- Linux – check installation steps on Github link
  Repo versions tend to be outdated – built it from source ;-)



## The capture file

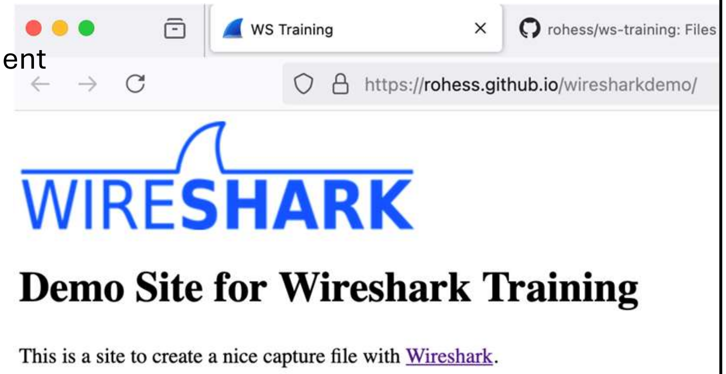
- .pcap format – industry standard for all kind of packet capture software (tcpdump, dumbcap etc.)
- .pcapng format – adds meta data to pcap (file notes, packet notes, custom DNS names + timestamps with microseconds)
- contains interesting traffic & noise -> remove the noise

- -> open the sample capture file from https://github.com/rohess/ws-training

# What's in it:

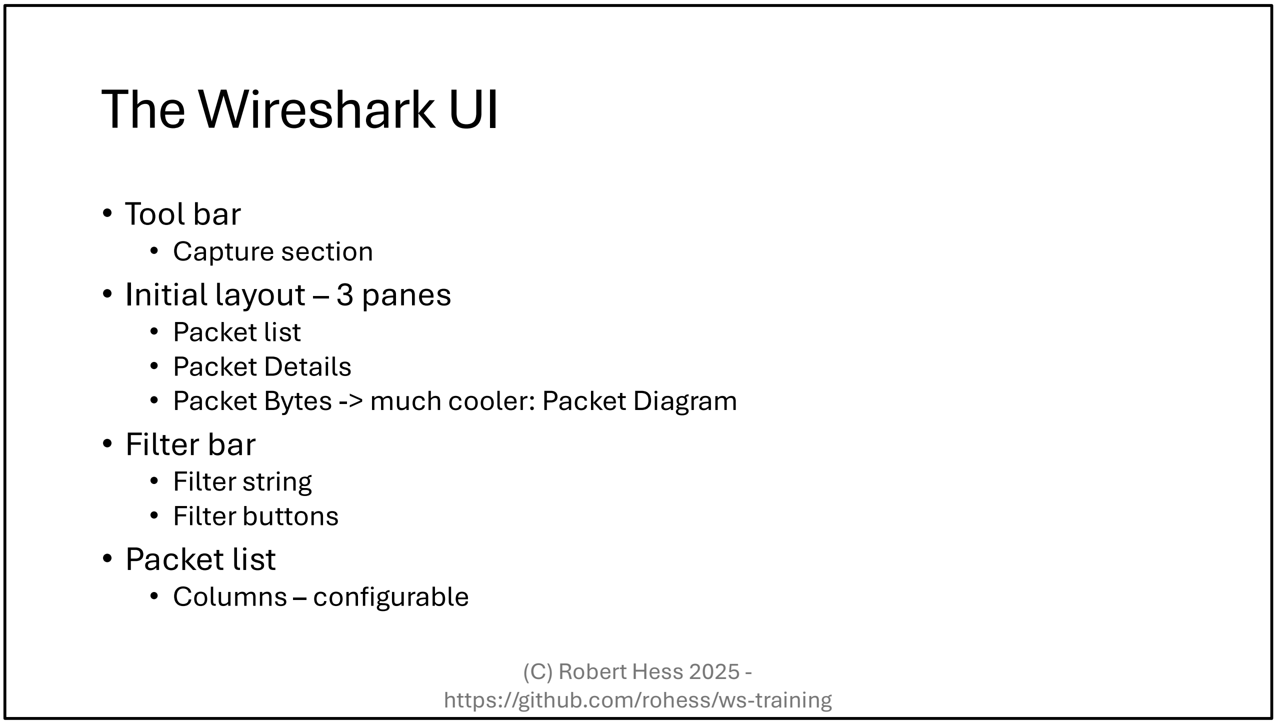
Download of a single webpage via Chrome

- DNS requests
- TCP Connection establishment
- TLS negotiation
- HTTP traffic downloading
  - HTML Code
  - Picture
  - favicon
- Connection tear down


WS Training
rohess/ws-training: Files
https://rohess.github.io/wiresharkdemo/
WIRESHARK

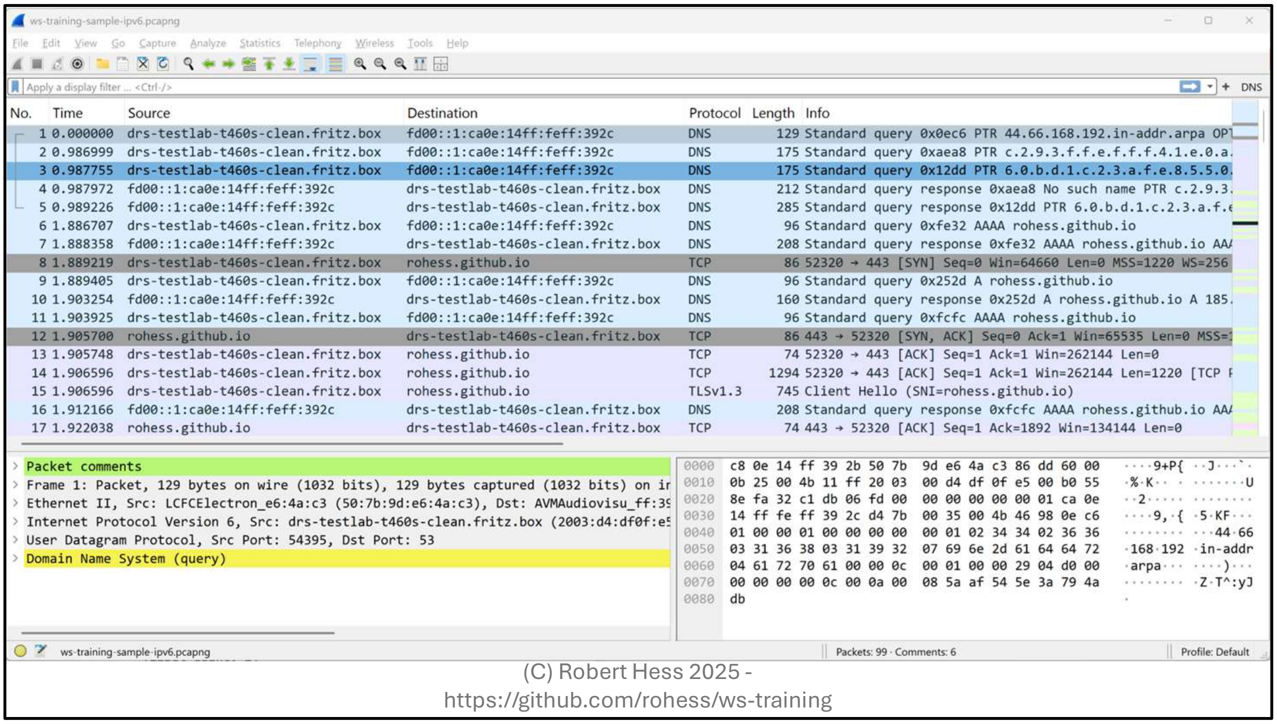
Demo Site for Wireshark Training

This is a site to create a nice capture file with Wireshark.



# The Wireshark UI

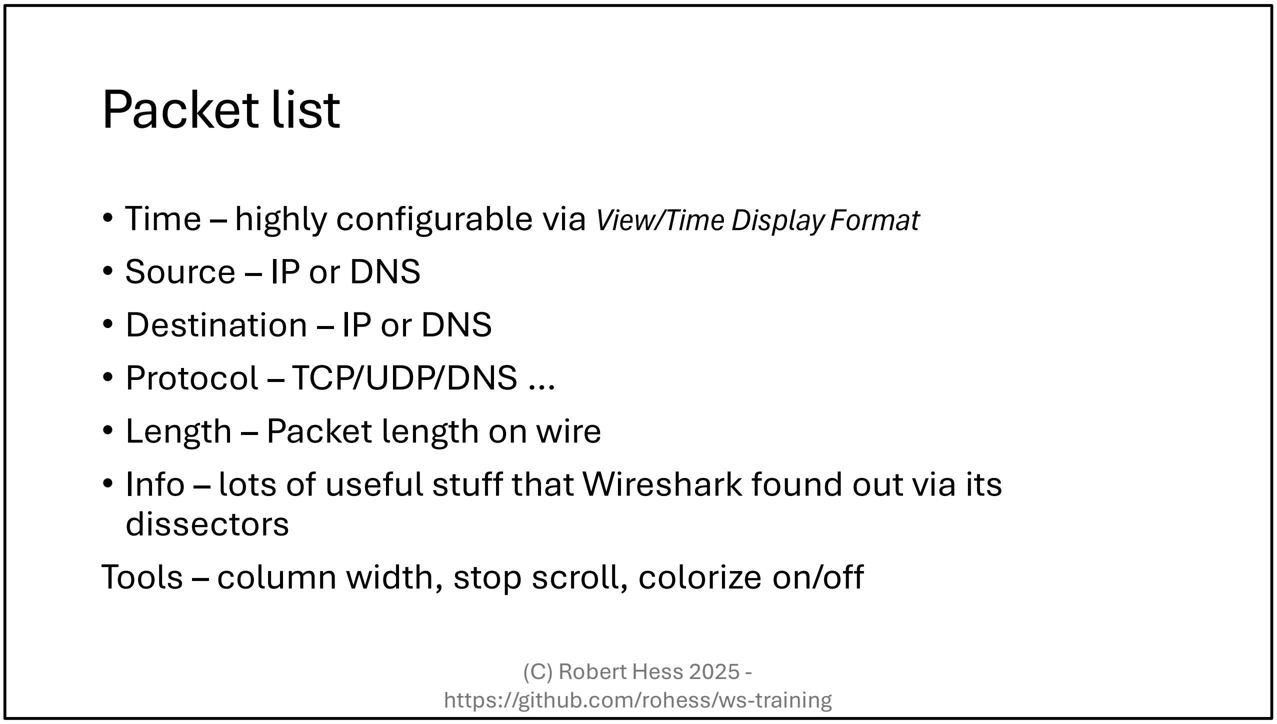
- Tool bar
  - Capture section
- Initial layout – 3 panes
  - Packet list
  - Packet Details
  - Packet Bytes -> much cooler: Packet Diagram
- Filter bar
  - Filter string
  - Filter buttons
- Packet list
  - Columns – configurable


ws-training-sample-ipv6.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
DNS

```
No. Time       Source                          Destination                     Protocol Length Info
  1 0.000000   drs-testlab-t460s-clean.fritz.box fd00::1:ca0e:14ff:feff:392c   DNS       129 Standard query 0x0ec6 PTR 44.66.168.192.in-addr.arpa OPT
  2 0.986999   drs-testlab-t460s-clean.fritz.box fd00::1:ca0e:14ff:feff:392c   DNS       175 Standard query 0xaea8 PTR c.2.9.3.f.f.e.f.f.f.4.1.e.0.a.
  3 0.987755   drs-testlab-t460s-clean.fritz.box fd00::1:ca0e:14ff:feff:392c   DNS       175 Standard query 0x12dd PTR 6.0.b.d.1.c.2.3.a.f.e.8.5.5.0.
  4 0.987972   fd00::1:ca0e:14ff:feff:392c   drs-testlab-t460s-clean.fritz.box DNS       212 Standard query response 0xaea8 No such name PTR c.2.9.3.
  5 0.989226   fd00::1:ca0e:14ff:feff:392c   drs-testlab-t460s-clean.fritz.box DNS       285 Standard query response 0x12dd PTR 6.0.b.d.1.c.2.3.a.f.e
  6 1.886707   drs-testlab-t460s-clean.fritz.box fd00::1:ca0e:14ff:feff:392c   DNS        96 Standard query 0xfe32 AAAA rohess.github.io
  7 1.888358   fd00::1:ca0e:14ff:feff:392c   drs-testlab-t460s-clean.fritz.box DNS       208 Standard query response 0xfe32 AAAA rohess.github.io AAA
  8 1.889219   drs-testlab-t460s-clean.fritz.box rohess.github.io              TCP        86 52320 → 443 [SYN] Seq=0 Win=64660 Len=0 MSS=1220 WS=256
  9 1.889405   drs-testlab-t460s-clean.fritz.box fd00::1:ca0e:14ff:feff:392c   DNS        96 Standard query 0x252d A rohess.github.io
 10 1.903254   fd00::1:ca0e:14ff:feff:392c   drs-testlab-t460s-clean.fritz.box DNS       160 Standard query response 0x252d A rohess.github.io A 185.
 11 1.903925   drs-testlab-t460s-clean.fritz.box fd00::1:ca0e:14ff:feff:392c   DNS        96 Standard query 0xfcfc AAAA rohess.github.io
 12 1.905700   rohess.github.io              drs-testlab-t460s-clean.fritz.box TCP        86 443 → 52320 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1
 13 1.905748   drs-testlab-t460s-clean.fritz.box rohess.github.io              TCP        74 52320 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
 14 1.906596   drs-testlab-t460s-clean.fritz.box rohess.github.io              TCP      1294 52320 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=1220 [TCP P
 15 1.906596   drs-testlab-t460s-clean.fritz.box rohess.github.io              TLSv1.3   745 Client Hello (SNI=rohess.github.io)
 16 1.912166   fd00::1:ca0e:14ff:feff:392c   drs-testlab-t460s-clean.fritz.box DNS       208 Standard query response 0xfcfc AAAA rohess.github.io AAA
 17 1.922038   rohess.github.io              drs-testlab-t460s-clean.fritz.box TCP        74 443 → 52320 [ACK] Seq=1 Ack=1892 Win=134144 Len=0
```

```
> Packet comments
> Frame 1: Packet, 129 bytes on wire (1032 bits), 129 bytes captured (1032 bits) on ir
> Ethernet II, Src: LCFCElectron_e6:4a:c3 (50:7b:9d:e6:4a:c3), Dst: AVMAudiovisu_ff:39
> Internet Protocol Version 6, Src: drs-testlab-t460s-clean.fritz.box (2003:d4:df0f:e5
> User Datagram Protocol, Src Port: 54395, Dst Port: 53
> Domain Name System (query)
```

```
0000  c8 0e 14 ff 39 2b 50 7b  9d e6 4a c3 86 dd 60 00   ····9+P{ ··J···`·
0010  0b 25 00 4b 11 ff 20 03  00 d4 df 0f e5 00 b0 55   ·%·K·· · ·······U
0020  8e fa 32 c1 db 06 fd 00  00 00 00 00 00 01 ca 0e   ··2····· ········
0030  14 ff fe ff 39 2c d4 7b  00 35 00 4b 46 98 0e c6   ····9,·{ ·5·KF···
0040  01 00 00 01 00 00 00 00  00 01 02 34 34 02 36 36   ········ ···44·66
0050  03 31 36 38 03 31 39 32  07 69 6e 2d 61 64 64 72   ·168·192 ·in-addr
0060  04 61 72 70 61 00 00 0c  00 01 00 00 29 04 d0 00   ·arpa··· ····)···
0070  00 00 00 00 0c 00 0a 00  08 5a af 54 5e 3a 79 4a   ········ ·Z·T^:yJ
0080  db                                                 ·
```

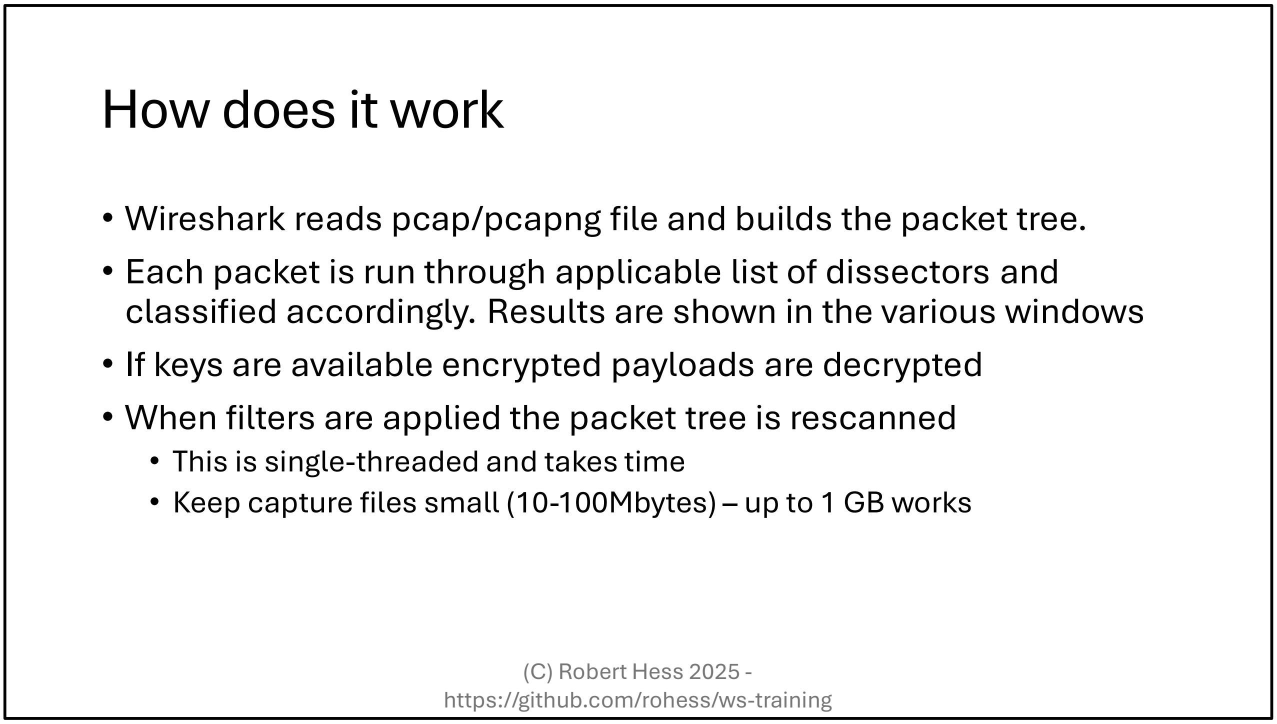


## Packet list

- Time – highly configurable via *View/Time Display Format*
- Source – IP or DNS
- Destination – IP or DNS
- Protocol – TCP/UDP/DNS …
- Length – Packet length on wire
- Info – lots of useful stuff that Wireshark found out via its dissectors

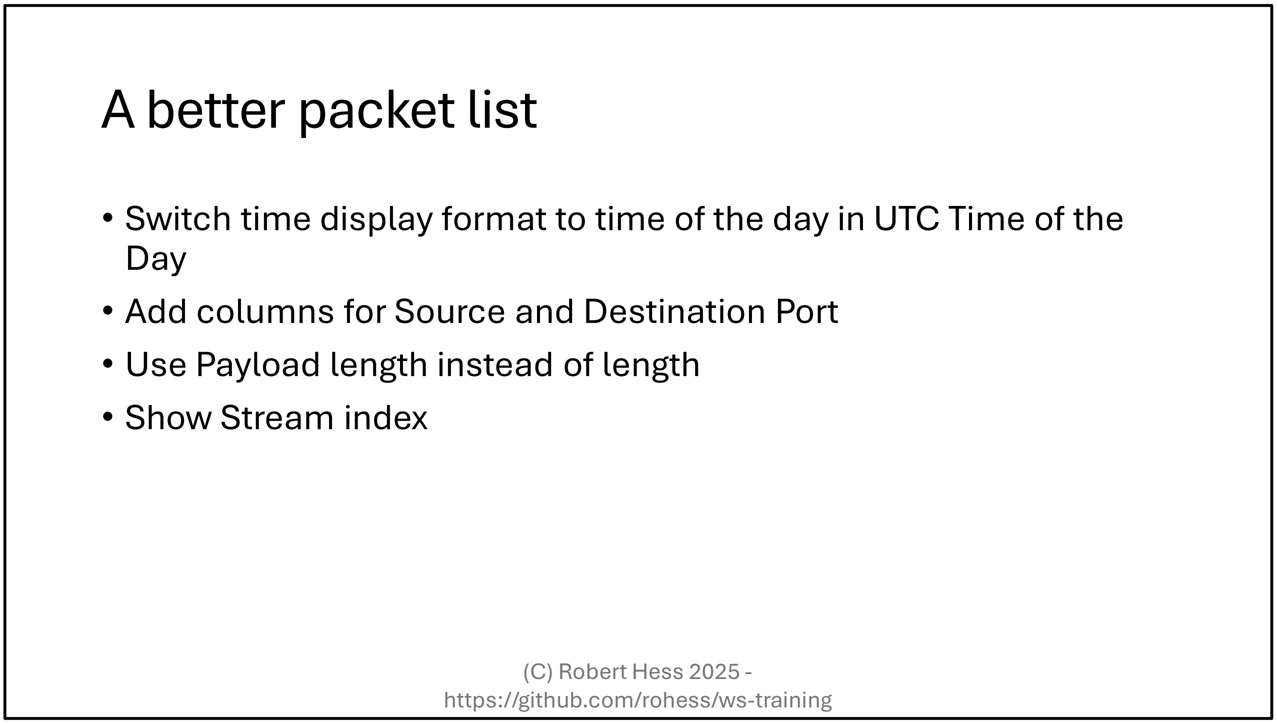
Tools – column width, stop scroll, colorize on/off

Switch on



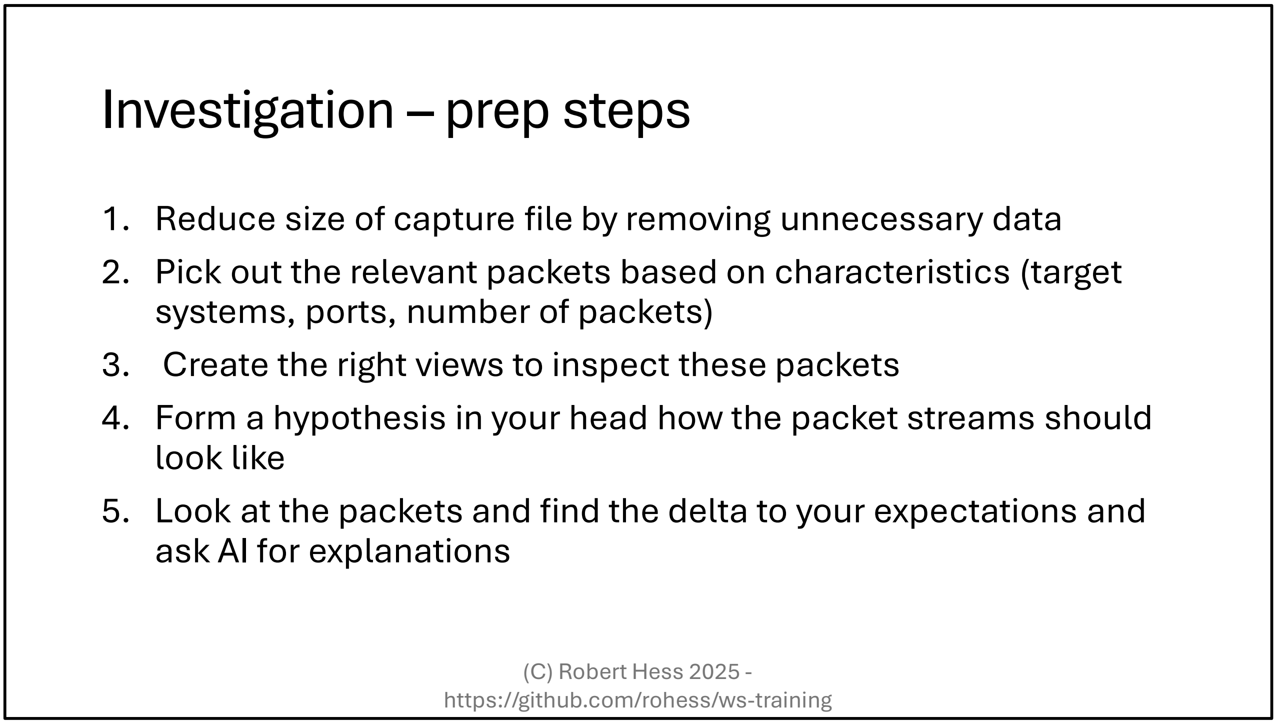
## How does it work

- Wireshark reads pcap/pcapng file and builds the packet tree.
- Each packet is run through applicable list of dissectors and classified accordingly. Results are shown in the various windows
- If keys are available encrypted payloads are decrypted
- When filters are applied the packet tree is rescanned
  - This is single-threaded and takes time
  - Keep capture files small (10-100Mbytes) – up to 1 GB works



# A better packet list

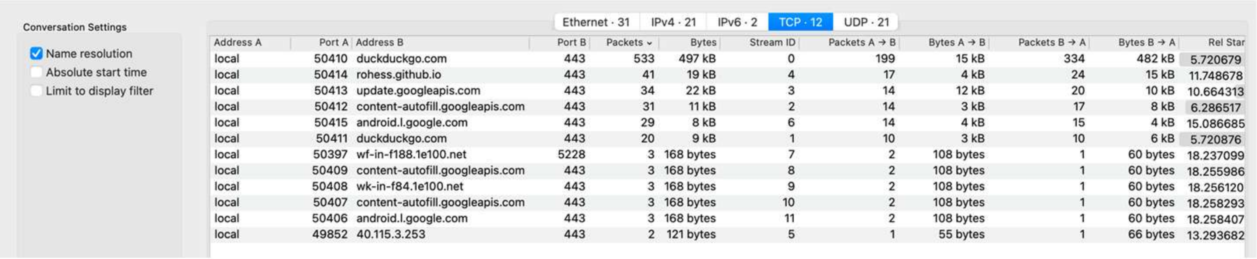
- Switch time display format to time of the day in UTC Time of the Day
- Add columns for Source and Destination Port
- Use Payload length instead of length
- Show Stream index



# Investigation – prep steps

1. Reduce size of capture file by removing unnecessary data
2. Pick out the relevant packets based on characteristics (target systems, ports, number of packets)
3. Create the right views to inspect these packets
4. Form a hypothesis in your head how the packet streams should look like
5. Look at the packets and find the delta to your expectations and ask AI for explanations

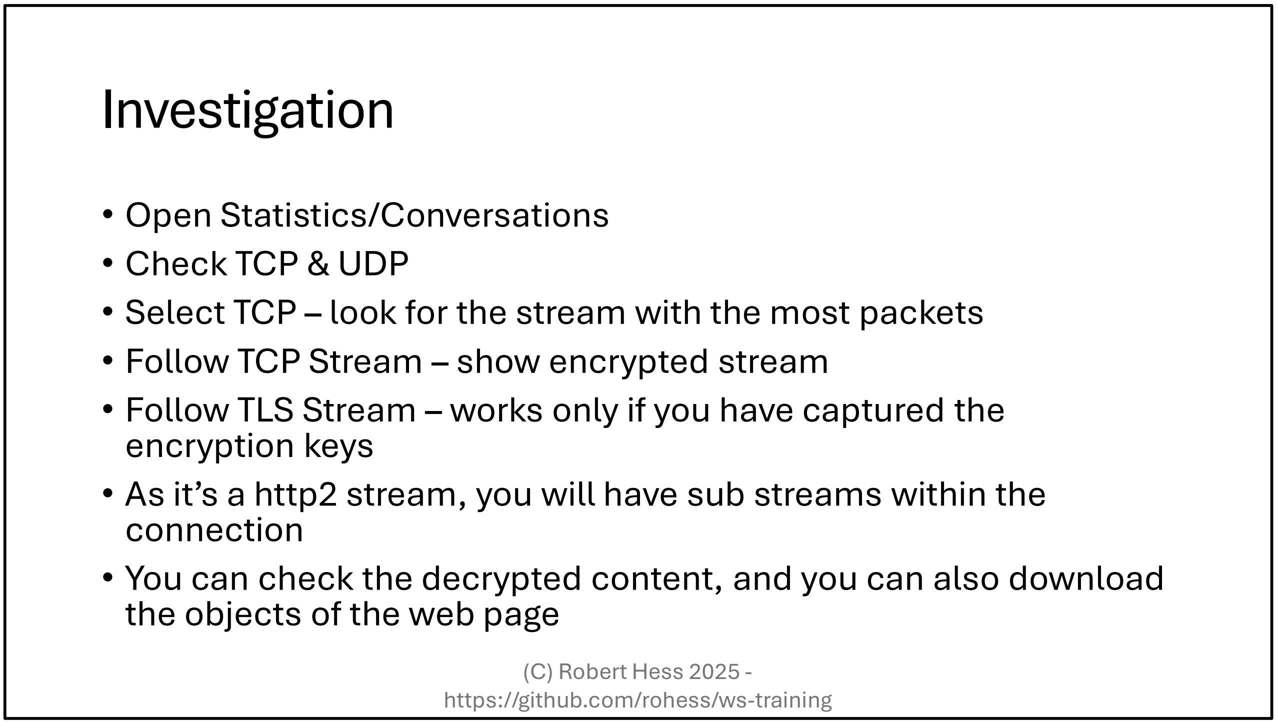
# Statistics/Conversations



Conversation Settings

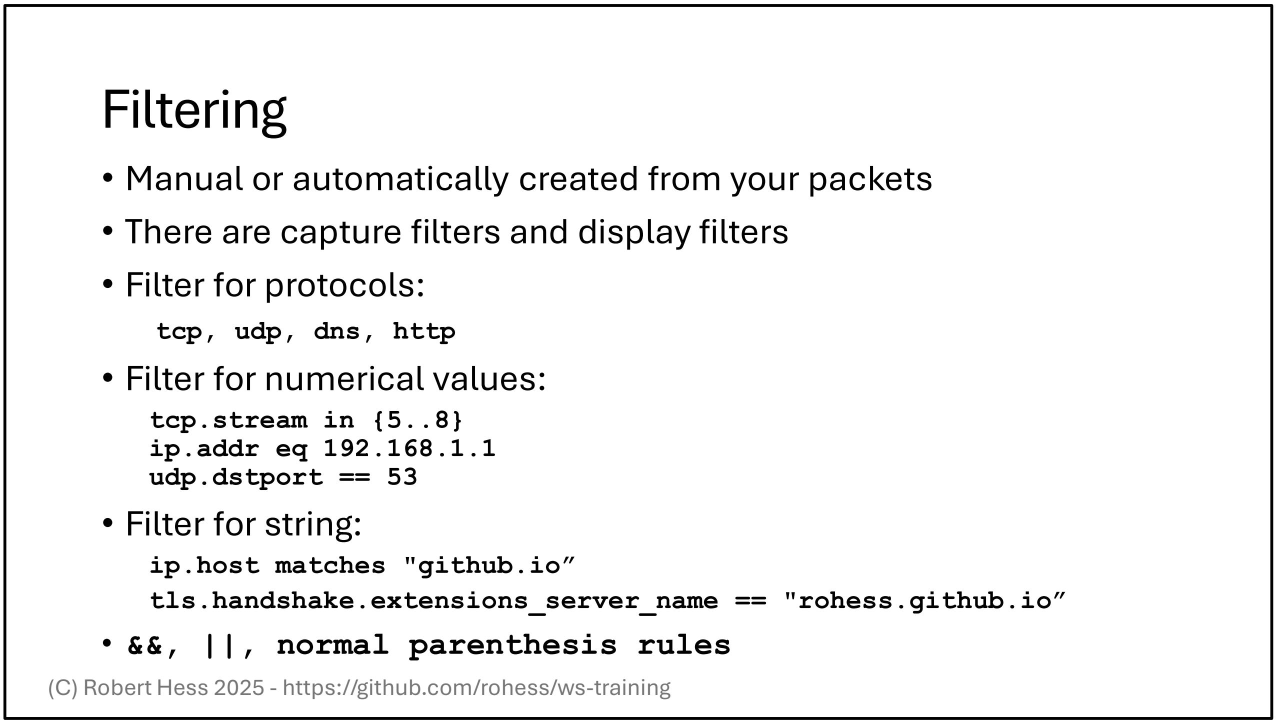
- ☑ Name resolution
- □ Absolute start time
- □ Limit to display filter

| Address A | Port A | Address B | Port B | Packets ⌄ | Bytes | Stream ID | Packets A → B | Bytes A → B | Packets B → A | Bytes B → A | Rel Star |
|---|---|---|---|---|---|---|---|---|---|---|---|
| local | 50410 | duckduckgo.com | 443 | 533 | 497 kB | 0 | 199 | 15 kB | 334 | 482 kB | 5.720679 |
| local | 50414 | rohess.github.io | 443 | 41 | 19 kB | 4 | 17 | 4 kB | 24 | 15 kB | 11.748678 |
| local | 50413 | update.googleapis.com | 443 | 34 | 22 kB | 3 | 14 | 12 kB | 20 | 10 kB | 10.664313 |
| local | 50412 | content-autofill.googleapis.com | 443 | 31 | 11 kB | 2 | 14 | 3 kB | 17 | 8 kB | 6.286517 |
| local | 50415 | android.l.google.com | 443 | 29 | 8 kB | 6 | 14 | 4 kB | 15 | 4 kB | 15.086685 |
| local | 50411 | duckduckgo.com | 443 | 20 | 9 kB | 1 | 10 | 3 kB | 10 | 6 kB | 5.720876 |
| local | 50397 | wf-in-f188.1e100.net | 5228 | 3 | 168 bytes | 7 | 2 | 108 bytes | 1 | 60 bytes | 18.237099 |
| local | 50409 | content-autofill.googleapis.com | 443 | 3 | 168 bytes | 8 | 2 | 108 bytes | 1 | 60 bytes | 18.255986 |
| local | 50408 | wk-in-f84.1e100.net | 443 | 3 | 168 bytes | 9 | 2 | 108 bytes | 1 | 60 bytes | 18.256120 |
| local | 50407 | content-autofill.googleapis.com | 443 | 3 | 168 bytes | 10 | 2 | 108 bytes | 1 | 60 bytes | 18.258293 |
| local | 50406 | android.l.google.com | 443 | 3 | 168 bytes | 11 | 2 | 108 bytes | 1 | 60 bytes | 18.258407 |
| local | 49852 | 40.115.3.253 | 443 | 2 | 121 bytes | 5 | 1 | 55 bytes | 1 | 66 bytes | 13.293682 |



## Investigation

- Open Statistics/Conversations
- Check TCP & UDP
- Select TCP – look for the stream with the most packets
- Follow TCP Stream – show encrypted stream
- Follow TLS Stream – works only if you have captured the encryption keys
- As it’s a http2 stream, you will have sub streams within the connection
- You can check the decrypted content, and you can also download the objects of the web page



## Filtering

- Manual or automatically created from your packets
- There are capture filters and display filters
- Filter for protocols:

```
tcp, udp, dns, http
```

- Filter for numerical values:

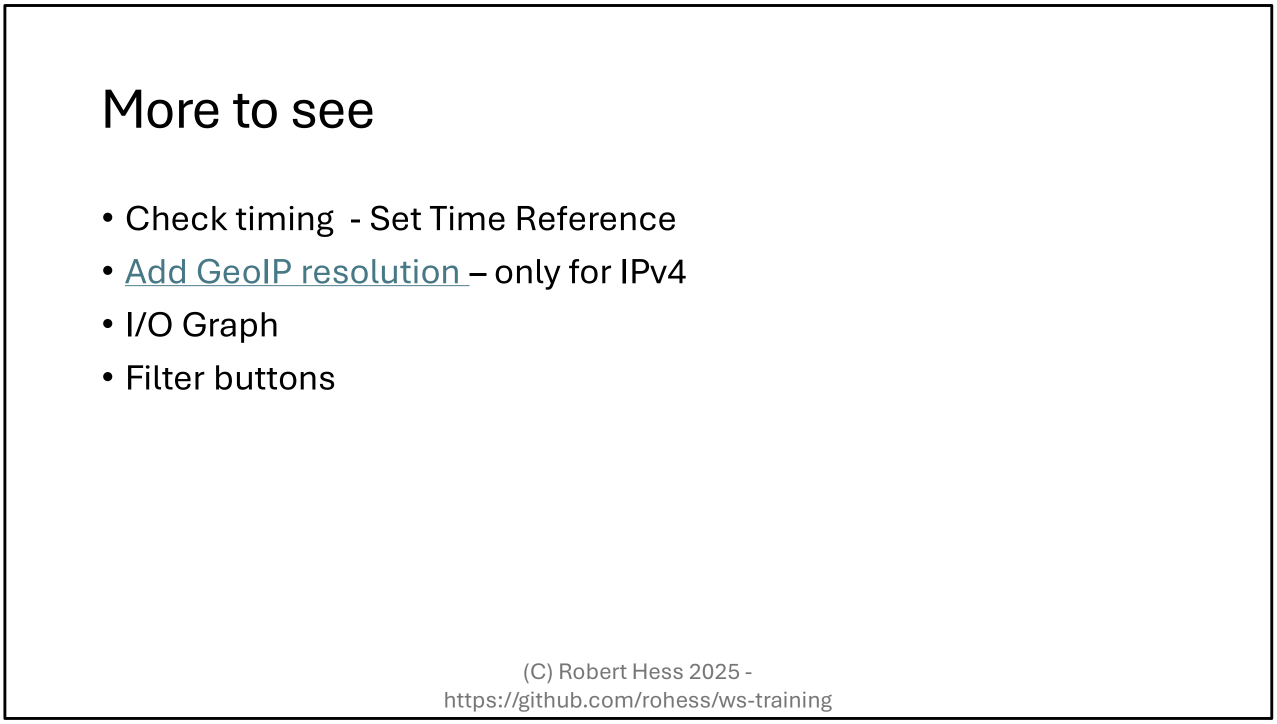
```
tcp.stream in {5..8}
ip.addr eq 192.168.1.1
udp.dstport == 53
```

- Filter for string:

```
ip.host matches "github.io”
tls.handshake.extensions_server_name == "rohess.github.io”
```

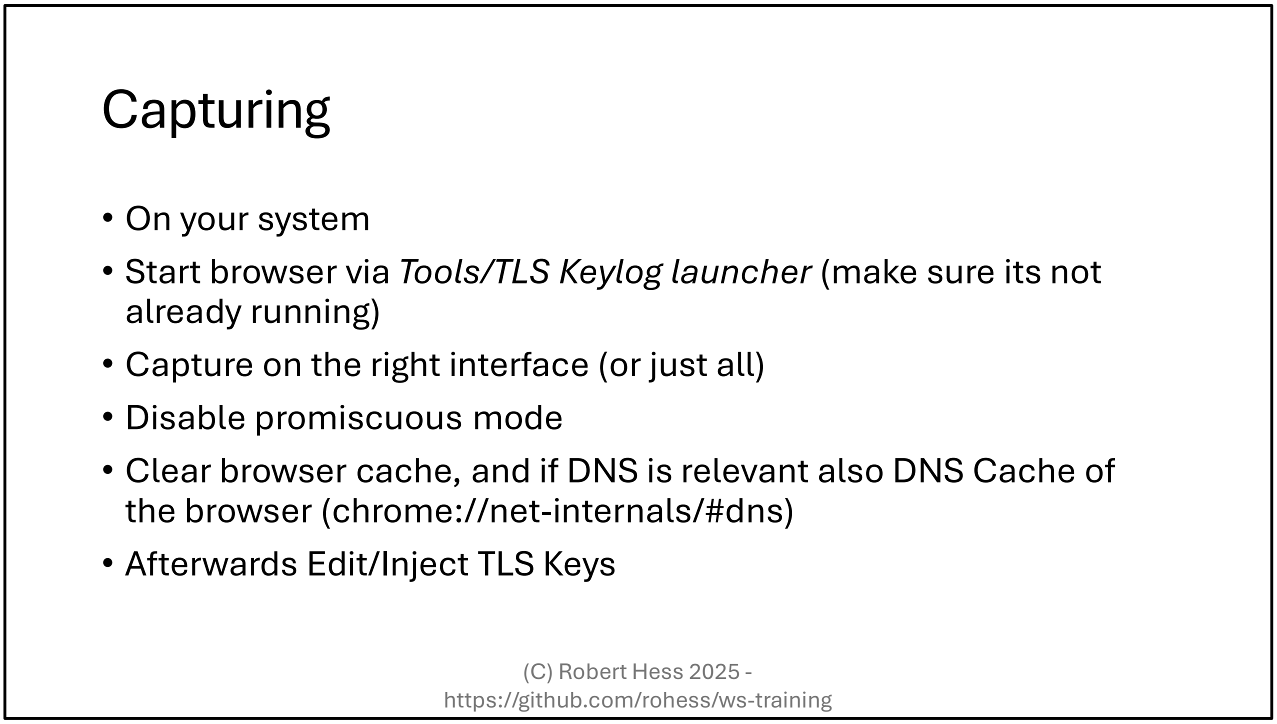
- **`&&, ||, normal parenthesis rules`**

The difference is that capture filters can run at capture time without dissecting the packet, and hence are fast. Display filters work on the dissected packet tree, and offer more complex filtering.



## More to see

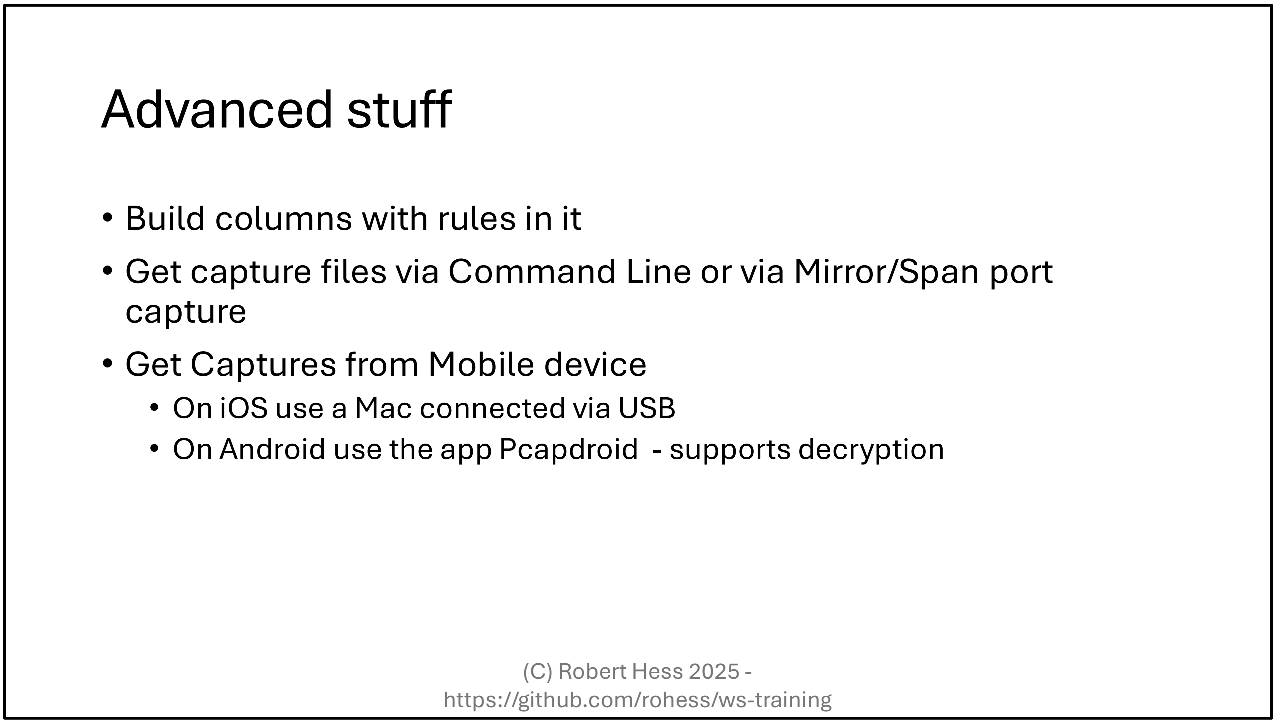
- Check timing  - Set Time Reference
- Add GeoIP resolution – only for IPv4
- I/O Graph
- Filter buttons



## Capturing

- On your system
- Start browser via *Tools/TLS Keylog launcher* (make sure its not already running)
- Capture on the right interface (or just all)
- Disable promiscuous mode
- Clear browser cache, and if DNS is relevant also DNS Cache of the browser (chrome://net-internals/#dns)
- Afterwards Edit/Inject TLS Keys

Promiscuous mode gives you everything that strands on your interface, namely lots of broadcasts
This is _most_ of the time irrelevant – unless you want to see mdns, IPv6 neighbour solicitation & router advertisements



## Advanced stuff

- Build columns with rules in it
- Get capture files via Command Line or via Mirror/Span port capture
- Get Captures from Mobile device
  - On iOS use a Mac connected via USB
  - On Android use the app Pcapdroid  - supports decryption

GeoIP uses Maxminds DB -> https://wiki.wireshark.org/HowToUseGeoIP
If you need different path on different systems you can use a file "maxmind_db_paths" in the top level of the profile directory – this can contain sveral paths. Make sure to use / (slash) instead of backslash, also on Windows

Mirror/Span ports may re-order your packets, thus confusing wireshark to show dup acks and spurious retransmissions.

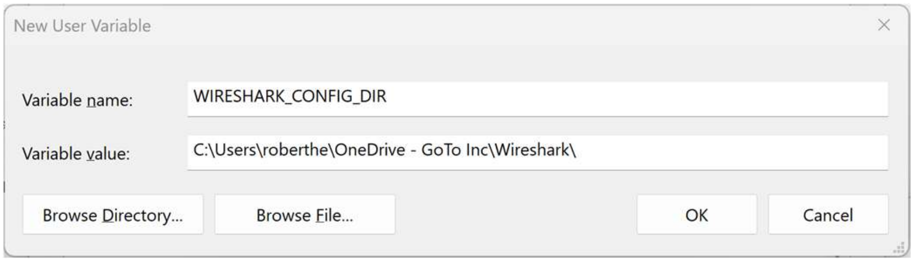
Decryption on Android works only for some apps, (e.g. Chrome) but not for others as they will only use the system cert store and not the user store -> hence no machine in the middle -> no TLS keys

# Profiles

Contains: Display Filters, Columns, coloring Rules, Layout, Disabled Protocols, Decode as Rules, I/O Graphs, Recent files

Load from a shared drive to have the same config everywhere

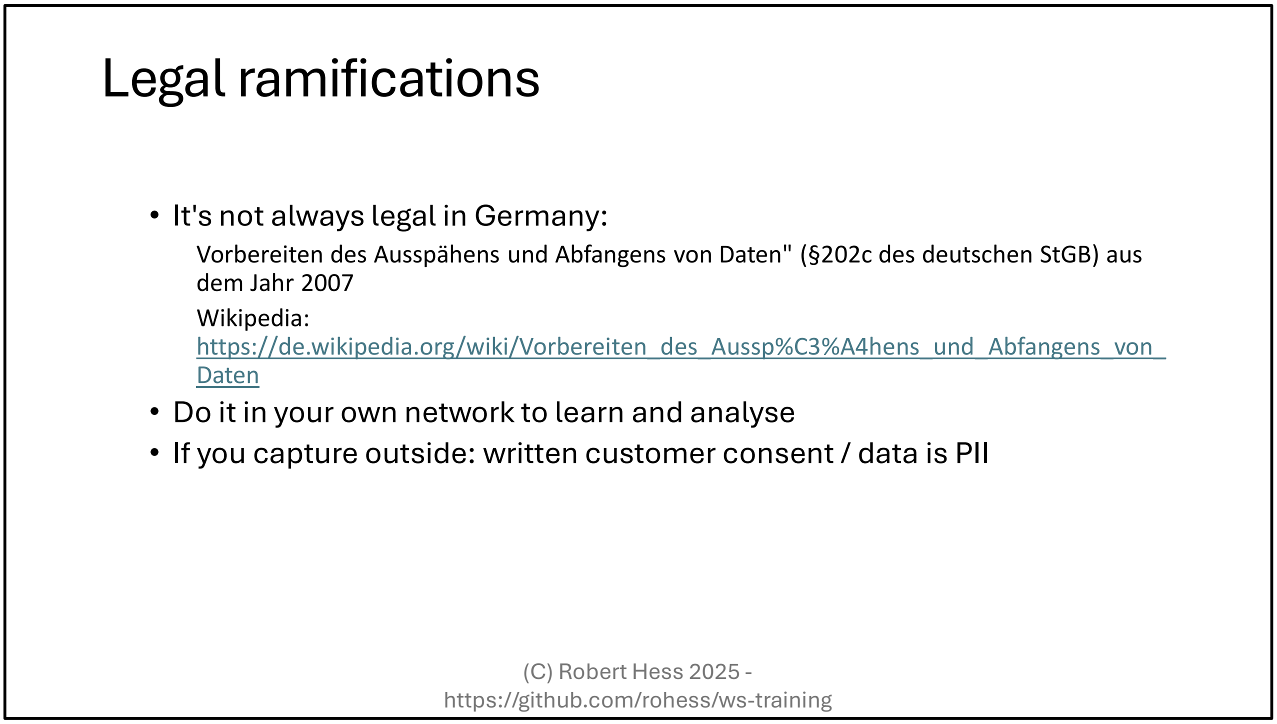
Profile directory is set via env variable



Variable name: WIRESHARK_CONFIG_DIR

Variable value: C:\Users\roberthe\OneDrive - GoTo Inc\Wireshark\

There may be, of cause merge conflicts, in which case WS creates a new profile by adding the hostname

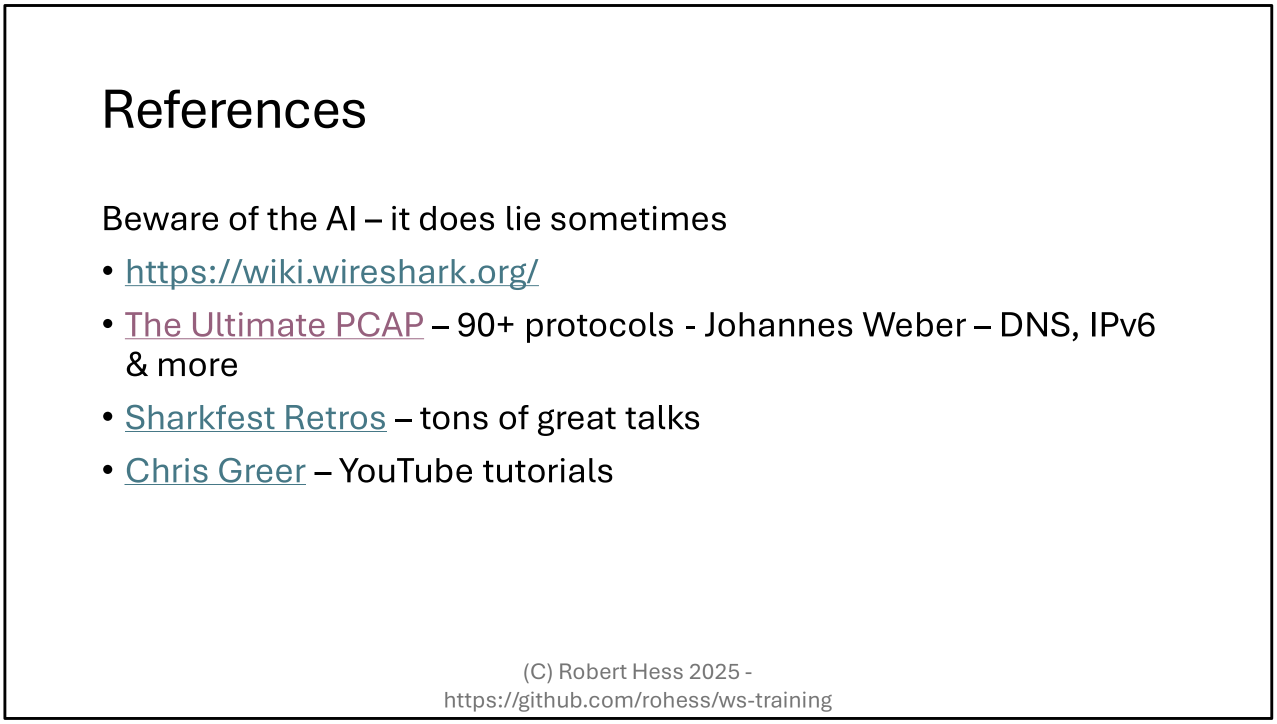


# Legal ramifications

- It's not always legal in Germany:

  Vorbereiten des Ausspähens und Abfangens von Daten" (§202c des deutschen StGB) aus dem Jahr 2007

  Wikipedia: https://de.wikipedia.org/wiki/Vorbereiten_des_Aussp%C3%A4hens_und_Abfangens_von_Daten
- Do it in your own network to learn and analyse
- If you capture outside: written customer consent / data is PII



## References

Beware of the AI – it does lie sometimes

- https://wiki.wireshark.org/
- The Ultimate PCAP – 90+ protocols - Johannes Weber – DNS, IPv6 & more
- Sharkfest Retros – tons of great talks
- Chris Greer – YouTube tutorials

# Have fun & thanks