Wireshark Filtering in Computer Networks

Federico Montori, PhD
University of Bologna

Who am I

Dr. Federico Montori

PostDoc research fellow in Internet of Things, Mobile Crowdsensing and IoT Data Analytics

- UniBo: https://www.unibo.it/sitoweb/federico.montori2/
- IoT Prism Lab: http://iot-prism-lab.nws.cs.unibo.it/team/federico-montori/
- email: federico.montori2@unibo.it

Goals

- Understanding by doing
- Look for specific network information
- See in practice things that you learned
- Give a kickoff to getting hands dirty

Outline

1. General introduction to packet sniffing
2. Tutorial on Wireshark
3. Capture filters and Display filters
4. Hands-on exercises

The slides are inspired to the WireShark course by Dr. Luca Bedogni

All material is a courtesy of WireShark Labs, J.F. Kurose, K.W. Ross
(https://gaia.cs.umass.edu/kurose_ross/wireshark.htm)

Pre-requisites

In order to participate actively in the hands-on tutorial you need to do a couple of steps:

- Download and install WireShark (https://www.wireshark.org/download.html)
  - You may want to do it in a Virtual Machine if you feel you want to go hardcore later…
  - Linux users can find it on the repo
- Download the pre-set WireShark traces at http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip

Packet Sniffers

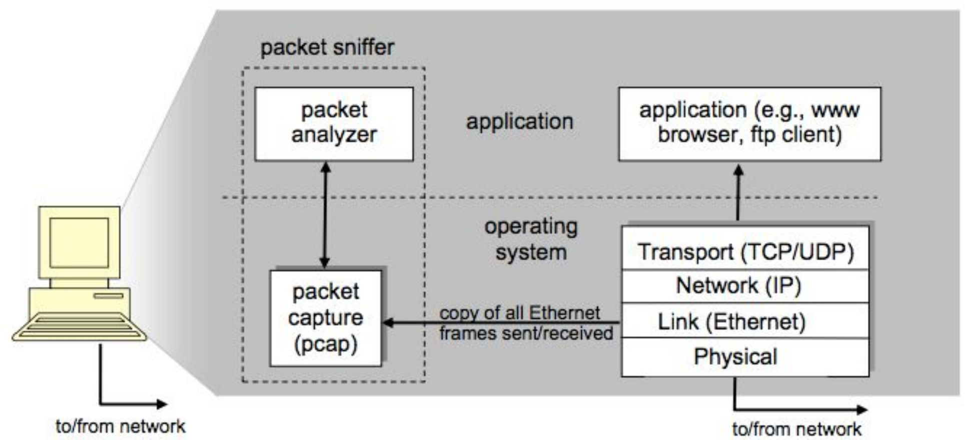
Packet sniffing is the operation of capturing data flowing through the network to look for information in network packets.

Frequently used by system administrator to troubleshoot network issues:

- Why traffic is slow
- Detect intrusions

Considered security tools:

- Because it gives all the tools to assess it



Packet Sniffers

It is a passive technique:

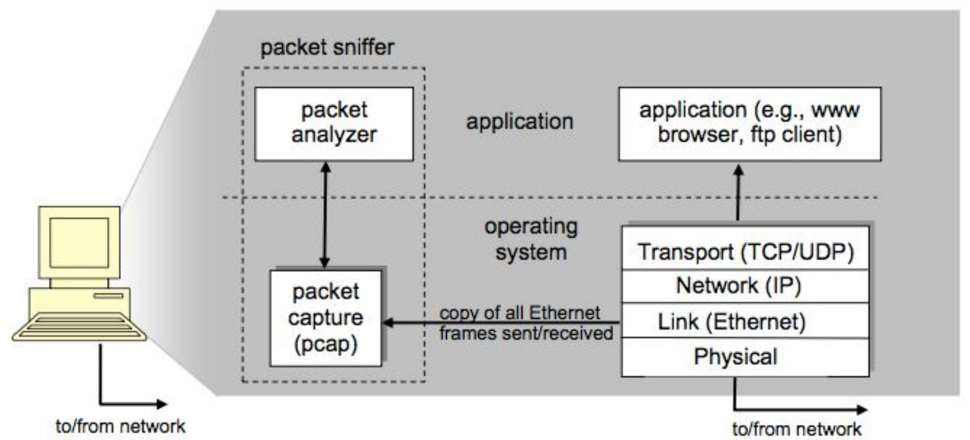
- One or more of your network interfaces to listen for everything (or a subset…)
- Packets are copied and displayed to the user
- The packet sniffer is just listening
- Actually, information does not change

The packet capture library (PCAP):

- Receives the packets with filters.

The packet analyzer:

- Shows the packet contents
- Decodes nested fields



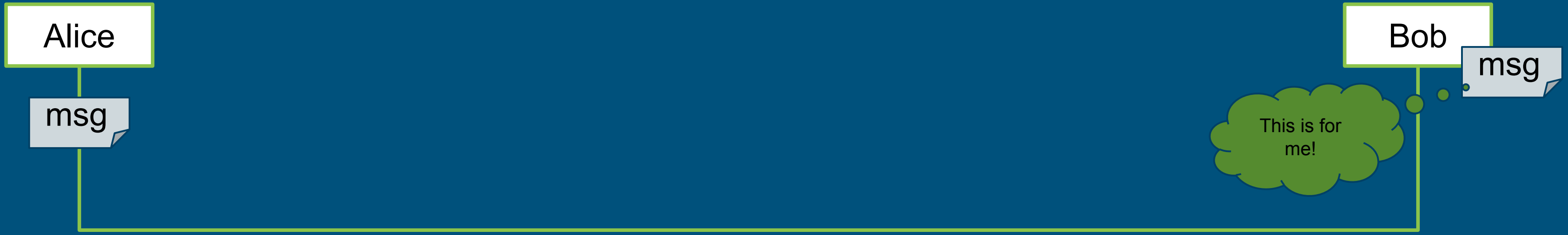
PROMISCUOUS MODE:

- Don’t throw it away if it isn’t for you

How do they work?

Intuitively, you may think that when systems communicate over the network their packets go directly to the destination...

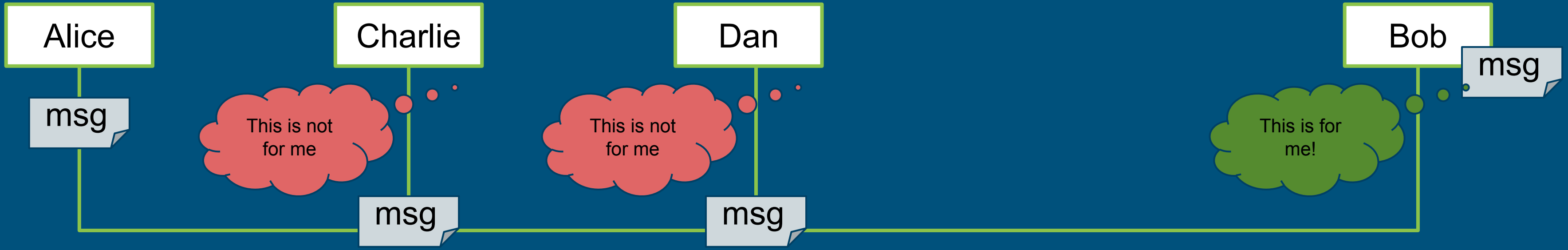
- Instead they are sent in broadcast
- Every node in the network overhears the packet
- The node checks if it is the destination, or if it needs to reroute it, or discard it
- Something against it? Wait for it...



How do they work?

Intuitively, you may think that when systems communicate over the network their packets go directly to the destination...

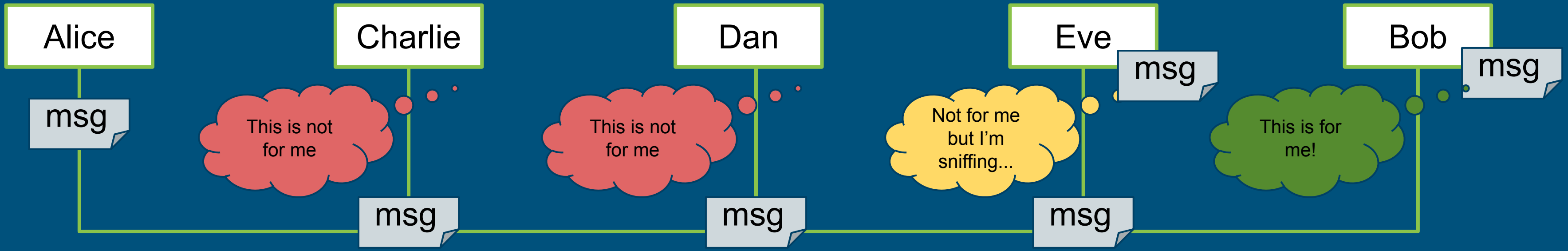
- Instead they are sent in broadcast
- Every node in the network overhears the packet
- The node checks if it is the destination, or if it needs to reroute it, or discard it
- Something against it? Wait for it...



How do they work?

Intuitively, you may think that when systems communicate over the network their packets go directly to the destination...

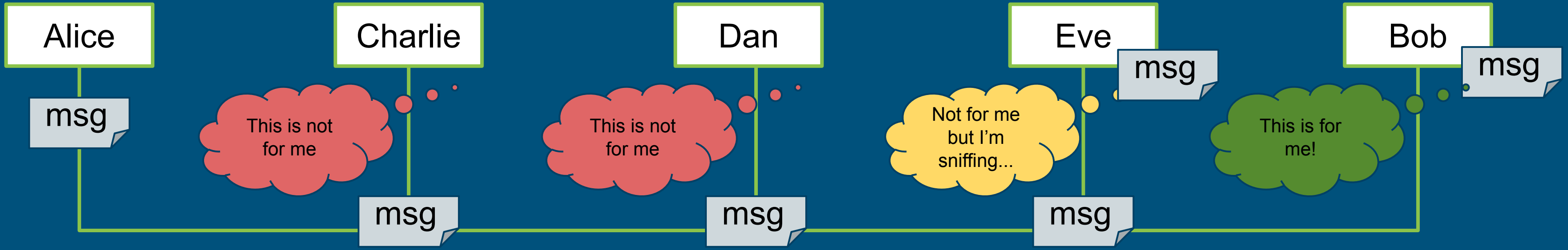
- Instead they are sent in broadcast
- Every node in the network overhears the packet
- The node checks if it is the destination, or if it needs to reroute it, or discard it
- Something against it? Wait for it...



How do they work?

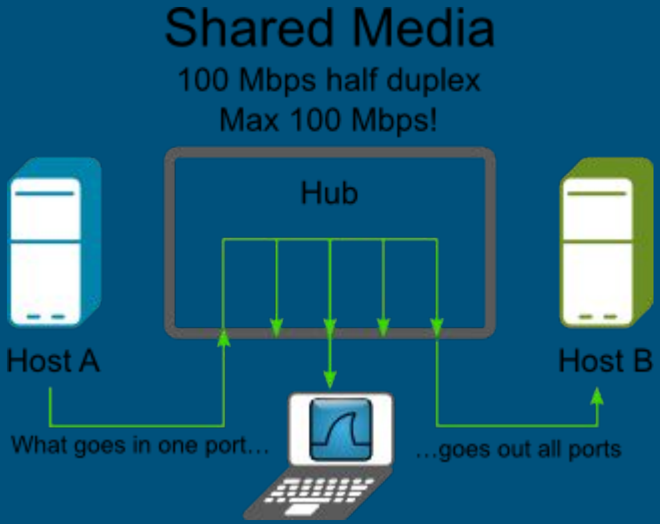
Well, ACTUALLY over the last couple of decades you can't really sniff everything….

- Switched ethernet LANs
  - Your host is only on one segment… ARP may be your enemy here
- Only Broadcasts, multicasts and your segment (plus all wireless obviously)



Old Network Setups

If your sharing media is below OSI layer 2, well, then it works like wireless pretty much...



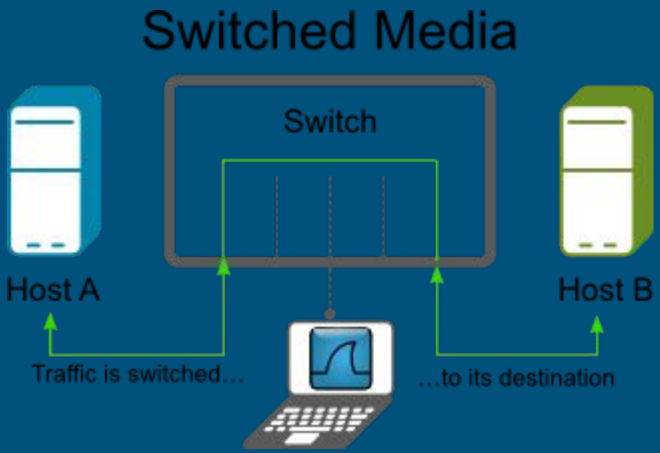
Remember:

- Hubs (layer 1)
- Repeaters (layer 1)
- Switches (layer 2)
- Routers (layer 3)

Read more at: https://wiki.wireshark.org/CaptureSetup/Ethernet#Switched_Ethernet

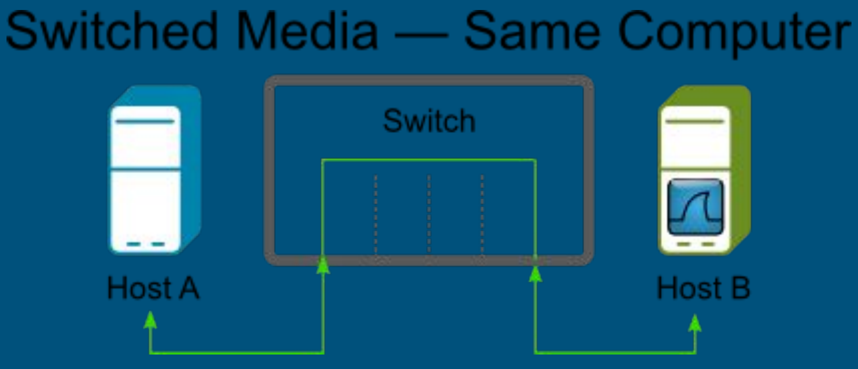
Really Sniffing Everything

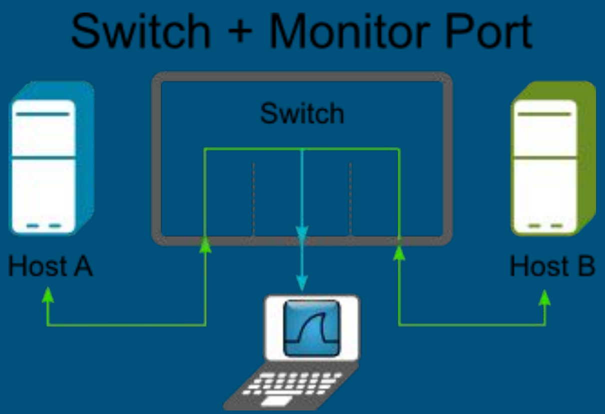
This applies only **<u>if you are the network administrator</u>** or you can mess with the wires of the switch (not advised).



Read more at:
https://wiki.wireshark.org/CaptureSetup/Ethernet#Switched_Ethernet

Really Sniffing Everything





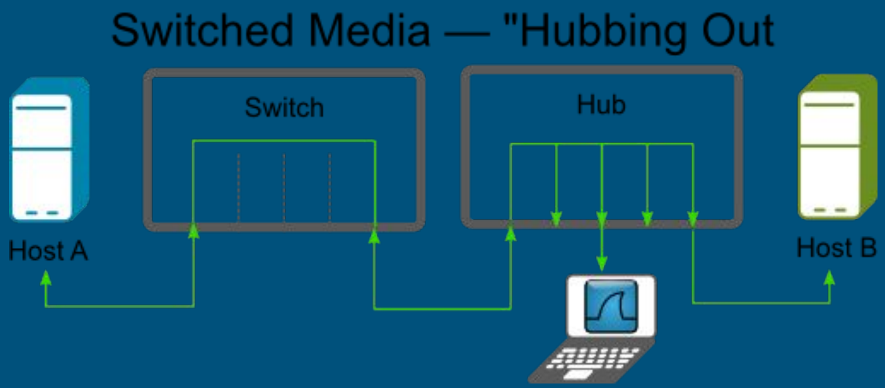
If you want to capture the traffic to/from B, just sniff from B

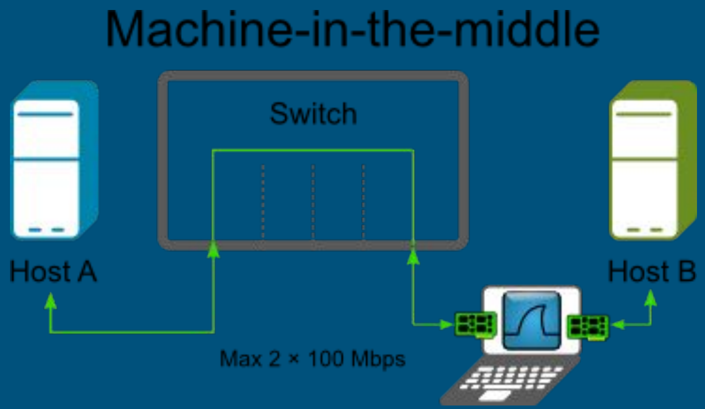
Needless to say, you need to have access to B…

Use a router or a switch with a monitor port

PORT MIRRORING $$

Really Sniffing Everything





Use an old hub on the network segment.
You can also use a TAP, which is more sophisticated.

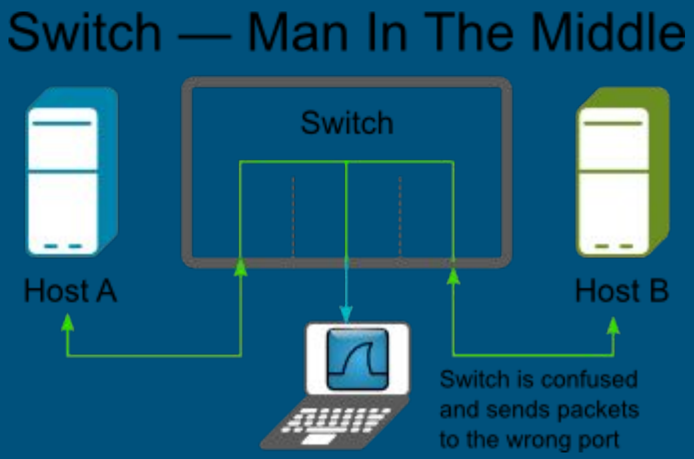
You need to unplug stuff...

You can set up your monitoring machine as a bridge, however you will need two separate NICs...

Really Sniffing Everything

The ILLEGAL ways...

- ARP Poisoning
  - Trick the machines into believing that your MAC is the MAC of the other machine, so all the traffic is directed to you.
- MAC Flooding
  - Send plenty of fake ARP messages to the router until its table is filled and no ARP is used anymore to keep up the pace.


Switch — Man In The Middle
Switch
Host A
Host B
Switch is confused and sends packets to the wrong port

Please note these are not nice and you should not try them unless the LAN is yours.

Did I say Wireless?

Well, that’s also a problem… If wireless cuts you out at the PHY level then you’re done.

- TDMA in general
  - LTE…
- For WiFi we are pretty much covered…

Simple Example

Suppose you want to visit unibo.it (consider a wireless environment)

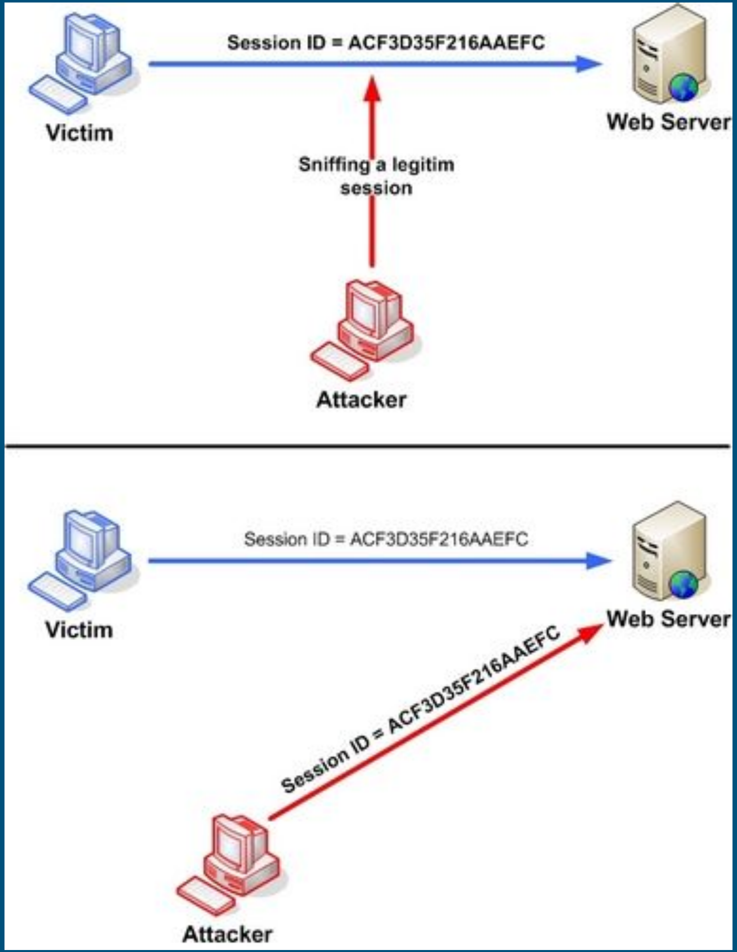
- Basically, you shout “Somebody give me unibo.it!”
- The message is overheard by anyone on the network
  - Including the router, who is the intended recipient
- The router sends it to the destination
- Once it receives the answer, it send the message again on the network
- Everybody overhears it
  - Including you, who are the intended recipient

What can you sniff?

DATA LINK FRAMES!

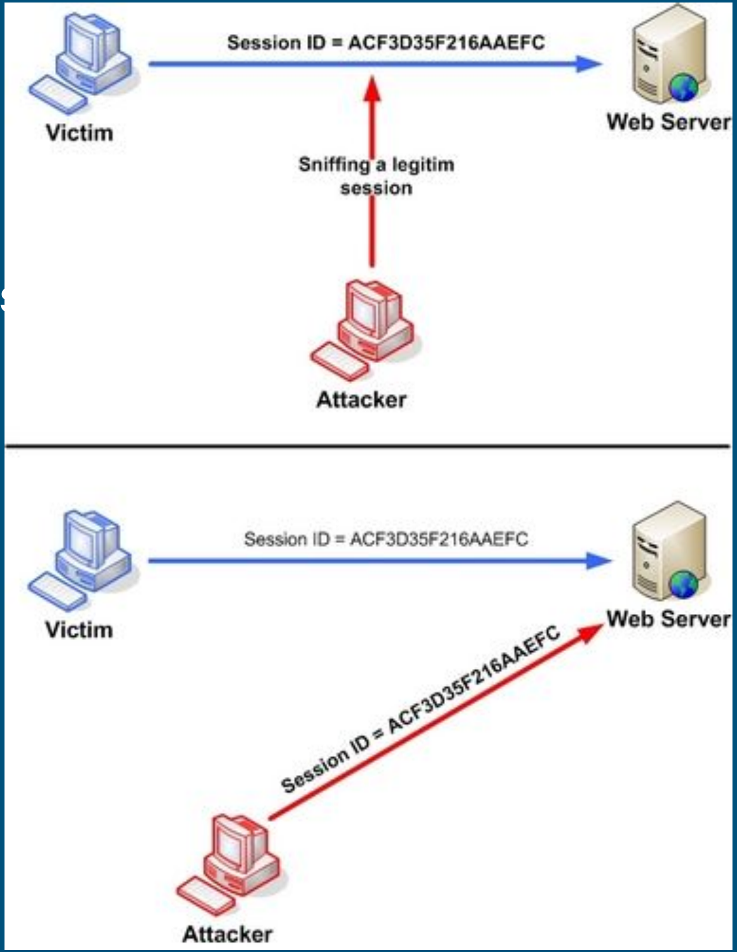
Then the analyzer decodes them (unless they are encoded) and gives you back the highest layer (plus all the envelopes).

Clearly, you find the higher layer datagram in the payload of the smaller layer one...


Session ID = ACF3D35F216AAEFC
Victim
Web Server
Sniffing a legitim session
Attacker
Session ID = ACF3D35F216AAEFC
Victim
Web Server
Session ID = ACF3D35F216AAEFC
Attacker

What can you sniff?

- Basically, all the information sent in clear
- Anyone with a packet sniffer can gain acce
  to such information
- If the connection is encrypted, the information is more secure
  - But still, you are receiving it
- Consider if your user credentials for a harmless website are sent in plain text
  - And you use the same credentials for gmail
  - ... and for you bank account ...
- Typical man-in-the-middle
  - Example: cookie hijacking


Victim
Session ID = ACF3D35F216AAEFC
Web Server
Sniffing a legitim session
Attacker
Victim
Session ID = ACF3D35F216AAEFC
Web Server
Session ID = ACF3D35F216AAEFC
Attacker

Wireshark

What is WireShark?

It is a network analyzer tool: it allows us to see all the packets that go through a network…

- Why is my network stuck every Friday evening from 6PM to 8PM?
- Why computer X can’t connect to the Internet?
- Why the A department can’t connect to the internal servers?

Wireshark helps us troubleshoot the network (correct use)

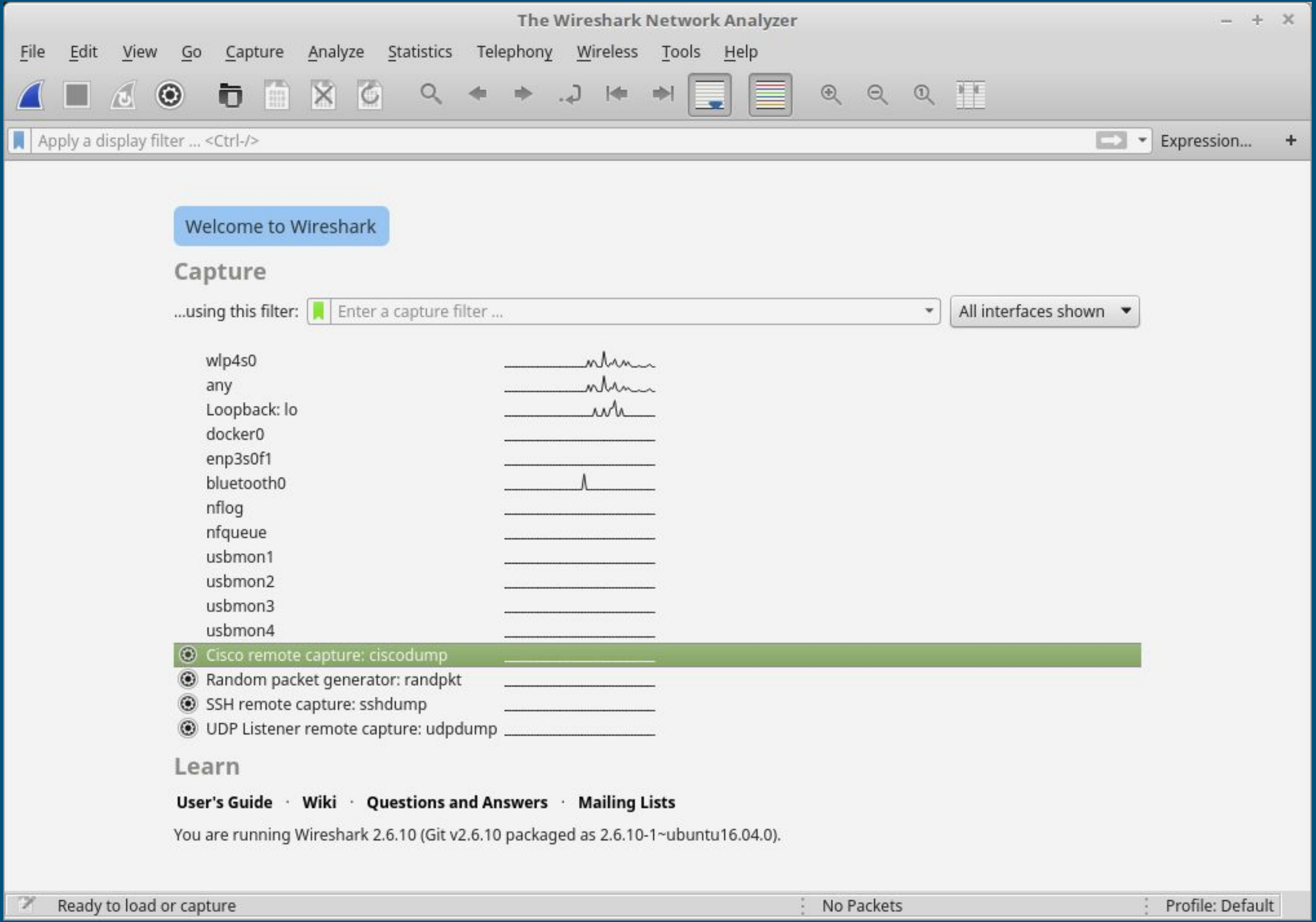
Available for Windows/MAC OS/Linux: https://www.wireshark.org/download.html

- Open Source with GUI

What is WireShark?

- What people use wireshark for:
  - Network administrators -> troubleshoot network problems
  - Network security engineers -> examine security problems
  - QA engineers -> verify network applications
  - Developers -> debug protocol implementations
  - People -> learn network protocol internals
  - Attackers -> you can imagine...
- Some features:
  - Capture live packet data and display it
  - Import network traces and save them
  - Filtering/Coloring/Search
  - Create network statistics

How it looks like



Capture

...using this filter: Enter a capture filter ... All interfaces shown

wlp4s0
any
Loopback: lo
docker0
enp3s0f1
bluetooth0
nflog
nfqueue
usbmon1
usbmon2
usbmon3
usbmon4
Cisco remote capture: ciscodump
Random packet generator: randpkt
SSH remote capture: sshdump
UDP Listener remote capture: udpdump

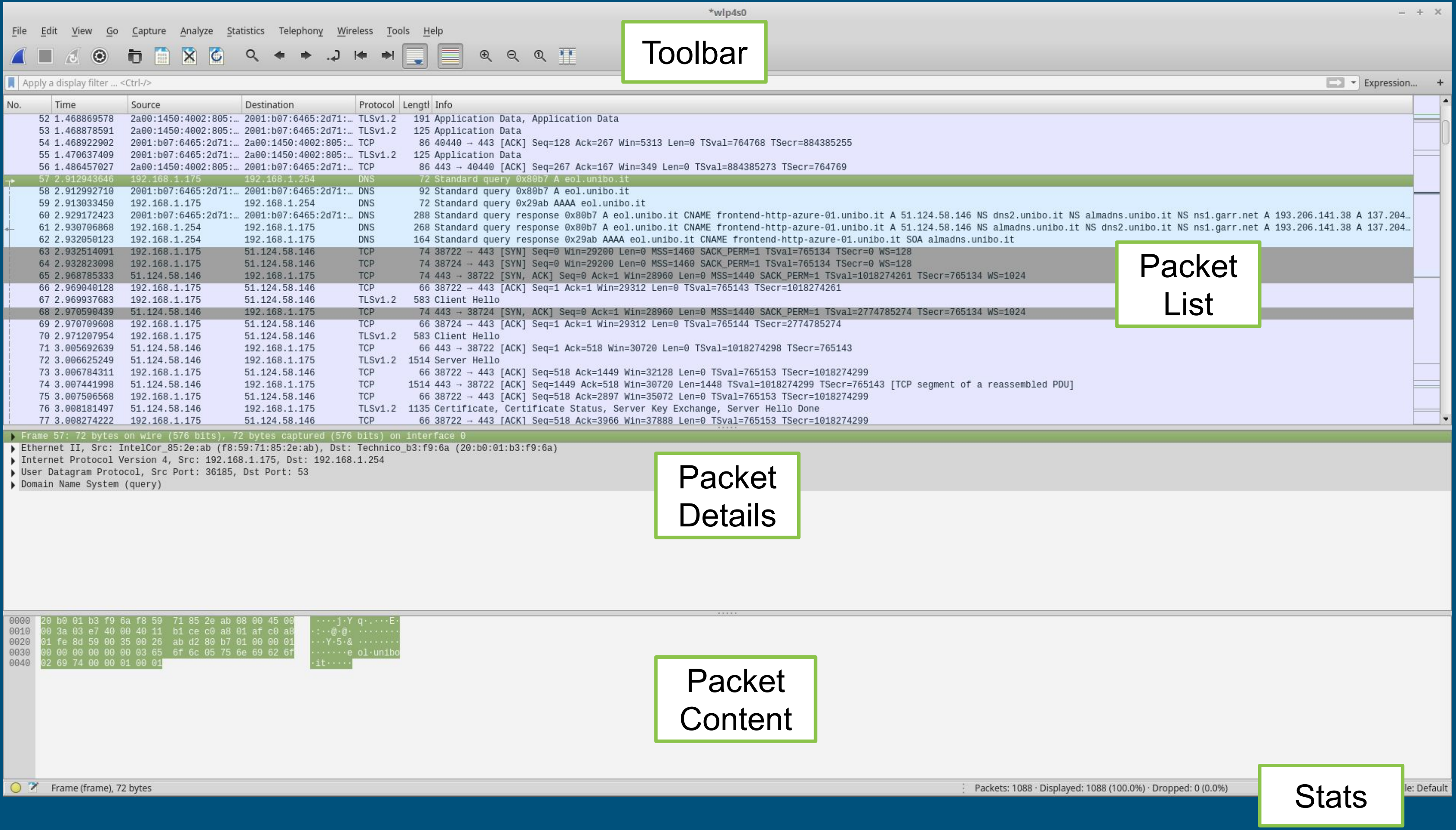
Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 2.6.10 (Git v2.6.10 packaged as 2.6.10-1~ubuntu16.04.0).

Ready to load or capture | No Packets | Profile: Default

How it looks like


*wlp4s0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Toolbar
Packet List
Packet Details
Packet Content
Stats

```
No.   Time          Source                 Destination            Protocol Lengtl Info
   52 1.468869578   2a00:1450:4002:805:…   2001:b07:6465:2d71:…   TLSv1.2   191 Application Data, Application Data
   53 1.468878591   2a00:1450:4002:805:…   2001:b07:6465:2d71:…   TLSv1.2   125 Application Data
   54 1.468922902   2001:b07:6465:2d71:…   2a00:1450:4002:805:…   TCP        86 40440 → 443 [ACK] Seq=128 Ack=267 Win=5313 Len=0 TSval=764768 TSecr=884385255
   55 1.470637409   2001:b07:6465:2d71:…   2a00:1450:4002:805:…   TLSv1.2   125 Application Data
   56 1.486457027   2a00:1450:4002:805:…   2001:b07:6465:2d71:…   TCP        86 443 → 40440 [ACK] Seq=267 Ack=167 Win=349 Len=0 TSval=884385273 TSecr=764769
   57 2.912943646   192.168.1.175          192.168.1.254          DNS        72 Standard query 0x80b7 A eol.unibo.it
   58 2.912992710   2001:b07:6465:2d71:…   2001:b07:6465:2d71:…   DNS        92 Standard query 0x80b7 A eol.unibo.it
   59 2.913033450   192.168.1.175          192.168.1.254          DNS        72 Standard query 0x29ab AAAA eol.unibo.it
   60 2.929172423   2001:b07:6465:2d71:…   2001:b07:6465:2d71:…   DNS       288 Standard query response 0x80b7 A eol.unibo.it CNAME frontend-http-azure-01.unibo.it A 51.124.58.146 NS dns2.unibo.it NS almadns.unibo.it NS ns1.garr.net A 193.206.141.38 A 137.204…
   61 2.930706868   192.168.1.254          192.168.1.175          DNS       268 Standard query response 0x80b7 A eol.unibo.it CNAME frontend-http-azure-01.unibo.it A 51.124.58.146 NS almadns.unibo.it NS dns2.unibo.it NS ns1.garr.net A 193.206.141.38 A 137.204…
   62 2.932050123   192.168.1.254          192.168.1.175          DNS       164 Standard query response 0x29ab AAAA eol.unibo.it CNAME frontend-http-azure-01.unibo.it SOA almadns.unibo.it
   63 2.932514091   192.168.1.175          51.124.58.146          TCP        74 38722 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=765134 TSecr=0 WS=128
   64 2.932823098   192.168.1.175          51.124.58.146          TCP        74 38724 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=765134 TSecr=0 WS=128
   65 2.968785333   51.124.58.146          192.168.1.175          TCP        74 443 → 38722 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1440 SACK_PERM=1 TSval=1018274261 TSecr=765134 WS=1024
   66 2.969040128   192.168.1.175          51.124.58.146          TCP        66 38722 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=765143 TSecr=1018274261
   67 2.969937683   192.168.1.175          51.124.58.146          TLSv1.2   583 Client Hello
   68 2.970590439   51.124.58.146          192.168.1.175          TCP        74 443 → 38724 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1440 SACK_PERM=1 TSval=2774785274 TSecr=765134 WS=1024
   69 2.970709608   192.168.1.175          51.124.58.146          TCP        66 38724 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=765144 TSecr=2774785274
   70 2.971207954   192.168.1.175          51.124.58.146          TLSv1.2   583 Client Hello
   71 3.005692639   51.124.58.146          192.168.1.175          TCP        66 443 → 38722 [ACK] Seq=1 Ack=518 Win=30720 Len=0 TSval=1018274298 TSecr=765143
   72 3.006625249   51.124.58.146          192.168.1.175          TLSv1.2  1514 Server Hello
   73 3.006784311   192.168.1.175          51.124.58.146          TCP        66 38722 → 443 [ACK] Seq=518 Ack=1449 Win=32128 Len=0 TSval=765153 TSecr=1018274299
   74 3.007441998   51.124.58.146          192.168.1.175          TCP      1514 443 → 38722 [ACK] Seq=1449 Ack=518 Win=30720 Len=1448 TSval=1018274299 TSecr=765143 [TCP segment of a reassembled PDU]
   75 3.007506568   192.168.1.175          51.124.58.146          TCP        66 38722 → 443 [ACK] Seq=518 Ack=2897 Win=35072 Len=0 TSval=765153 TSecr=1018274299
   76 3.008181497   51.124.58.146          192.168.1.175          TLSv1.2  1135 Certificate, Certificate Status, Server Key Exchange, Server Hello Done
   77 3.008274222   192.168.1.175          51.124.58.146          TCP        66 38722 → 443 [ACK] Seq=518 Ack=3966 Win=37888 Len=0 TSval=765153 TSecr=1018274299
```

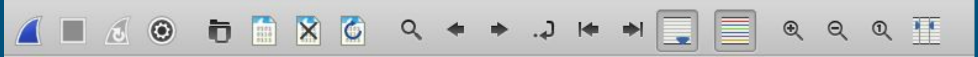
```
▶ Frame 57: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface 0
▶ Ethernet II, Src: IntelCor_85:2e:ab (f8:59:71:85:2e:ab), Dst: Technico_b3:f9:6a (20:b0:01:b3:f9:6a)
▶ Internet Protocol Version 4, Src: 192.168.1.175, Dst: 192.168.1.254
▶ User Datagram Protocol, Src Port: 36185, Dst Port: 53
▶ Domain Name System (query)
```

```
0000  20 b0 01 b3 f9 6a f8 59  71 85 2e ab 08 00 45 00    ····j·Y q·.···E·
0010  00 3a 03 e7 40 00 40 11  b1 ce c0 a8 01 af c0 a8   ·:··@·@· ········
0020  01 fe 8d 59 00 35 00 26  ab d2 80 b7 01 00 00 01   ···Y·5·& ········
0030  00 00 00 00 00 00 03 65  6f 6c 05 75 6e 69 62 6f   ·······e ol·unibo
0040  02 69 74 00 00 01 00 01                            ·it·····
```

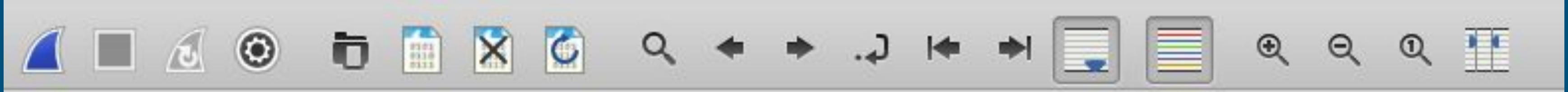
```
Frame (frame), 72 bytes          Packets: 1088 · Displayed: 1088 (100.0%) · Dropped: 0 (0.0%)          le: Default
```

Wireshark Menu

- **File** – Open, merge, export and print capture files
- **Edit** – Search packets, mark them, preferences
- **View** – Coloring packets and view options
- **Go** – Through this menu it is possible to go to a specific packet
- **Capture** – To start capturing and edit capture filters
- **Analyze** – Filtering packets, dissecting protocols
- **Statistics** – To generate and display statistics
- **Telephony** – Telephony related statistics
- **Wireless** – To show wireless related statistics
- **Tools** – Various tools available in wireshark
- **Help** – Help, manual pages

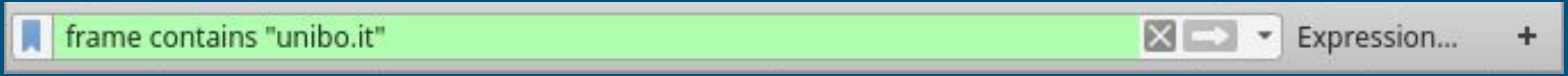


Wireshark Toolbar



- Start
- Stop
- Restart
- Options
- Open
- Save
- Close
- Reload
- Find
- Go to packets
- Auto scroll
- Colorize
- Zoom options
- Resize Columns

Wireshark Filtering Toolbar



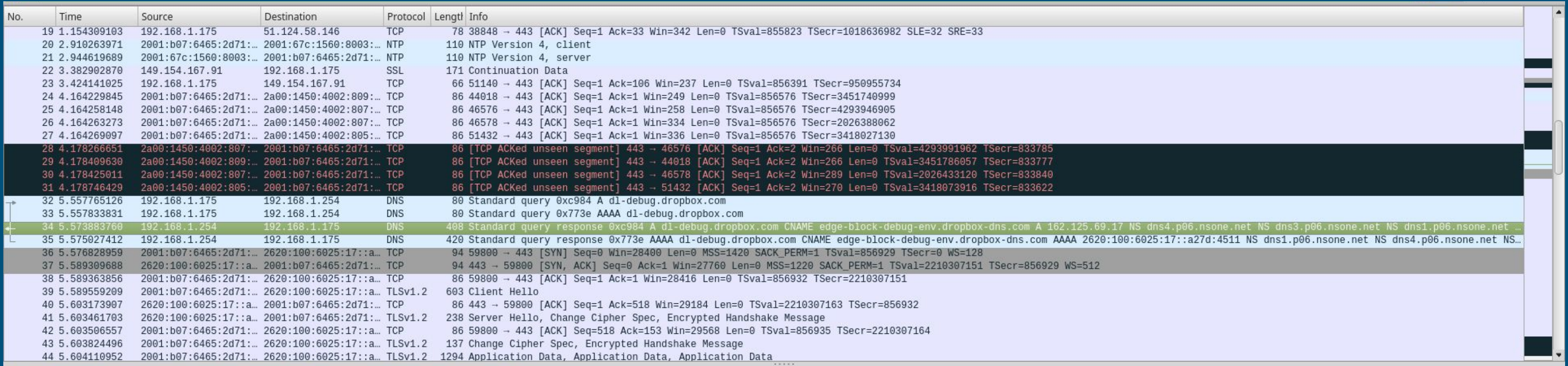
frame contains "unibo.it"

- Bookmarks
- Filter Input
- Clear
- Apply

It is probably one of the most powerful tools of wireshark: we’ll see how many packets are generated even in low populated networks in short time

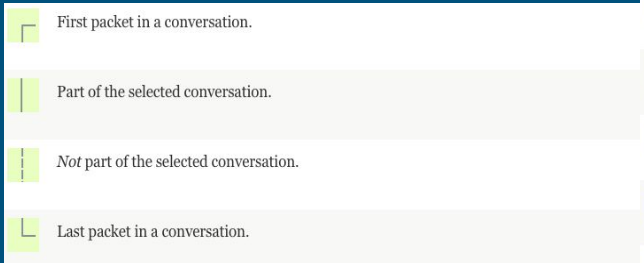
- Filtering is essential

The Packet List Panel



No.	Time	Source	Destination	Protocol	Lengtl	Info
19	1.154309103	192.168.1.175	51.124.58.146	TCP	78	38848 → 443 [ACK] Seq=1 Ack=33 Win=342 Len=0 TSval=855823 TSecr=1018636982 SLE=32 SRE=33
20	2.910263971	2001:b07:6465:2d71:…	2001:67c:1560:8003:…	NTP	110	NTP Version 4, client
21	2.944619689	2001:67c:1560:8003:…	2001:b07:6465:2d71:…	NTP	110	NTP Version 4, server
22	3.382902870	149.154.167.91	192.168.1.175	SSL	171	Continuation Data
23	3.424141025	192.168.1.175	149.154.167.91	TCP	66	51140 → 443 [ACK] Seq=1 Ack=106 Win=237 Len=0 TSval=856391 TSecr=950955734
24	4.164229845	2001:b07:6465:2d71:…	2a00:1450:4002:809:…	TCP	86	44018 → 443 [ACK] Seq=1 Ack=1 Win=249 Len=0 TSval=856576 TSecr=3451740999
25	4.164258148	2001:b07:6465:2d71:…	2a00:1450:4002:807:…	TCP	86	46576 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=0 TSval=856576 TSecr=4293946905
26	4.164263273	2001:b07:6465:2d71:…	2a00:1450:4002:807:…	TCP	86	46578 → 443 [ACK] Seq=1 Ack=1 Win=334 Len=0 TSval=856576 TSecr=2026388062
27	4.164269097	2001:b07:6465:2d71:…	2a00:1450:4002:805:…	TCP	86	51432 → 443 [ACK] Seq=1 Ack=1 Win=336 Len=0 TSval=856576 TSecr=3418027130
28	4.178266651	2a00:1450:4002:807:…	2001:b07:6465:2d71:…	TCP	86	[TCP ACKed unseen segment] 443 → 46576 [ACK] Seq=1 Ack=2 Win=266 Len=0 TSval=4293991962 TSecr=833785
29	4.178409630	2a00:1450:4002:809:…	2001:b07:6465:2d71:…	TCP	86	[TCP ACKed unseen segment] 443 → 44018 [ACK] Seq=1 Ack=2 Win=266 Len=0 TSval=3451786057 TSecr=833777
30	4.178425011	2a00:1450:4002:807:…	2001:b07:6465:2d71:…	TCP	86	[TCP ACKed unseen segment] 443 → 46578 [ACK] Seq=1 Ack=2 Win=289 Len=0 TSval=2026433120 TSecr=833840
31	4.178746429	2a00:1450:4002:805:…	2001:b07:6465:2d71:…	TCP	86	[TCP ACKed unseen segment] 443 → 51432 [ACK] Seq=1 Ack=2 Win=270 Len=0 TSval=3418073916 TSecr=833622
32	5.557765126	192.168.1.175	192.168.1.254	DNS	80	Standard query 0xc984 A dl-debug.dropbox.com
33	5.557833831	192.168.1.175	192.168.1.254	DNS	80	Standard query 0x773e AAAA dl-debug.dropbox.com
34	5.573883760	192.168.1.254	192.168.1.175	DNS	408	Standard query response 0xc984 A dl-debug.dropbox.com CNAME edge-block-debug-env.dropbox-dns.com A 162.125.69.17 NS dns4.p06.nsone.net NS dns3.p06.nsone.net NS dns1.p06.nsone.net …
35	5.575027412	192.168.1.254	192.168.1.175	DNS	420	Standard query response 0x773e AAAA dl-debug.dropbox.com CNAME edge-block-debug-env.dropbox-dns.com AAAA 2620:100:6025:17::a27d:4511 NS dns1.p06.nsone.net NS dns4.p06.nsone.net NS…
36	5.576828959	2001:b07:6465:2d71:…	2620:100:6025:17::a…	TCP	94	59800 → 443 [SYN] Seq=0 Win=28400 Len=0 MSS=1420 SACK_PERM=1 TSval=856929 TSecr=0 WS=128
37	5.589309688	2620:100:6025:17::a…	2001:b07:6465:2d71:…	TCP	94	443 → 59800 [SYN, ACK] Seq=0 Ack=1 Win=27760 Len=0 MSS=1220 SACK_PERM=1 TSval=2210307151 TSecr=856929 WS=512
38	5.589363856	2001:b07:6465:2d71:…	2620:100:6025:17::a…	TCP	86	59800 → 443 [ACK] Seq=1 Ack=1 Win=28416 Len=0 TSval=856932 TSecr=2210307151
39	5.589559209	2001:b07:6465:2d71:…	2620:100:6025:17::a…	TLSv1.2	603	Client Hello
40	5.603173907	2620:100:6025:17::a…	2001:b07:6465:2d71:…	TCP	86	443 → 59800 [ACK] Seq=1 Ack=518 Win=29184 Len=0 TSval=2210307163 TSecr=856932
41	5.603461703	2620:100:6025:17::a…	2001:b07:6465:2d71:…	TLSv1.2	238	Server Hello, Change Cipher Spec, Encrypted Handshake Message
42	5.603506557	2001:b07:6465:2d71:…	2620:100:6025:17::a…	TCP	86	59800 → 443 [ACK] Seq=518 Ack=153 Win=29568 Len=0 TSval=856935 TSecr=2210307164
43	5.603824496	2001:b07:6465:2d71:…	2620:100:6025:17::a…	TLSv1.2	137	Change Cipher Spec, Encrypted Handshake Message
44	5.604110952	2001:b07:6465:2d71:…	2620:100:6025:17::a…	TLSv1.2	1294	Application Data, Application Data, Application Data

- One packet per line
- If selected, info about the flow
- Source, destination, protocol, etc...

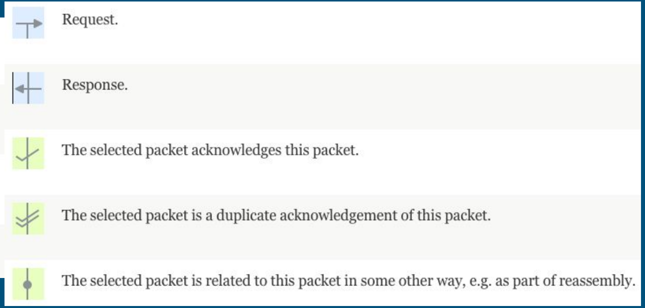


First packet in a conversation.

Part of the selected conversation.

Not part of the selected conversation.

Last packet in a conversation.



Request.

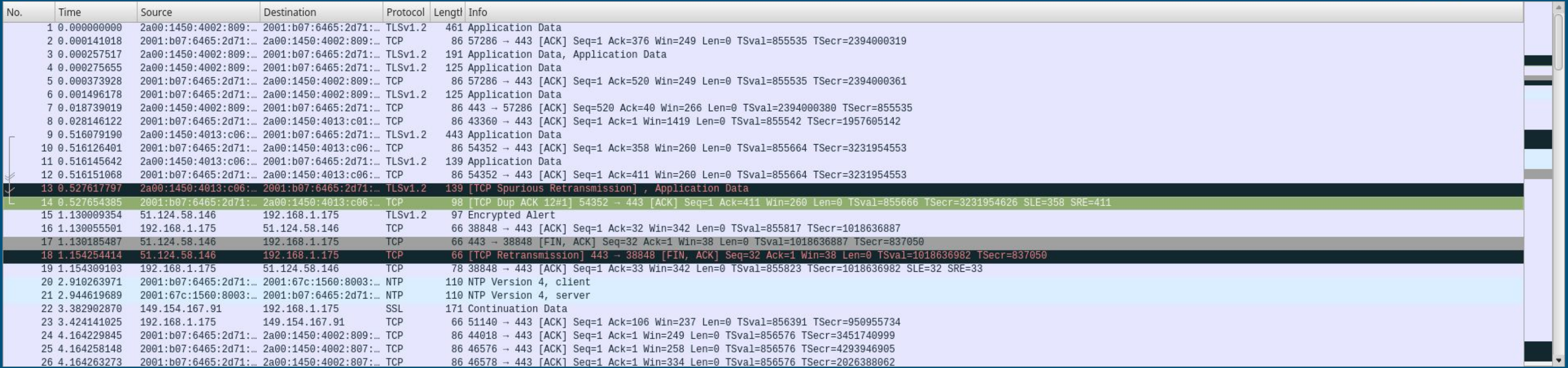
Response.

The selected packet acknowledges this packet.

The selected packet is a duplicate acknowledgement of this packet.

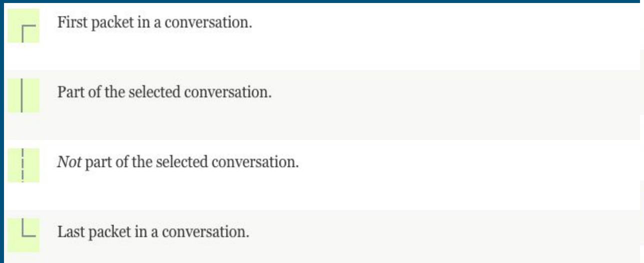
The selected packet is related to this packet in some other way, e.g. as part of reassembly.

The Packet List Panel



No.	Time	Source	Destination	Protocol	Lengtl	Info
1	0.000000000	2a00:1450:4002:809:…	2001:b07:6465:2d71:…	TLSv1.2	461	Application Data
2	0.000141018	2001:b07:6465:2d71:…	2a00:1450:4002:809:…	TCP	86	57286 → 443 [ACK] Seq=1 Ack=376 Win=249 Len=0 TSval=855535 TSecr=2394000319
3	0.000257517	2a00:1450:4002:809:…	2001:b07:6465:2d71:…	TLSv1.2	191	Application Data, Application Data
4	0.000275655	2a00:1450:4002:809:…	2001:b07:6465:2d71:…	TLSv1.2	125	Application Data
5	0.000373928	2001:b07:6465:2d71:…	2a00:1450:4002:809:…	TCP	86	57286 → 443 [ACK] Seq=1 Ack=520 Win=249 Len=0 TSval=855535 TSecr=2394000361
6	0.001496178	2001:b07:6465:2d71:…	2a00:1450:4002:809:…	TLSv1.2	125	Application Data
7	0.018739019	2a00:1450:4002:809:…	2001:b07:6465:2d71:…	TCP	86	443 → 57286 [ACK] Seq=520 Ack=40 Win=266 Len=0 TSval=2394000380 TSecr=855535
8	0.028146122	2001:b07:6465:2d71:…	2a00:1450:4013:c01:…	TCP	86	43360 → 443 [ACK] Seq=1 Ack=1 Win=1419 Len=0 TSval=855542 TSecr=1957605142
9	0.516079190	2a00:1450:4013:c06:…	2001:b07:6465:2d71:…	TLSv1.2	443	Application Data
10	0.516126401	2001:b07:6465:2d71:…	2a00:1450:4013:c06:…	TCP	86	54352 → 443 [ACK] Seq=1 Ack=358 Win=260 Len=0 TSval=855664 TSecr=3231954553
11	0.516145642	2a00:1450:4013:c06:…	2001:b07:6465:2d71:…	TLSv1.2	139	Application Data
12	0.516151068	2001:b07:6465:2d71:…	2a00:1450:4013:c06:…	TCP	86	54352 → 443 [ACK] Seq=1 Ack=411 Win=260 Len=0 TSval=855664 TSecr=3231954553
13	0.527617797	2a00:1450:4013:c06:…	2001:b07:6465:2d71:…	TLSv1.2	139	[TCP Spurious Retransmission] , Application Data
14	0.527654385	2001:b07:6465:2d71:…	2a00:1450:4013:c06:…	TCP	98	[TCP Dup ACK 12#1] 54352 → 443 [ACK] Seq=1 Ack=411 Win=260 Len=0 TSval=855666 TSecr=3231954626 SLE=358 SRE=411
15	1.130009354	51.124.58.146	192.168.1.175	TLSv1.2	97	Encrypted Alert
16	1.130055501	192.168.1.175	51.124.58.146	TCP	66	38848 → 443 [ACK] Seq=1 Ack=32 Win=342 Len=0 TSval=855817 TSecr=1018636887
17	1.130185487	51.124.58.146	192.168.1.175	TCP	66	443 → 38848 [FIN, ACK] Seq=32 Ack=1 Win=38 Len=0 TSval=1018636887 TSecr=837050
18	1.154254414	51.124.58.146	192.168.1.175	TCP	66	[TCP Retransmission] 443 → 38848 [FIN, ACK] Seq=32 Ack=1 Win=38 Len=0 TSval=1018636982 TSecr=837050
19	1.154309103	192.168.1.175	51.124.58.146	TCP	78	38848 → 443 [ACK] Seq=1 Ack=33 Win=342 Len=0 TSval=855823 TSecr=1018636982 SLE=32 SRE=33
20	2.910263971	2001:b07:6465:2d71:…	2001:67c:1560:8003:…	NTP	110	NTP Version 4, client
21	2.944619689	2001:67c:1560:8003:…	2001:b07:6465:2d71:…	NTP	110	NTP Version 4, server
22	3.382902870	149.154.167.91	192.168.1.175	SSL	171	Continuation Data
23	3.424141025	192.168.1.175	149.154.167.91	TCP	66	51140 → 443 [ACK] Seq=1 Ack=106 Win=237 Len=0 TSval=856391 TSecr=950955734
24	4.164229845	2001:b07:6465:2d71:…	2a00:1450:4002:809:…	TCP	86	44018 → 443 [ACK] Seq=1 Ack=1 Win=249 Len=0 TSval=856576 TSecr=3451740999
25	4.164258148	2001:b07:6465:2d71:…	2a00:1450:4002:807:…	TCP	86	46576 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=0 TSval=856576 TSecr=4293946905
26	4.164263273	2001:b07:6465:2d71:…	2a00:1450:4002:807:…	TCP	86	46578 → 443 [ACK] Seq=1 Ack=1 Win=334 Len=0 TSval=856576 TSecr=2026388062

- One packet per line
- If selected, info about the flow
- Source, destination, protocol, etc...

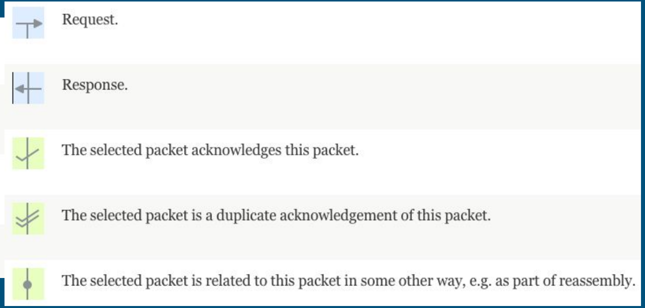


First packet in a conversation.

Part of the selected conversation.

Not part of the selected conversation.

Last packet in a conversation.



Request.

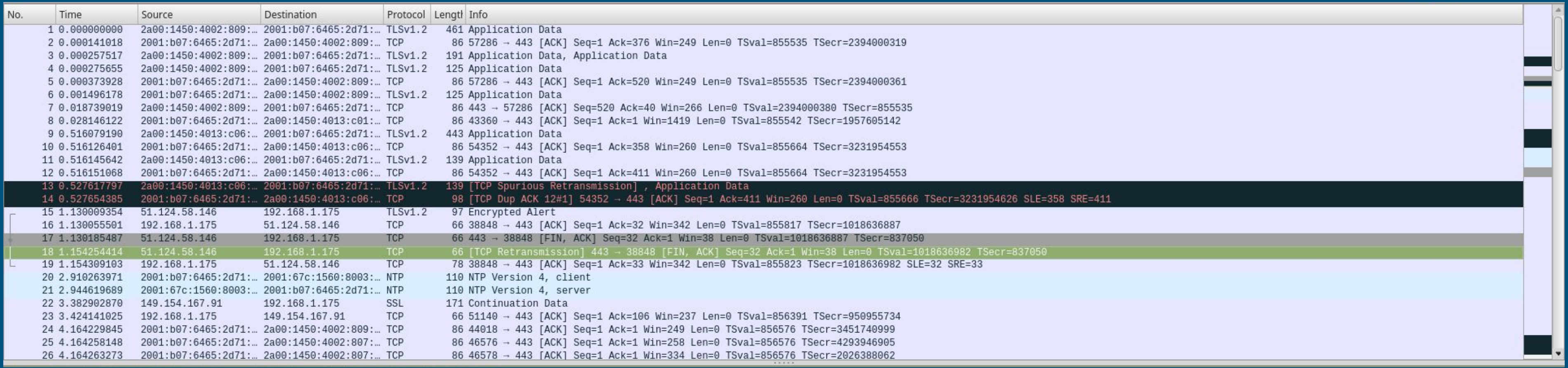
Response.

The selected packet acknowledges this packet.

The selected packet is a duplicate acknowledgement of this packet.

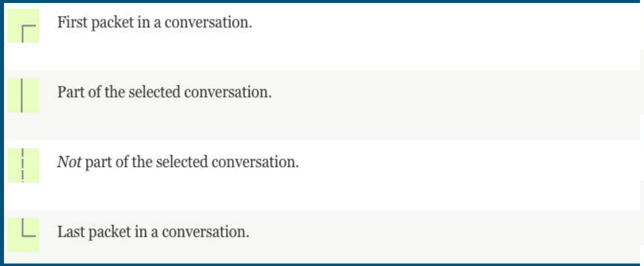
The selected packet is related to this packet in some other way, e.g. as part of reassembly.

The Packet List Panel

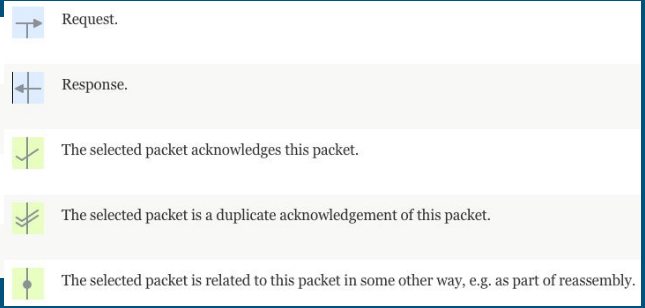


```
No.   Time         Source               Destination          Protocol  Lengtl Info
    1 0.000000000  2a00:1450:4002:809:… 2001:b07:6465:2d71:… TLSv1.2   461 Application Data
    2 0.000141018  2001:b07:6465:2d71:… 2a00:1450:4002:809:… TCP        86 57286 → 443 [ACK] Seq=1 Ack=376 Win=249 Len=0 TSval=855535 TSecr=2394000319
    3 0.000257517  2a00:1450:4002:809:… 2001:b07:6465:2d71:… TLSv1.2   191 Application Data, Application Data
    4 0.000275655  2a00:1450:4002:809:… 2001:b07:6465:2d71:… TLSv1.2   125 Application Data
    5 0.000373928  2001:b07:6465:2d71:… 2a00:1450:4002:809:… TCP        86 57286 → 443 [ACK] Seq=1 Ack=520 Win=249 Len=0 TSval=855535 TSecr=2394000361
    6 0.001496178  2001:b07:6465:2d71:… 2a00:1450:4002:809:… TLSv1.2   125 Application Data
    7 0.018739019  2a00:1450:4002:809:… 2001:b07:6465:2d71:… TCP        86 443 → 57286 [ACK] Seq=520 Ack=40 Win=266 Len=0 TSval=2394000380 TSecr=855535
    8 0.028146122  2001:b07:6465:2d71:… 2a00:1450:4013:c01:… TCP        86 43360 → 443 [ACK] Seq=1 Ack=1 Win=1419 Len=0 TSval=855542 TSecr=1957605142
    9 0.516079190  2a00:1450:4013:c06:… 2001:b07:6465:2d71:… TLSv1.2   443 Application Data
   10 0.516126401  2001:b07:6465:2d71:… 2a00:1450:4013:c06:… TCP        86 54352 → 443 [ACK] Seq=1 Ack=358 Win=260 Len=0 TSval=855664 TSecr=3231954553
   11 0.516145642  2a00:1450:4013:c06:… 2001:b07:6465:2d71:… TLSv1.2   139 Application Data
   12 0.516151068  2001:b07:6465:2d71:… 2a00:1450:4013:c06:… TCP        86 54352 → 443 [ACK] Seq=1 Ack=411 Win=260 Len=0 TSval=855664 TSecr=3231954553
   13 0.527617797  2a00:1450:4013:c06:… 2001:b07:6465:2d71:… TLSv1.2   139 [TCP Spurious Retransmission] , Application Data
   14 0.527654385  2001:b07:6465:2d71:… 2a00:1450:4013:c06:… TCP        98 [TCP Dup ACK 12#1] 54352 → 443 [ACK] Seq=1 Ack=411 Win=260 Len=0 TSval=855666 TSecr=3231954626 SLE=358 SRE=411
┌  15 1.130009354  51.124.58.146        192.168.1.175        TLSv1.2    97 Encrypted Alert
│  16 1.130055501  192.168.1.175        51.124.58.146        TCP        66 38848 → 443 [ACK] Seq=1 Ack=32 Win=342 Len=0 TSval=855817 TSecr=1018636887
│  17 1.130185487  51.124.58.146        192.168.1.175        TCP        66 443 → 38848 [FIN, ACK] Seq=32 Ack=1 Win=38 Len=0 TSval=1018636887 TSecr=837050
│  18 1.154254414  51.124.58.146        192.168.1.175        TCP        66 [TCP Retransmission] 443 → 38848 [FIN, ACK] Seq=32 Ack=1 Win=38 Len=0 TSval=1018636982 TSecr=837050
└  19 1.154309103  192.168.1.175        51.124.58.146        TCP        78 38848 → 443 [ACK] Seq=1 Ack=33 Win=342 Len=0 TSval=855823 TSecr=1018636982 SLE=32 SRE=33
   20 2.910263971  2001:b07:6465:2d71:… 2001:67c:1560:8003:… NTP       110 NTP Version 4, client
   21 2.944619689  2001:67c:1560:8003:… 2001:b07:6465:2d71:… NTP       110 NTP Version 4, server
   22 3.382902870  149.154.167.91       192.168.1.175        SSL       171 Continuation Data
   23 3.424141025  192.168.1.175        149.154.167.91       TCP        66 51140 → 443 [ACK] Seq=1 Ack=106 Win=237 Len=0 TSval=856391 TSecr=950955734
   24 4.164229845  2001:b07:6465:2d71:… 2a00:1450:4002:809:… TCP        86 44018 → 443 [ACK] Seq=1 Ack=1 Win=249 Len=0 TSval=856576 TSecr=3451740999
   25 4.164258148  2001:b07:6465:2d71:… 2a00:1450:4002:807:… TCP        86 46576 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=0 TSval=856576 TSecr=4293946905
   26 4.164263273  2001:b07:6465:2d71:… 2a00:1450:4002:807:… TCP        86 46578 → 443 [ACK] Seq=1 Ack=1 Win=334 Len=0 TSval=856576 TSecr=2026388062
```

- One packet per line
- If selected, info about the flow
- Source, destination, protocol, etc...



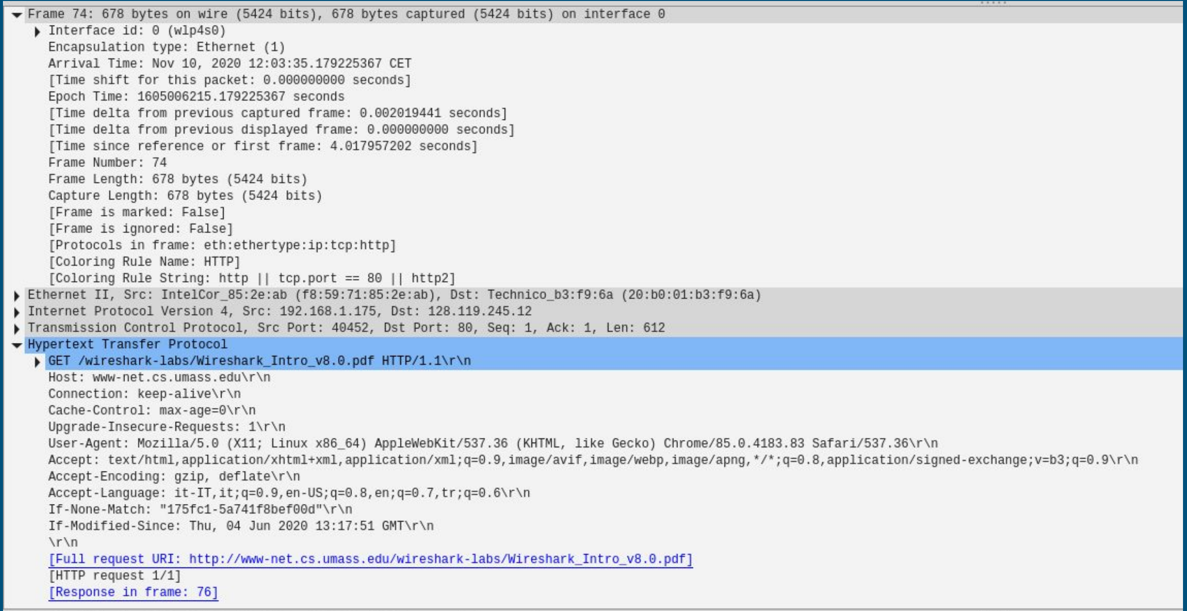
┌	First packet in a conversation.
│	Part of the selected conversation.
┆	*Not* part of the selected conversation.
└	Last packet in a conversation.



→	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.

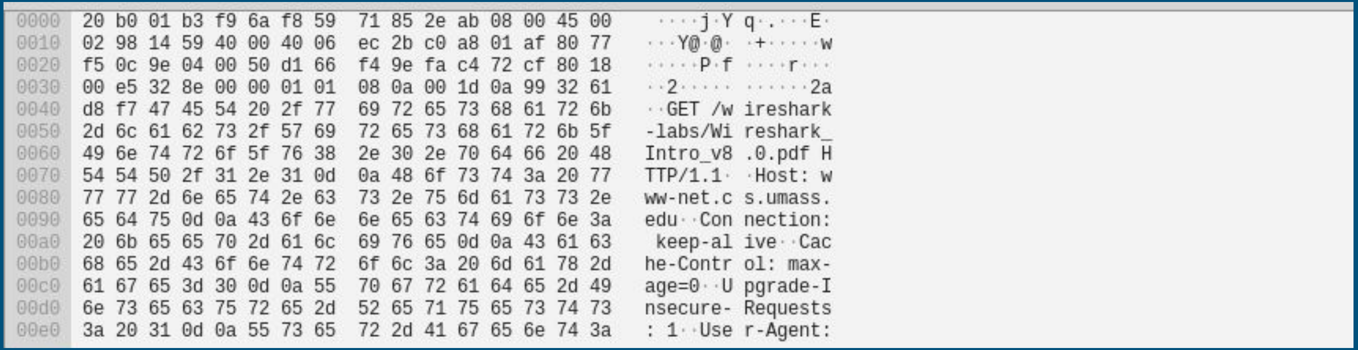
The Packet Details Panel

- It shows the details for a specific selected packet
- It can display additional information enclosed in brackets
- It also shows links if wireshark detects a link with another packet



```
Frame 74: 678 bytes on wire (5424 bits), 678 bytes captured (5424 bits) on interface 0
  Interface id: 0 (wlp4s0)
  Encapsulation type: Ethernet (1)
  Arrival Time: Nov 10, 2020 12:03:35.179225367 CET
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1605006215.179225367 seconds
  [Time delta from previous captured frame: 0.002019441 seconds]
  [Time delta from previous displayed frame: 0.000000000 seconds]
  [Time since reference or first frame: 4.017957202 seconds]
  Frame Number: 74
  Frame Length: 678 bytes (5424 bits)
  Capture Length: 678 bytes (5424 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:tcp:http]
  [Coloring Rule Name: HTTP]
  [Coloring Rule String: http || tcp.port == 80 || http2]
Ethernet II, Src: IntelCor_85:2e:ab (f8:59:71:85:2e:ab), Dst: Technico_b3:f9:6a (20:b0:01:b3:f9:6a)
Internet Protocol Version 4, Src: 192.168.1.175, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 40452, Dst Port: 80, Seq: 1, Ack: 1, Len: 612
Hypertext Transfer Protocol
  GET /wireshark-labs/Wireshark_Intro_v8.0.pdf HTTP/1.1\r\n
  Host: www-net.cs.umass.edu\r\n
  Connection: keep-alive\r\n
  Cache-Control: max-age=0\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7,tr;q=0.6\r\n
  If-None-Match: "175fc1-5a741f8bef00d"\r\n
  If-Modified-Since: Thu, 04 Jun 2020 13:17:51 GMT\r\n
  \r\n
  [Full request URI: http://www-net.cs.umass.edu/wireshark-labs/Wireshark_Intro_v8.0.pdf]
  [HTTP request 1/1]
  [Response in frame: 76]
```

The packet Bytes Panel



```
0000  20 b0 01 b3 f9 6a f8 59  71 85 2e ab 08 00 45 00    ····j·Y q·.···E·
0010  02 98 14 59 40 00 40 06  ec 2b c0 a8 01 af 80 77   ···Y@·@· ·+·····w
0020  f5 0c 9e 04 00 50 d1 66  f4 9e fa c4 72 cf 80 18   ·····P·f ····r···
0030  00 e5 32 8e 00 00 01 01  08 0a 00 1d 0a 99 32 61   ··2····· ······2a
0040  d8 f7 47 45 54 20 2f 77  69 72 65 73 68 61 72 6b   ··GET /w ireshark
0050  2d 6c 61 62 73 2f 57 69  72 65 73 68 61 72 6b 5f   -labs/Wi reshark_
0060  49 6e 74 72 6f 5f 76 38  2e 30 2e 70 64 66 20 48   Intro_v8 .0.pdf H
0070  54 54 50 2f 31 2e 31 0d  0a 48 6f 73 74 3a 20 77   TTP/1.1· ·Host: w
0080  77 77 2d 6e 65 74 2e 63  73 2e 75 6d 61 73 73 2e   ww-net.c s.umass.
0090  65 64 75 0d 0a 43 6f 6e  6e 65 63 74 69 6f 6e 3a   edu··Con nection:
00a0  20 6b 65 65 70 2d 61 6c  69 76 65 0d 0a 43 61 63    keep-al ive··Cac
00b0  68 65 2d 43 6f 6e 74 72  6f 6c 3a 20 6d 61 78 2d   he-Contr ol: max-
00c0  61 67 65 3d 30 0d 0a 55  70 67 72 61 64 65 2d 49   age=0··U pgrade-I
00d0  6e 73 65 63 75 72 65 2d  52 65 71 75 65 73 74 73   nsecure- Requests
00e0  3a 20 31 0d 0a 55 73 65  72 2d 41 67 65 6e 74 3a   : 1··Use r-Agent:
```

Shows the dump of the packet

- Shown in hexadecimal or binary

More than one page may available in case wireshark reassembled more than one packet together

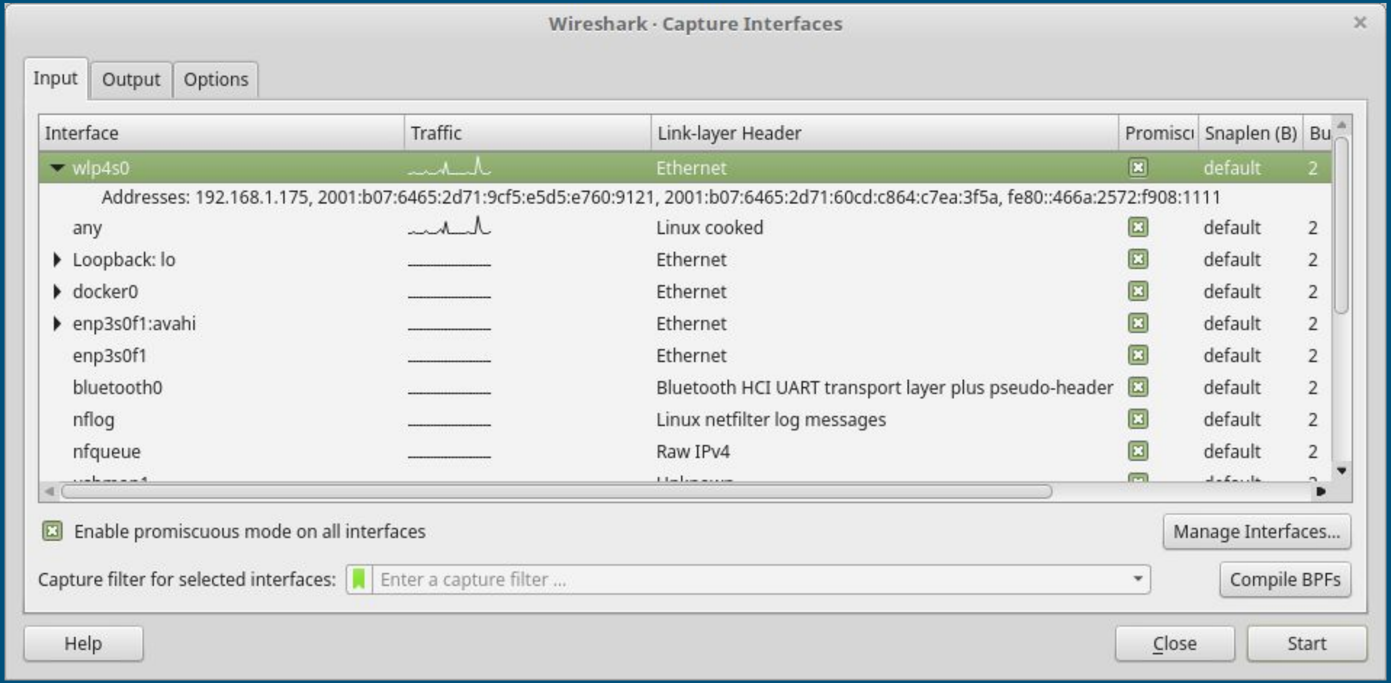
Capturing Live Network Data

Capturing live network data is one of the core components of wireshark

- Can capture from different network interfaces
- Triggers to stop capturing data (elapsed time, number of packets..)
- Live show of packet details
- Live filtering of packets
- Save packets
- Can simultaneously capture from different network interfaces

Start a new capture

Capture > Options...



Wireshark · Capture Interfaces

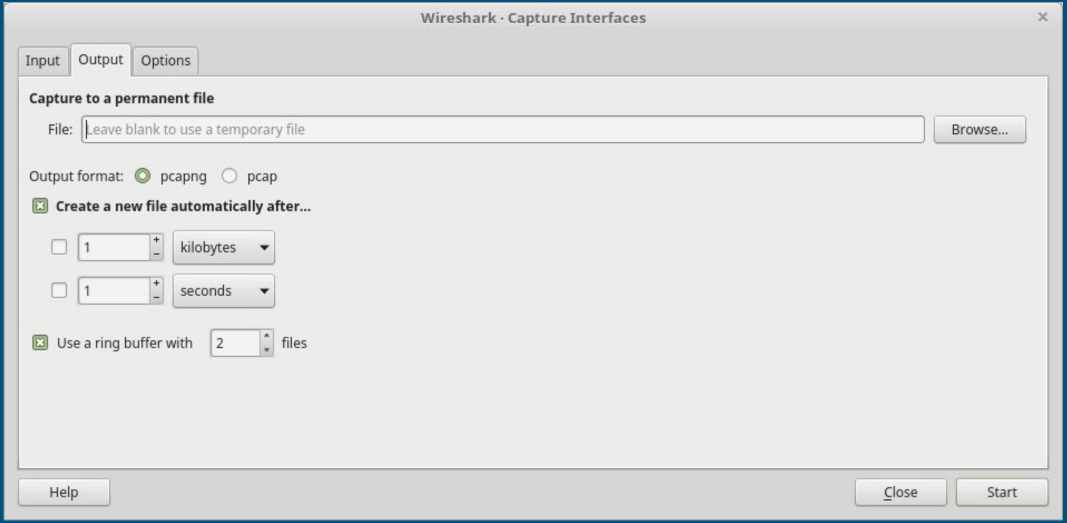
Interface	Traffic	Link-layer Header	Promisci	Snaplen (B)	Bu
wlp4s0		Ethernet	☒	default	2
Addresses: 192.168.1.175, 2001:b07:6465:2d71:9cf5:e5d5:e760:9121, 2001:b07:6465:2d71:60cd:c864:c7ea:3f5a, fe80::466a:2572:f908:1111					
any		Linux cooked	☒	default	2
Loopback: lo		Ethernet	☒	default	2
docker0		Ethernet	☒	default	2
enp3s0f1:avahi		Ethernet	☒	default	2
enp3s0f1		Ethernet	☒	default	2
bluetooth0		Bluetooth HCI UART transport layer plus pseudo-header	☒	default	2
nflog		Linux netfilter log messages	☒	default	2
nfqueue		Raw IPv4	☒	default	2

☒ Enable promiscuous mode on all interfaces

Capture filter for selected interfaces: Enter a capture filter ...

Capture Options

It is possible to save a capture to a file (or multiple files). Consider using this feature if you plan to work with a heavily congested network or if you plan to perform a long-term capture



Capture to a permanent file

File: Leave blank to use a temporary file

Output format: ● pcapng ○ pcap

☒ Create a new file automatically after...

☐ 1 kilobytes

☐ 1 seconds

☒ Use a ring buffer with 2 files

1. No filename → Temporary file
   - (wiresharkXXXX)
2. Filename → Fixed file
   - (overwritten every time)
3. Filename + auto → More files, every time a new one
   - (foo_00001_20100205110102.cap, _00001_20100205110337.cap, ...)
4. Filename + auto + ring buffer → Files get replaced from the beginning after buffer is full
   - (foo_00001_20100205110102.cap, _00001_20100205110337.cap, ...)

Let’s start with the practice

1. Use your own PC and let’s take a look at what we see

Let’s start with the practice

1. Use your own PC and let’s take a look at what we see

Pretty messy huh? There’s so much traffic going on without us being fully aware… even when the machine is idle. Let’s see if we can cut this to a minimum.

2. Use a Virtual Machine (several network options, let’s try with NAT).

Let’s start with the practice

1. Use your own PC and let’s take a look at what we see

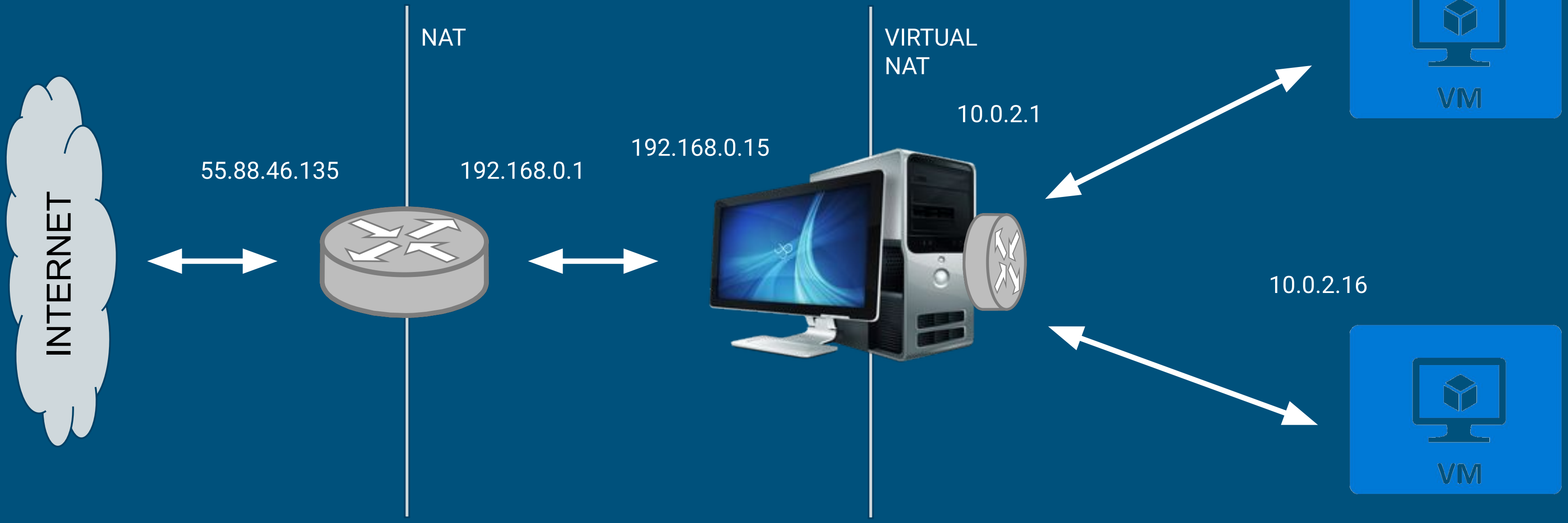
Pretty messy huh? There’s so much traffic going on without us being fully aware… even when the machine is idle. Let’s see if we can cut this to a minimum.

2. Use a Virtual Machine (several network options, let’s try with NAT).

That’s good, however it is more difficult to see what the other hosts are doing, because our VM is cut out from the world.

3. Use several virtual machines connected to the same NAT Network!

Little off topic



You can take a look on how to do it quite easily with Virtual Box, check here: https://www.dedoimedo.com/computers/virtualbox-nat-networks.html

Wireshark Filters

Wireshark empowers the user with powerful filtering expressions:

- Can filter based on the content of a packet
- Can filter based on the IP of a packet
- Can filter based on the protocol of a packet
- Many others...
- It supports comparison and boolean operators

- **Capture Filters**: set before capturing (much like IPTABLES)
- **Display Filters**: set while capturing (only a “visualization filter”)

Capture Filters

Wireshark uses the libpcap filter language. The general syntax is as follows:

```
[not] primitive [and|or [not] primitive...]
```

Example: `tcp port 23 and host 10.0.0.5`

Capture filters are different from Display filters (they use a different syntax)

`Primitives:`

- [src/dst] host <host>
- ether [src/dst] host <host>
- gateway host <host>
- [src|dst] net <net> [{mask <mask>}|{len <len>}]
- [tcp|udp] [src|dst] port <port>
- less|greater <length>
- ip|ether proto <protocol>
- ether|ip broadcast|multicast
- <expr> relop <expr>

More at
https://www.wireshark.org/docs/wsug_html_chunked/ChCapCaptureFilterSection.html

Display Filters - Comparison Operators

Operator	Description	Example
==	Equal	ip.src == 192.168.1.1
!=	Not equal	ip.src != 192.168.1.1
> < >= <=	Greater/Less than	frame.len >10
contains	Field contains a value	sip.To contains “a1762”
matches	Field matches a regexp	http.host matches “acme\.(org|com|net)”
&	Bitwise AND	tcp.flags & 0x02

Display Filters - Combination Operators

Operator	Description	Example
and	Logical AND	ip.src == 192.168.1.1 and tcp.flags.fin
or	Logical OR	ip.src == 192.168.1.1 or tcp.flags.fin
not	Logical NOT	not llc
xor	Logical XOR	ip.dst == 10.0.6.29 xor ip.src == 10.0.6.29
[...]	Slice Operator (see next)	eth.src[0:3] == 00:00:83
in	Membership (see next)	tcp.port in {80 443 8080}

Display Filters - Slice Operator

Used to select subsequences of a sequence

Simply put brackets after a label:

- eth.src[0:3] == 00:00:83
- eth.src[1-2] == 00:83
- eth.src[:4] != 00:83:45:21
- eth.src[4:] != 00:83
- eth.src[4] != 00
- eth.src[0:3,1-2,:4,4:,2] == 00:00:83:00:83:00:00:83:00:20:20:83

: expects a length, - expects the end

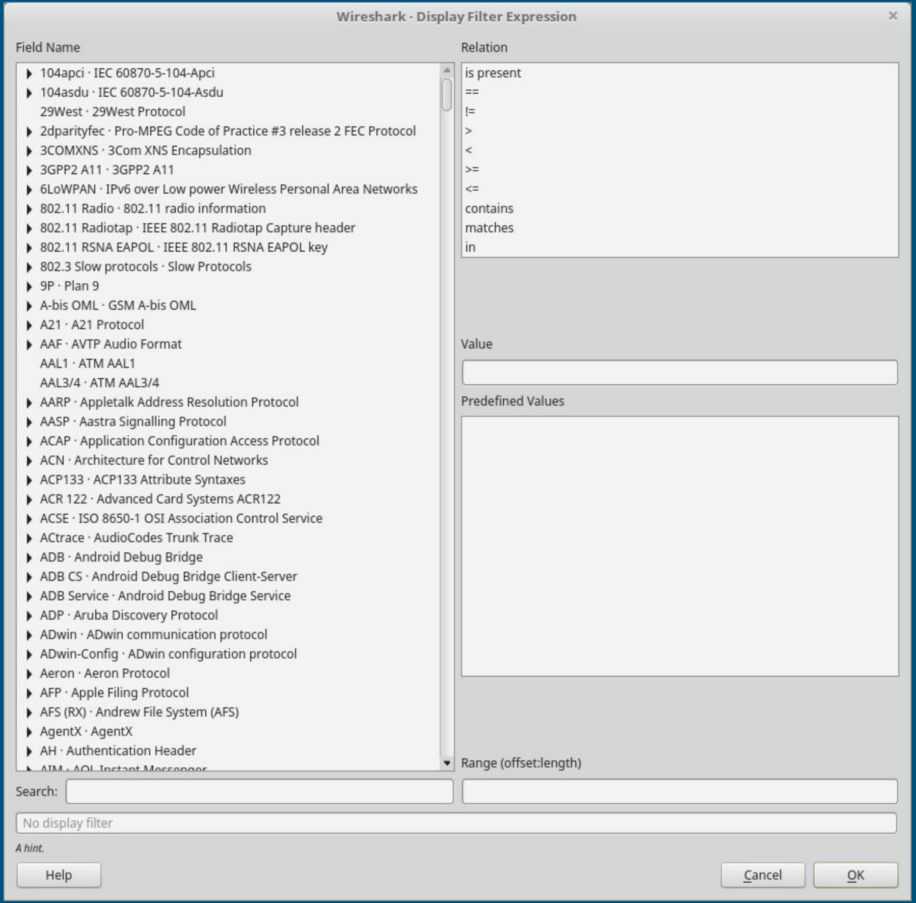
Display Filters - Membership Operator

Used to test a field against a set of values, simply use in after a label and put the set inside {}:

- tcp.port in {80 443 8080}
    - This is equal to tcp.port == 80 or tcp.port == 443 or tcp.port == 8080
- tcp.port in {443 4430..4434}
    - This is equal to tcp.port == 443 or (tcp.port >= 4430 and tcp.port <= 4434)
- http.request.method in {“HEAD” “GET”}
- ip.addr in {10.0.0.5 .. 10.0.0.9 192.168.1.1 .. 192.168.1.9}
- frame.time_delta in {10 .. 10.5}
- Wireshark also offers simple functions that can be helpful when dealing with packet content: upper/lower and len/count

More on Display Filters

- Once you get used to WireShark, you will use rarely the dialog box
- At the beginning, it is an valuable tool to learn about WireShark display strings
- It has a search function which makes easier to navigate through all the possible fields
- Also possible to define relations between fields and labels
- Finally, it is also possible to save filters for later use



- 104apci · IEC 60870-5-104-Apci
- 104asdu · IEC 60870-5-104-Asdu
- 29West · 29West Protocol
- 2dparityfec · Pro-MPEG Code of Practice #3 release 2 FEC Protocol
- 3COMXNS · 3Com XNS Encapsulation
- 3GPP2 A11 · 3GPP2 A11
- 6LoWPAN · IPv6 over Low power Wireless Personal Area Networks
- 802.11 Radio · 802.11 radio information
- 802.11 Radiotap · IEEE 802.11 Radiotap Capture header
- 802.11 RSNA EAPOL · IEEE 802.11 RSNA EAPOL key
- 802.3 Slow protocols · Slow Protocols
- 9P · Plan 9
- A-bis OML · GSM A-bis OML
- A21 · A21 Protocol
- AAF · AVTP Audio Format
- AAL1 · ATM AAL1
- AAL3/4 · ATM AAL3/4
- AARP · Appletalk Address Resolution Protocol
- AASP · Aastra Signalling Protocol
- ACAP · Application Configuration Access Protocol
- ACN · Architecture for Control Networks
- ACP133 · ACP133 Attribute Syntaxes
- ACR 122 · Advanced Card Systems ACR122
- ACSE · ISO 8650-1 OSI Association Control Service
- ACtrace · AudioCodes Trunk Trace
- ADB · Android Debug Bridge
- ADB CS · Android Debug Bridge Client-Server
- ADB Service · Android Debug Bridge Service
- ADP · Aruba Discovery Protocol
- ADwin · ADwin communication protocol
- ADwin-Config · ADwin configuration protocol
- Aeron · Aeron Protocol
- AFP · Apple Filing Protocol
- AFS (RX) · Andrew File System (AFS)
- AgentX · AgentX
- AH · Authentication Header
- AIM · AOL Instant Messenger

- is present
- ==
- !=
- >
- <
- >=
- <=
- contains
- matches
- in

Example on Capture Filters

`host 192.168.1.200` - Capture only packets coming from or going to host 192.168.1.200

`ether host 00:00:5e:00:53:00` - Get all packets with source or destination MAC address equal to 00:00:5e:00:53:00

`host google.it` - Get all the packets coming from or going to host www.google.it

`not broadcast and not multicast` - Do not capture packets which are either broadcast or multicast

Example on Display Filters

`tcp.port == 80` - Display all packets with the port equals to 80 (default HTTP)

`tcp.port == 80 or tcp.port == 443` - Display all packets with the port equals to 80 (default HTTP) or 443 (default HTTPS). Same as `tcp.port in { 80 443 }`

`tcp.dstport == 80 and (tcp.srcport > 60000 and tcp.srcport < 64000)` - Display all packets directed to port 80 coming from a port within 60000 and 64000

Reverse Examples

You are seeing an unusual http traffic from IP 192.168.1.200, as it makes a lot of request to www.iamnotasafesite.danger. You are afraid that all its subnetwork (255.255.255.0) may be compromised. You want to identify all the hosts that are possibly compromised, what do you write?

Reverse Examples

You are seeing an unusual http traffic from IP 192.168.1.200, as it makes a lot of request to www.iamnotasafesite.danger. You are afraid that all its subnetwork (255.255.255.0) may be compromised. You want to identify all the hosts that are possibly compromised, what do you write?

- Capture: `src net 192.168.1.0/24`
- Display: `http.host www.iamnotasafesite.danger`

Reverse Examples

After analysing the previous attack, you discover that there are many sites using the domain .danger that may be harmful for your hosts. You also discover that there is a page, *nowillstealallyourpersonalbelongings.html*, which is the root cause of the problems. Identify all the hosts that visit such page.

Reverse Examples

After analysing the previous attack, you discover that there are many sites using the domain .danger that may be harmful for your hosts. You also discover that there is a page, *nowillstealallyourpersonalbelongings.html*, which is the root cause of the problems. Identify all the hosts that visit such page.

- Capture: `src net 192.168.1.0/24`
- Display: `http.host contains ".danger" and frame contains "nowillstealallyourpersonalbelongings.html"`

Exercise 0

- Open Wireshark
- Start a capture on at least the network interface through which you are connected to the internet
    - you can try through capture filters to only get IPv4 packets
- Go to www.google.com
- Stop the capture
- How many packets you see?
- Are all of them originated by you?
- How to find the packets related to your last internet search?
    - filter “http” packets… do they help out?

Exercise 0 (cont’d)

- Open Wireshark

- Let’s test the lower layer stuff
- Try to ping 8.8.8.8
  - Observe the ICMP packet, what’s inside at layers 3 and 2…
- Try to ping something in your local network (e.g. your router).
  - Any ARP packets coming along? If not, why? What do they mean?
- Try to force DHCP to give you another address (usually dhclient on linux).
  - How is the negotiation taking place?

Exercise 1

We will carry out this exercise concerning ethernet

- We are at ISO/OSI Level 2
- We need the **ethernet--ethereal-trace-1** from the wireshark-traces.zip file
- We will also investigate ARP related messages

Exercise 1

- Network interfaces have a unique address, called MAC
- Part of it identify the manufacturer, the rest is a progressive counter
- Computer receive messages through their MAC
- Through a protocol (ARP) it is possible to uniquely identify a computer on a network
- ARP builds its table dynamically
- How many mac addresses? With a 48 bit addressing space, 281,474,976,710,656

Exercise 1

Open the **ethernet-ethereal-trace-1** file look into HTTP part

- What is the ethernet address of your computer?
- What is the ethernet address of the destination?
- Is it the MAC address of gaia.cs.umass.edu?
- What device has this as ist Ethernet address?
- What is the hexadecimal value for the 2-byte Frame type field for IP?
- How many bytes from the very start of the Ethernet frame does the ASCII “G” in “GET” appear in the Ethernet frame?
- What is the source Ethernet address in the HTTP OK answer?
- Is it the address of gaia.cs.umass.edu?
- What is the destination address in the HTTP OK answer?

Exercise 1 (ans)

Open the **ethernet-ethereal-trace-1** file look into HTTP part

- What is the ethernet address of your computer? 00:d0:59:a9:3d:68
- What is the ethernet address of the destination? 00:06:25:da:af:73
- Is it the address of gaia.cs.umass.edu? No
- What device has this as ist Ethernet address? The router (Linksys)
- What is the hexadecimal value for the 2-byte Frame type field for IP? 0x0800
- How many bytes from the very start of the Ethernet frame does the ASCII “G” in “GET” appear in the Ethernet frame? 54 B. (14B Ethernet, 20B IP, 20B TCP)
- What is the source Ethernet address in the HTTP OK answer? 00:06:25:da:af:73
- Is it the address of gaia.cs.umass.edu? No
- What is the destination address in the HTTP OK answer? 00:d0:59:a9:3d:68

Exercise 1

Open the **ethernet-ethereal-trace-1** file look into the ARP part

- What are the source/destination addresses of the ARP request?
- What is the hexadecimal value for the 2-byte Frame type field?
- Does the ARP message contains the sender IP (if yes, what it is?)?
- In the ARP reply, what are the Ethernet and IP addresses of the machine having the Ethernet address whose corresponding IP address is being queried?

Exercise 1 (ans)

Open the **ethernet-ethereal-trace-1** file look into the ARP part

- What are the source/destination addresses of the ARP request? 00:d0:59:a9:3d:68 and ff:ff:ff:ff:ff:ff
- What is the hexadecimal value for the 2-byte Frame type field? 0x0806 (ARP)
- Does the ARP message contains the sender IP (if yes, what it is?)? Yes, 192.168.1.105
- In the ARP reply, what are the Ethernet and IP addresses of the machine having the Ethernet address whose corresponding IP address is being queried? 00:06:25:da:af:73 - 192.168.1.1.