

**INIZIO LEZIONE 9:10**

# Network Sniffing

↳ strumento che analizza i messaggi scambiati su una rete.

Le reti di base non hanno tante protezioni.

Es: nessun elemento che garantisce le parti che comunicano. Es: la macchina sorgente può scrivere un indirizzo diverso. I protocolli di rete li implementa il sistema operativo.

## Corso di reti di Calcolatori e Cybersecurity

Dov'è codice che implementa IP? Come lo fa?  
Dove lo implementa?

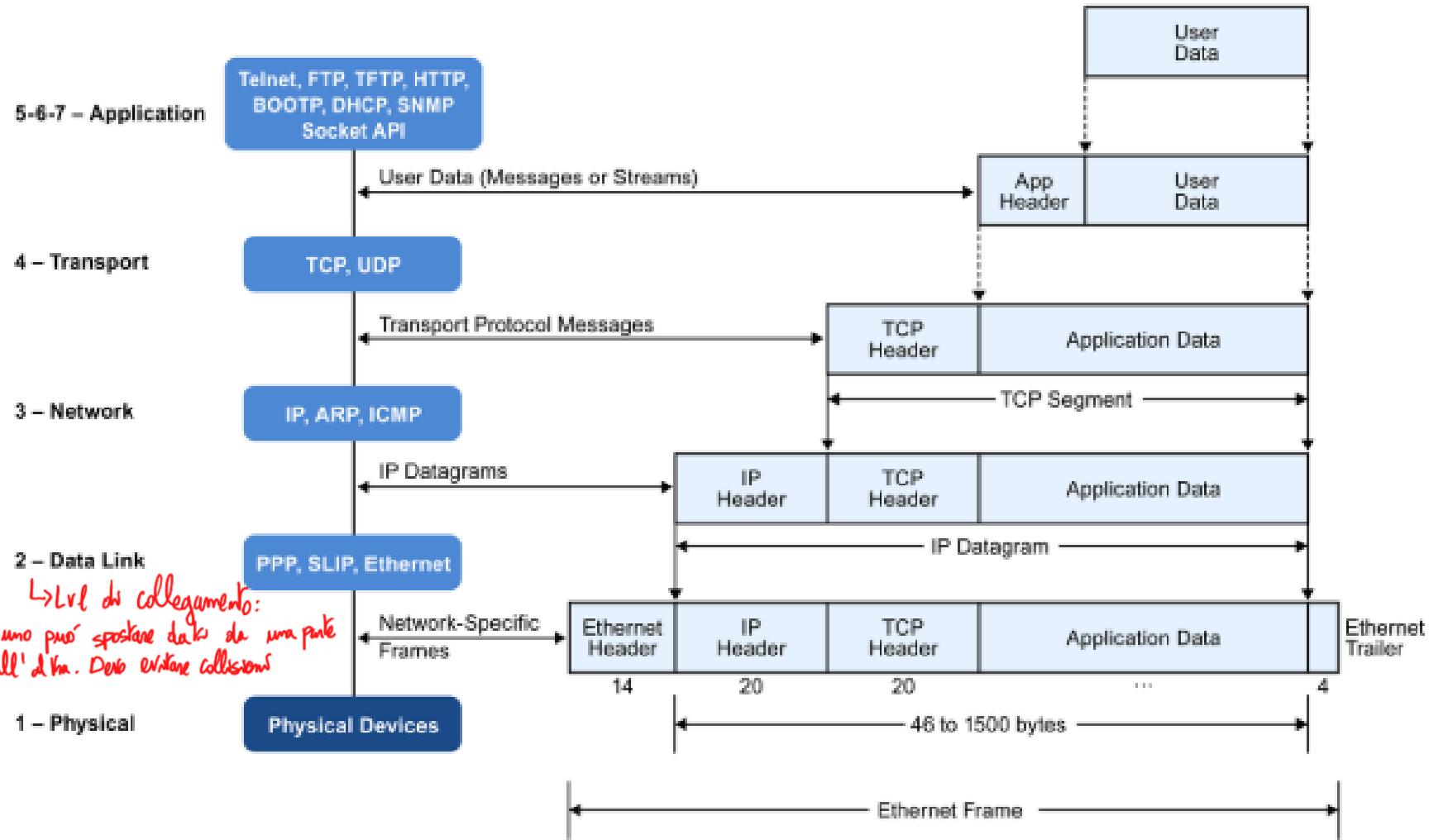
<date>  
Location



Università  
degli Studi  
della Campania  
*Luigi Vanvitelli*

# TCP/IP Network Stack

Ogni livello risolve problemi diversi.



BASE: Ho pacchetti formati da Header e Payload

Header porta info per spedire messaggi a livello collegato: indirizzo fisico della macchina.

RICORDA: Switch ha diversi livelli di intelligenza.

LVL RETE: collega macchine su reti diverse. Protocolli di routing sono il problema.

Usa servizi del livello sottostante. Se uso ethernet uso header per ethernet, se wifi per wifi.

Lui mi fa arrivare fino alla rete di destinazione. Poi avrò bisogno di un'associazione indirizzo IP/indirizzo MAC

Nota: se ho 2 schede di rete, ho 2 indirizzi MAC e quindi 2 IP.

- Per l'associazione uso ARP; visto che non so MAC scrivo un broadcast e risponde la macchina corretta. Ma non ho controllo per capire se lui è il vero, ARP non controlla.

NOTA: ARP si trova a livello di rete. Senza ARP, IP non può funzionare.

ICMP per errori e messaggi di configurazione. Tutti questi 3 protocolli hanno O nel CIA.

## TRASPORTO:

- So la macchina, ma non l'applicazione. TCP aggiunge anche affidabilità.

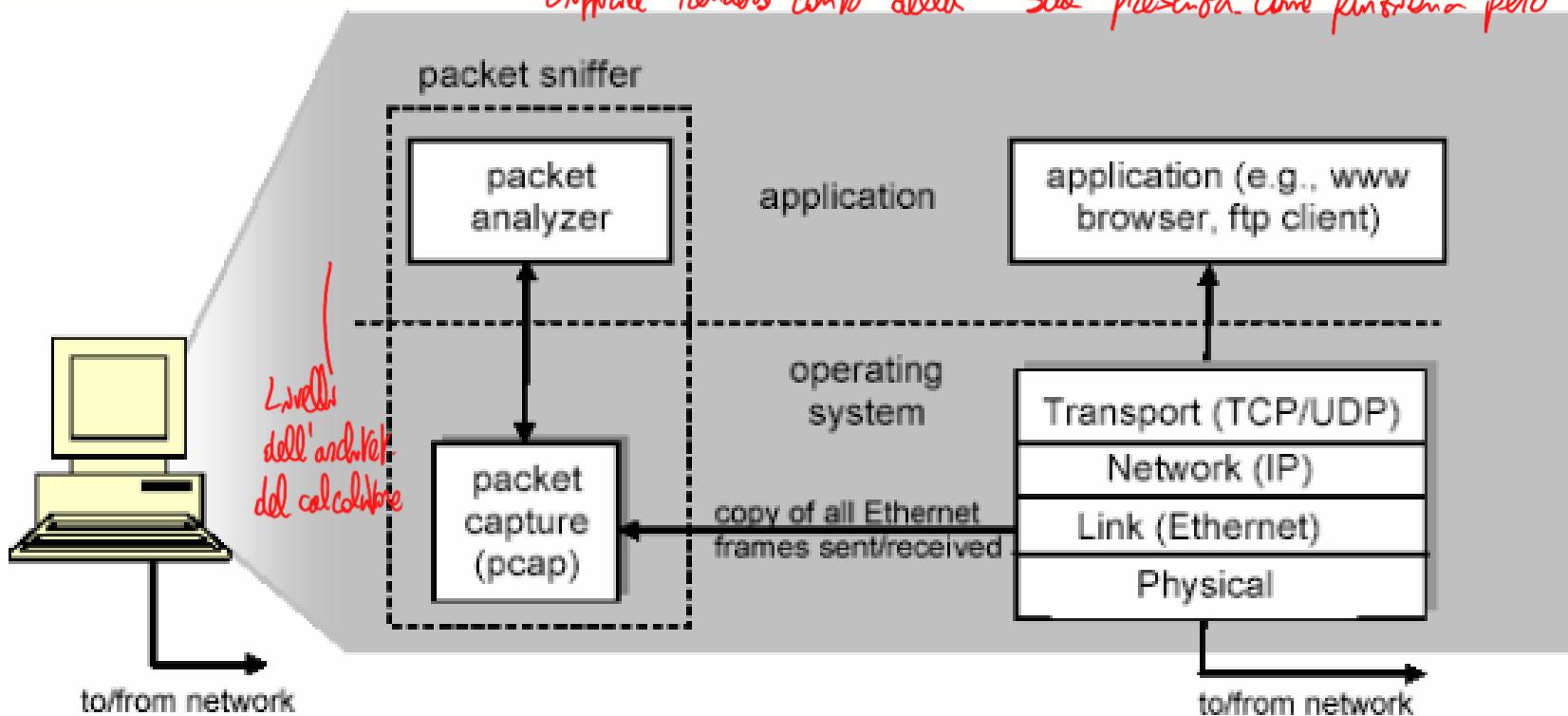
UDP è datagram oriented (pacchetti indipendenti). TCP è connection oriented.

Un protocollo di rete definisce l'insieme di regole per la comunicazione tra le parti interessate.

- Quali sono i tipi di punti che comunicano fra loro? (TCP: endpoint, IP terminali/mati di rete)
- Quali sono i tipi di messaggi che vengono spediti?
- Quali sono i formati dei messaggi coinvolti? (es. header http seguito da spazio e body ecc.)
- Quali sono le regole di comunicazione?

# Packet Sniffer

- Packet sniffer is a basic tool for observing network packet exchanges in a computer.
- A packet sniffer itself is passive. *Non altera niente. Legge e basta.  
Difficile renders conto della sua presenza. Come funziona però?*



- L'utile applicazione in rete non è dello stesso genere di un'applicazione su un'hardware (es. hardware di stampanti).

NOTA: Pacchetti di rete è quello a livello 2, implementato da scheda di rete.

Ma solo l'OS parla con la scheda di rete.

Ottimale packet capturing lo può fare solo l'OS. Ma servono librerie adattate.

Pcap è la più frequente.

DOMANDA: Sei connesso ad wifi (scheda di rete wifi). Riceve solo i pacchetti dedicati a sé?

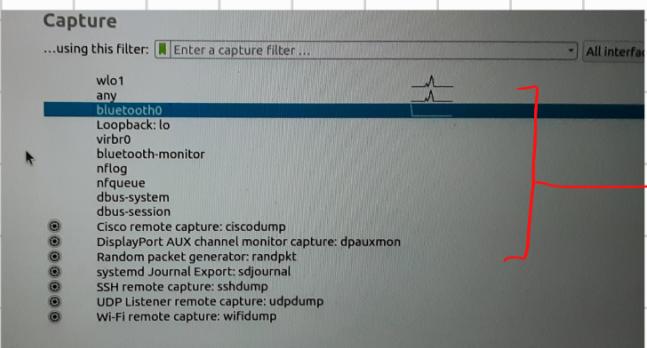
No. Riceve i pacchetti e li riceve tutti, perché il wifi è un broadcast. Quindi posso configurare la scheda di rete per ricevere tutti i pacch. Configurazione di monitoring.

Ma solo dallo stesso wifi? Sì, Tutti i messaggi con lo stesso SSID.

NOTA: Questo è solo il capturing. Avrò vari frame ethernet. Non è detto che un singolo messaggio IP entri in un frame ethernet ecc. Posso avere un frammento della rete monitorata.

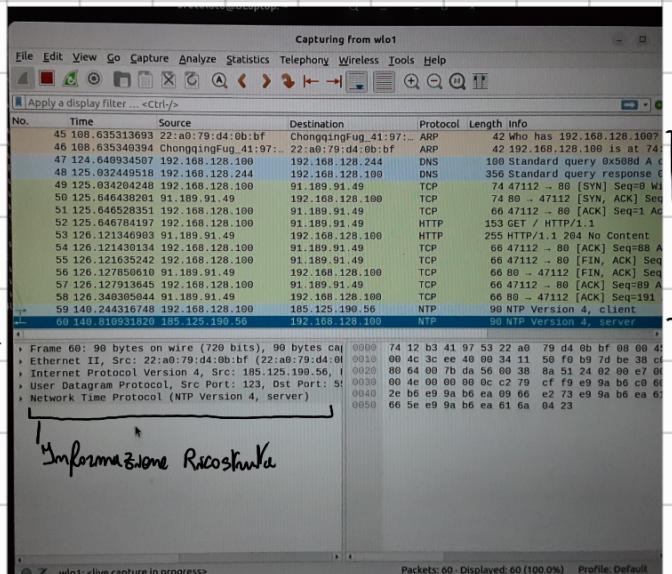
PACKET ANALYZER controlla col packet capture analizzando le cose ricevute.

Riesce a monitorare le sottoparti con tutte le info a tutti i livelli.



interfaccia di rete su cui ci dobbiamo connettere

- Analisi è pensata a livello IP. I dati vengono raccolti. Ogni riga è un pacchetto.
- In basso a destra c'è il pacchetto in focus.



Flusso di messaggi.

Ricorda: dei codici che non corrispondono a corrispondente utile vengono sostituiti da puntini (carattere di controllo).

NOTA: A volte vediamo HTTP a volte no. Come faccio a capire quel è il protocollo a livello applicativo? Dal numero di porta, 80 per HTTP. Se parla su porta differente non lo so e lo devo aggiungere l'informazione.

Right click: follow: TCP Stream: individua tutti i pacchetti che fanno parte dello stesso stream TCP.

Riproduce quindi tutti la sequenza di messaggi. Tutti i pacchetti interi, non solo header.

Rosso e blu sono i due flussi.

Nota: wireshark per vedere solo messaggi di quel flusso.

## BRIDGE DI RETE:



Tutti i pacchetti della rete 1 bridge li prende  
e li mette nella rete 2 e viceversa.

Possibile problema sui broadcast.

Ma se ho 3 reti?



Messaggi girano all'infinito?

NOTA: Recupero IP da NeverSSL

Faccio un ping a neverssl.com.

Trovò IP e visto sopra: ip\_addr == <ip address>

Oppure filtro gli scambi dns effettuati. Poi ho pensato che magari lo ha in cache.

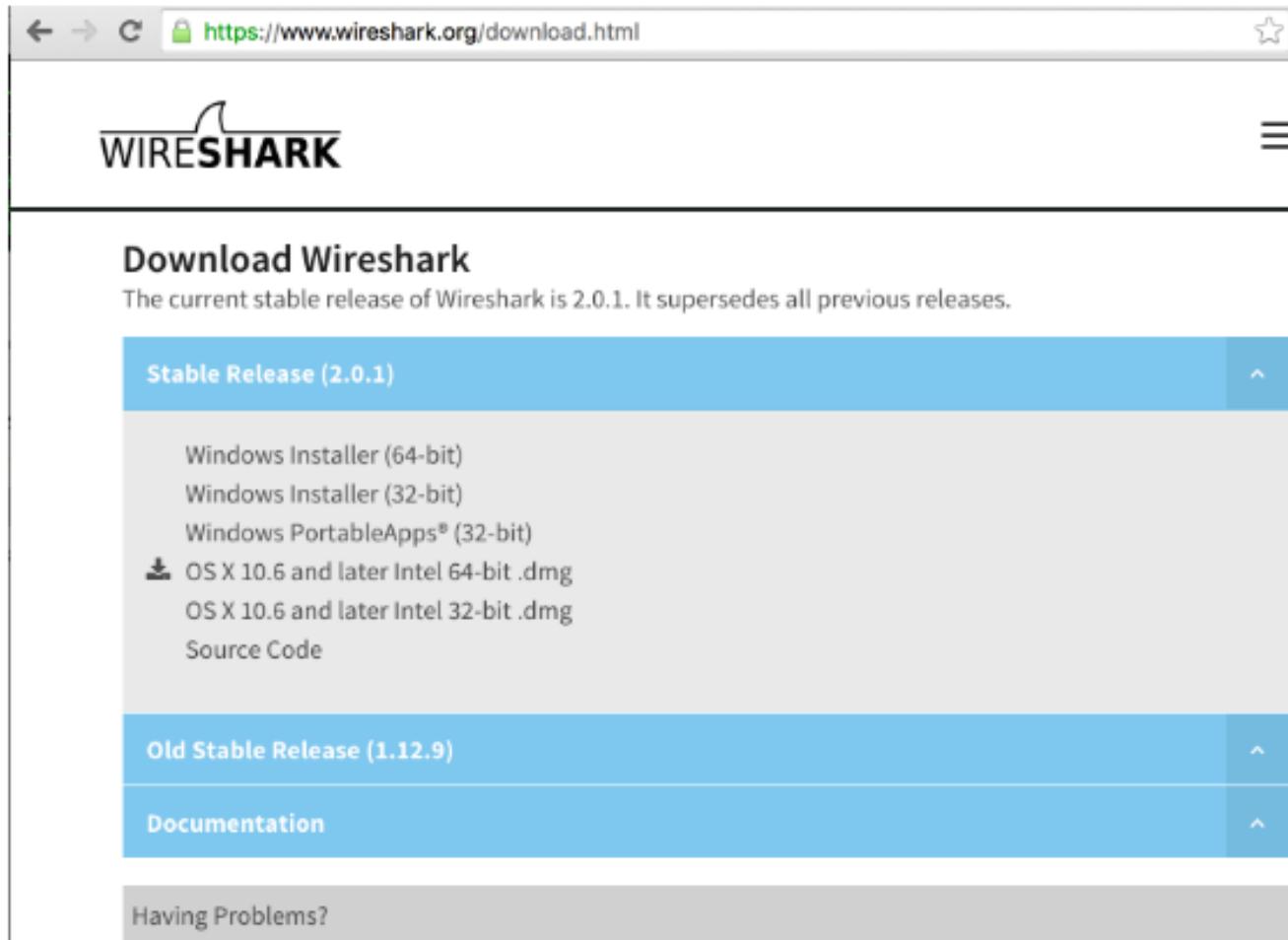
Allora cerco direttamente gli scambi http. Provo a fare follow.

NOTA: nell'header HTTP c'è campo host (nome del host).

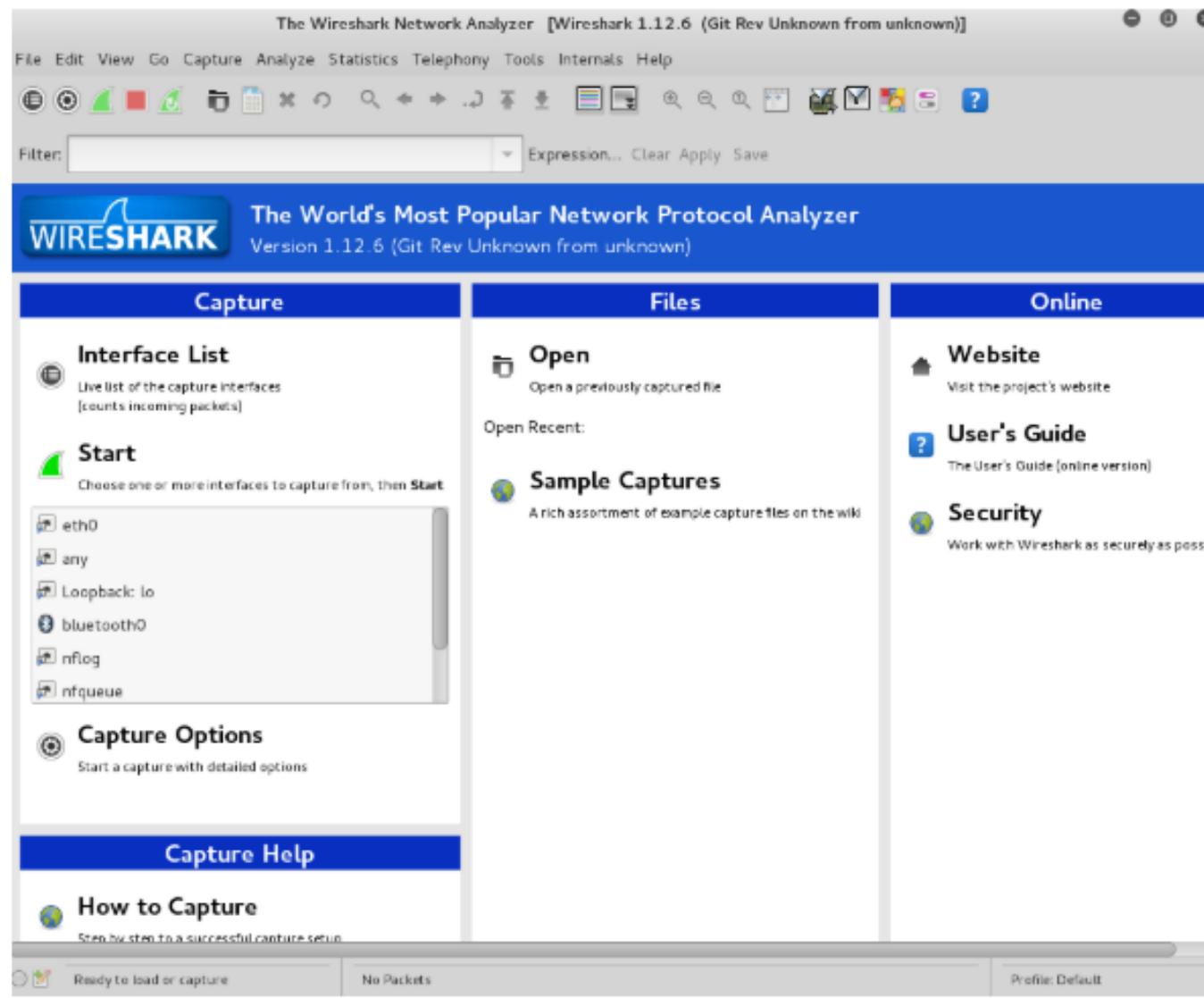
# WireShark -Getting Wireshark

---

- <https://www.wireshark.org/download.html>



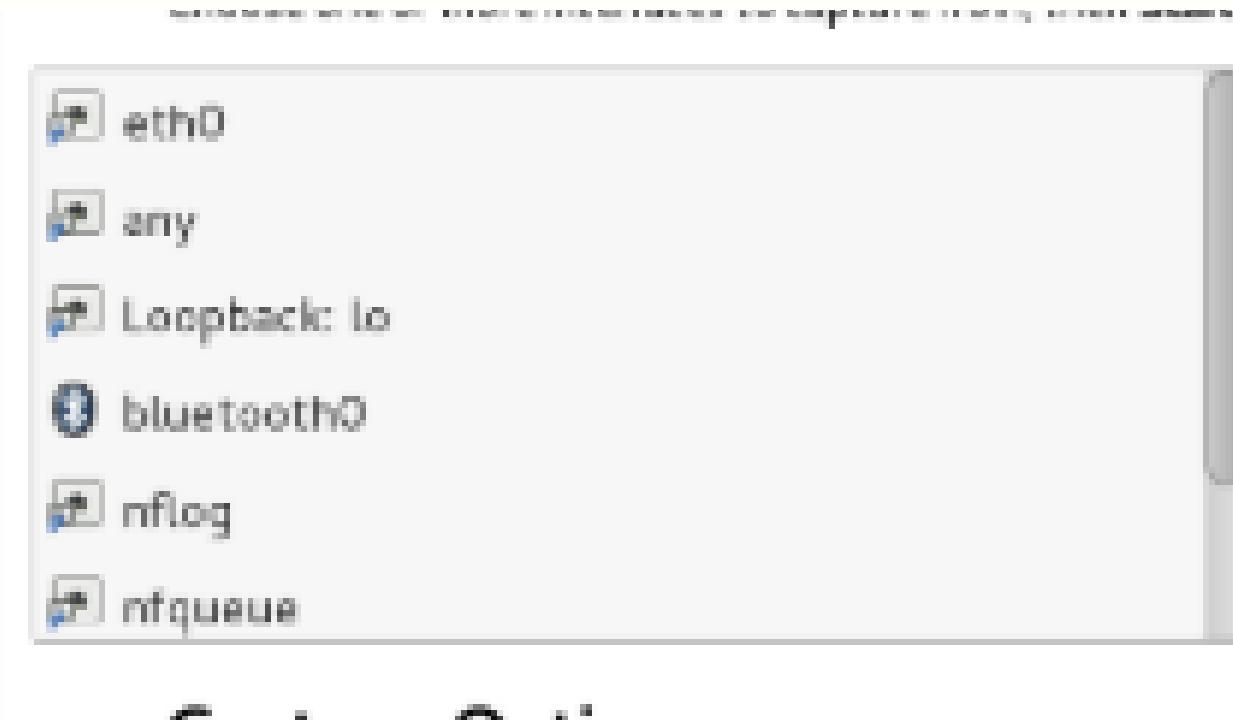
# Wireshark - Start



Versioni diverse  
Interfacce leggermente diverse....

# Wireshark - Start

---



Scegliere  
Interfaccia  
(NIC)

# Wireshark - Interfaccia

Capturing from eth0 [Wireshark 1.12.6 (Git Rev Unknown from unknown)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.108.151	172.16.108.2	DNS	72	Standard query 0x28c5 A www.kali.org
2	0.000003000	172.16.108.151	172.16.108.2	DNS	72	Standard query 0x630c AAAA www.kali.org
3	0.000201000	172.16.108.151	172.16.108.2	DNS	74	Standard query 0xbd4f A tools.kali.org
4	0.000350000	172.16.108.151	172.16.108.2	DNS	74	Standard query 0x3e4d AAAA tools.kali.org
5	0.001283000	172.16.108.151	172.16.108.2	DNS	86	Standard query 0x3dbb A www.offensive-security.com

Frame 4: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0  
Ethernet II, Src: VMware\_6d:7a:b5 (00:0c:29:6d:7a:b5), Dst: VMware\_f0:1a:b5 (00:50:56:f0:1a:b5)  
Internet Protocol Version 4, Src: 172.16.108.151 (172.16.108.151), Dst: 172.16.108.2 (172.16.108.2)  
User Datagram Protocol, Src Port: 46328 (46328), Dst Port: 53 (53)  
Domain Name System (query)

0000 00 50 56 6d 7a b5 00 0c 29 6d 7a 35 08 00 45 00 .PV.....Imz5..E.  
0010 00 3c 37 6f 40 00 40 11 d2 87 ac 10 6c 97 ac 10 <7e0.8. ....1...  
0020 6c 02 b4 f8 00 35 00 28 63 53 2e a0 01 00 00 01 1....5.| c8.....  
0030 00 00 00 00 00 00 05 74 6f 6f 6c 73 04 6b 61 6c .....t ols.kal  
0040 69 03 6f 72 67 00 00 1c 00 01 1.prd... .

eth0: <live capture in progress> File / It... Packets: 42 - Displayed: 42 (100.0%) Profile: Default

# Wireshark - Interfaccia

The screenshot shows the Wireshark interface with several annotations:

- command menus**: Points to the top menu bar.
- display filter specification**: Points to the "Display" filter input field.
- listing of captured packets**: Points to the main packet list table.
- details of selected packet header**: Points to the detailed packet header information for the selected packet.
- packet content in hexadecimal and ASCII**: Points to the bottom panes showing the raw hex and ASCII representations of the selected packet.

**Packet List (Selected Packet)**

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.46	128.121.50.122	TCP	1183 > http [SYN] Seq=0 Len=0 MSS=1460
2	0.127987	128.121.50.122	192.168.1.46	TCP	Http > 1183 [SYN, ACK] Seq=1 Len=57
3	0.128732	192.168.1.46	128.121.50.122	TCP	1183 > http [ACK] Seq=1 Len=57
4	0.131709	192.168.1.46	128.121.50.122	HTTP	GET /news/ HTTP/1.1
5	0.329841	128.121.50.122	192.168.1.46	TCP	[TCP segment of a reassembled PDU]
6	0.330326	192.168.1.46	192.168.1.46	HTTP	[TCP Previous segment lost] continuation
7	0.330487	192.168.1.46	128.121.50.122	TCP	1183 > http [ACK] Seq=517 Len=54
8	0.342042	128.121.50.122	192.168.1.46	TCP	[TCP Resynchronization] [TCP segment of a reassembled PDU]
9	0.342052	192.168.1.46	128.121.50.122	TCP	1183 > http [ACK] Seq=517 Len=54

**Selected Packet Details**

Frame 4 (710 bytes on wire, 710 bytes captured)  
Ethernet II, Src: Netgear\_61:8e:6d (00:09:5b:61:8e:6d), Dst: westelltr\_9f:92:b9 (00:0f:ab:9f:92:b9)  
Internet Protocol Version 4, Src: 192.168.1.46 (192.168.1.46), Dst: 128.121.50.122 (128.121.50.122)  
Transmission Control Protocol, Src Port: 1183 (filed), Dst Port: http (80), Seq: 1, Ack: 1, Len: 636  
hypertext Transfer Protocol  
GET /news/ HTTP/1.1\r\nHost: www.wireshark.org\r\nUser-Agent: Mozilla/5.0 (Windows NT 5.1; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4\r\nAccept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,\*/\*;q=0.8,video/ogg;q=0.3,image/png,\*/\*;\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7\r\nKeep-Alive: 300\r\nConnection: keep-alive\r\nReferer: http://www.wireshark.org/faq.html\r\nCookie: \_\_utma=87653150.62471437.1181007382.1181007382.1181007382.1.1.utm\r\n\r\n

**Selected Packet Hex/ASCII**

0000 00 0F 6B 0F 92 B9 00 09 5b 61 B6 C0 08 00 45 00 .... [S..E.  
0010 02 B8 0F 25 40 00 B0 06 74 51 C0 A8 01 46 B0 79 ...[O...TQ...Y  
0020 32 74 04 B8 00 50 9D BC B8 1B 48 C6 F1 B8 50 1B 22...P...H...P.  
0030 FF FF 77 74 00 00 47 49 52 20 2F 06 A1 97 73 0A 0F ....[.0B T /news/  
0040 20 48 54 54 50 2F 31 26 31 0D 04 48 87 73 74 3A HTTP/1.1.Host:  
0050 20 77 77 77 28 77 B9 72 65 73 00 01 72 00 0E 0F www.wireshark.org  
0060 72 67 00 00 73 65 72 20 41 67 65 66 74 0A 20 00 rig. User-Agent:  
0070 4d 61 72 65 66 6C 61 20 35 3A 30 20 2A 57 65 66 mozilla/5.0 (Win  
0080 64 6F 77 73 30 20 19 3B 20 37 69 64 6F 77 73 down; U: windows  
0090 20 46 54 20 05 26 31 3B 20 65 66 20 35 13 30 20 NT 5.1; en-US;  
00A0 72 76 3A 31 32 38 28 31 28 34 29 20 47 65 63 66 rv:1.8.1.4; Gecko  
00B0 0F 2F 32 10 10 17 20 11 31 31 30 46 69 72 61 61 o/20070515 Firefox

# Wireshark - Capture

---

- Avvia Wireshark
  - Seleziona NIC
  - Avvia Capture
- Avvia Browser
  - Vai su  
[www.neverssl.com](http://www.neverssl.com)
- Stop Wireshark
- Esercizio analizza i pacchetti catturati e individua la navigazione effettuata.

# Wireshark - Hint

---

- Codici Colore: i Pacchetti sono evidenziati con colori diversi:
  - Verde TCP, Dark blue DNS, light blue UDP, black TCP con errori
- Filtro
  - Nella barra filtro potete inserire filtri che fanno vedere solo alcuni pacchetti
    - http solo pacchetti HTTP
    - I filtri sono personalizzabili ..

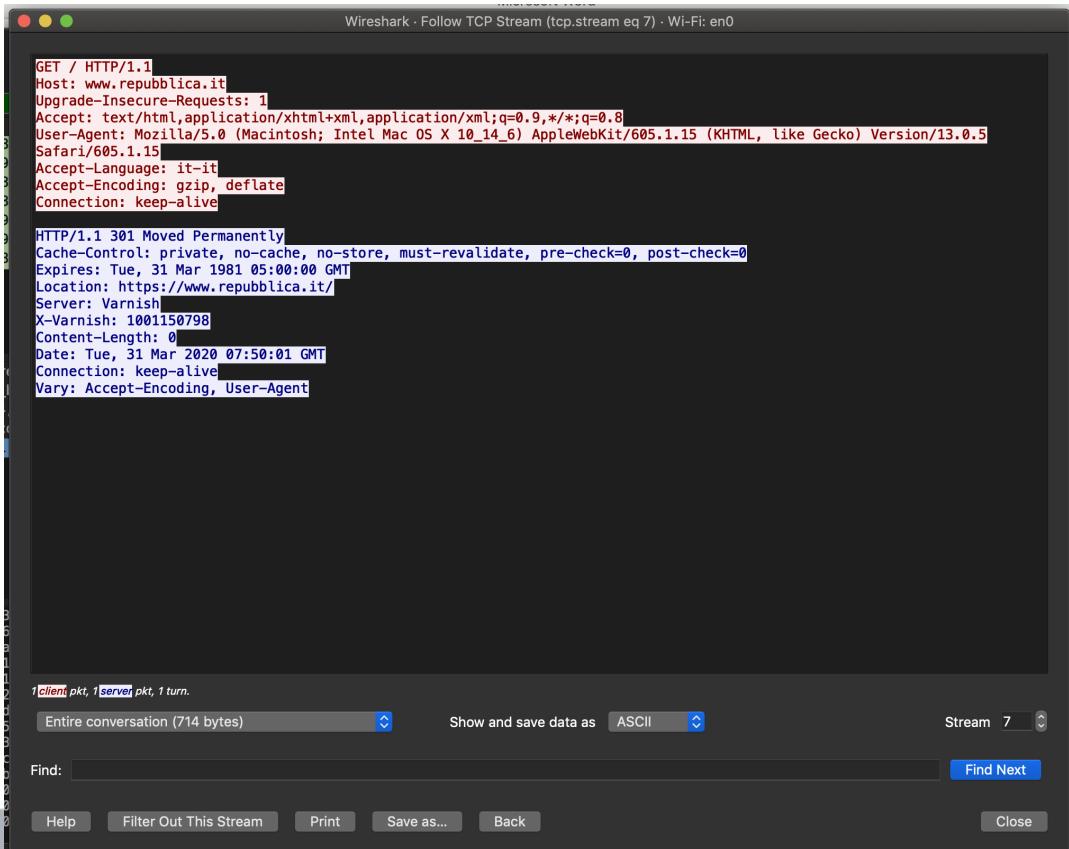
# Wireshark - Hint

---

- Codici Colore: i Pacchetti sono evidenziati con colori diversi:
  - Verde TCP, Dark blue DNS, light blue UDP, black TCP con errori
- Filtro
  - Nella barra filtro potete inserire filtri che fanno vedere solo alcuni pacchetti
    - http solo pacchetti HTTP
    - I filtri sono personalizzabili ..
    - http.host=www.neversll.com trovate solo i pacchetti con host www.neversll.com

# Seguire stream TCP

- http.host==www.repubblica.it
- Selezionate messaggio
- Click su Follow TCP Stream



The screenshot shows a Wireshark window titled "Follow TCP Stream (tcp.stream eq 7) · Wi-Fi: en0". The main pane displays two captured network frames. Frame 1 is a client GET request to www.repubblica.it, and frame 2 is a server HTTP/1.1 301 Moved Permanently response. The response includes headers such as Cache-Control, Expires, Location, Server, X-Varnish, Content-Length, Date, Connection, and Vary. The bottom status bar indicates "1 client pkt, 1 server pkt, 1 turn." and shows options for "Entire conversation (714 bytes)" and "Show and save data as ASCII".

```
GET / HTTP/1.1
Host: www.repubblica.it
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.5
Safari/605.1.15
Accept-Language: it-it
Accept-Encoding: gzip, deflate
Connection: keep-alive

HTTP/1.1 301 Moved Permanently
Cache-Control: private, no-cache, no-store, must-revalidate, pre-check=0, post-check=0
Expires: Tue, 31 Mar 1981 05:00:00 GMT
Location: https://www.repubblica.it/
Server: Varnish
X-Varnish: 1001150798
Content-Length: 0
Date: Tue, 31 Mar 2020 07:50:01 GMT
Connection: keep-alive
Vary: Accept-Encoding, User-Agent

1 client pkt, 1 server pkt, 1 turn.
Entire conversation (714 bytes) Show and save data as ASCII
Find: Find Next
Help Filter Out This Stream Print Save as... Back
Stream 7
```

# Analisi DNS

---

- Filtrate DNS
- Selezionate messaggio DNS
- Click su Follow UDP Stream



# Esercizio

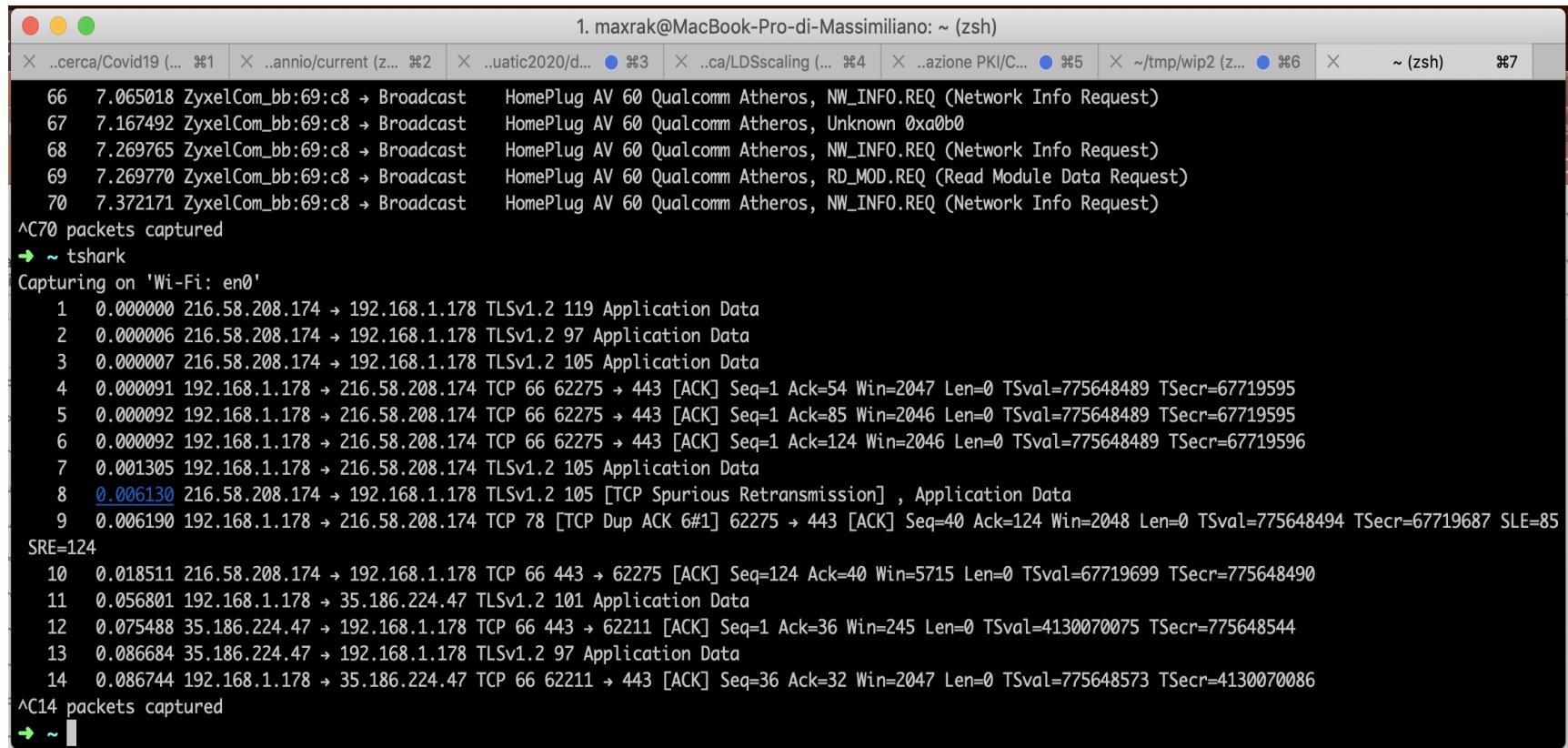
---

- Ricostruire handshake TCP con wireshark

# Tshark

---

- Analizzatore di pacchetti a linea di comando
  - tshark



The screenshot shows a macOS terminal window titled "1. maxrak@MacBook-Pro-di-Massimiliano: ~ (zsh)". The window contains the output of a tshark command capturing traffic on the Wi-Fi interface "en0". The output shows several HomePlug AV frames from a ZyxelCom device, followed by TLSv1.2 application data frames between two hosts. The terminal window has multiple tabs at the top, and the status bar at the bottom shows the number of packets captured.

```
1. maxrak@MacBook-Pro-di-Massimiliano: ~ (zsh)
..cerca/Covid19 (...)  1 | ..anno/current (z...)  2 | ..uatic2020/d...  3 | ..ca/LDSscaling (...)  4 | ..azione PKI/C...  5 | ~/tmp/wip2 (z...  6 | ~ (zsh)  7
66 7.065018 ZyxelCom_bb:69:c8 → Broadcast   HomePlug AV 60 Qualcomm Atheros, NW_INFO.REQ (Network Info Request)
67 7.167492 ZyxelCom_bb:69:c8 → Broadcast   HomePlug AV 60 Qualcomm Atheros, Unknown 0xa0b0
68 7.269765 ZyxelCom_bb:69:c8 → Broadcast   HomePlug AV 60 Qualcomm Atheros, NW_INFO.REQ (Network Info Request)
69 7.269770 ZyxelCom_bb:69:c8 → Broadcast   HomePlug AV 60 Qualcomm Atheros, RD_MOD.REQ (Read Module Data Request)
70 7.372171 ZyxelCom_bb:69:c8 → Broadcast   HomePlug AV 60 Qualcomm Atheros, NW_INFO.REQ (Network Info Request)

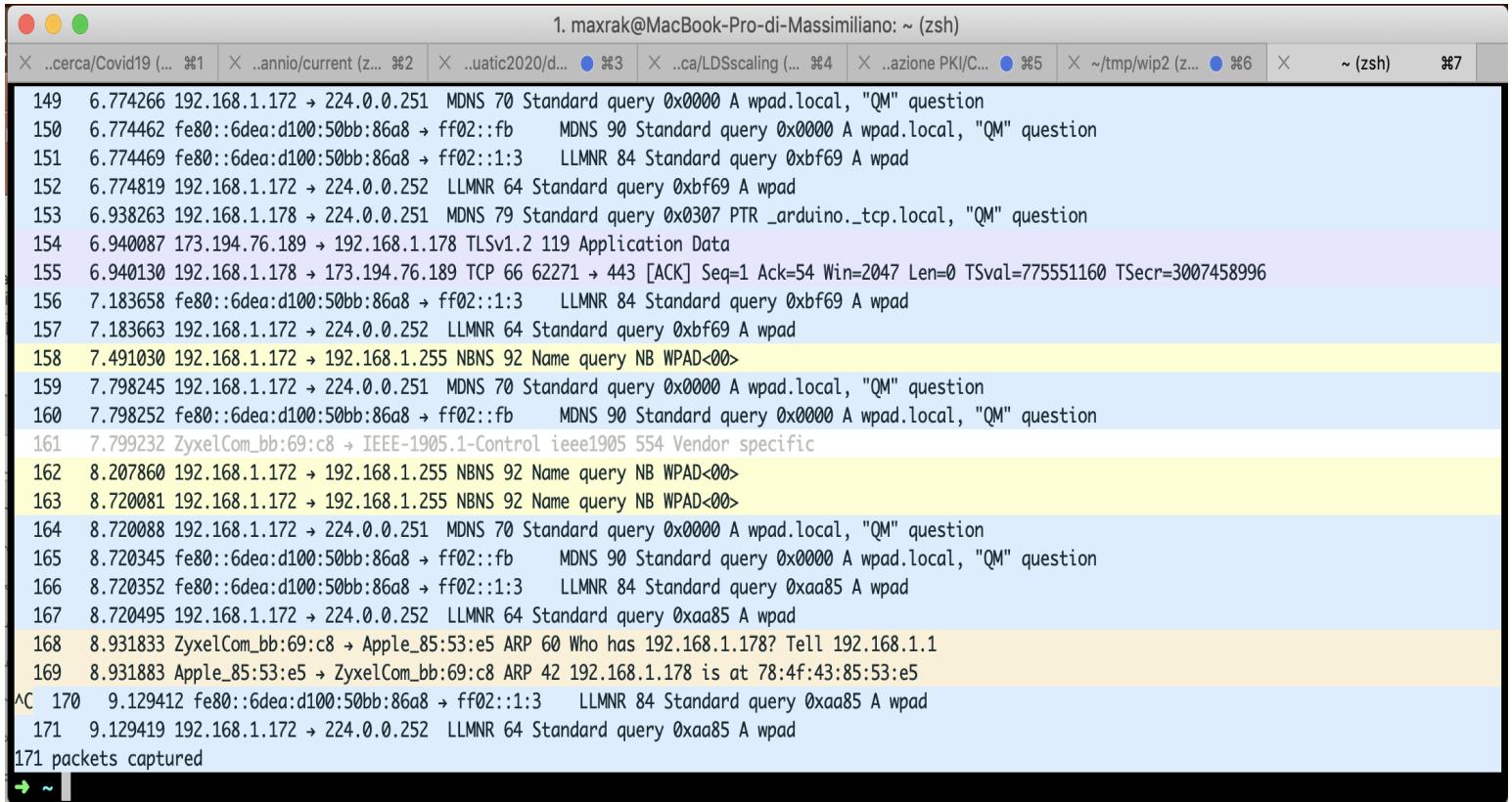
^C70 packets captured
→ ~ tshark
Capturing on 'Wi-Fi: en0'
1 0.000000 216.58.208.174 → 192.168.1.178 TLSv1.2 119 Application Data
2 0.000006 216.58.208.174 → 192.168.1.178 TLSv1.2 97 Application Data
3 0.000007 216.58.208.174 → 192.168.1.178 TLSv1.2 105 Application Data
4 0.000091 192.168.1.178 → 216.58.208.174 TCP 66 62275 → 443 [ACK] Seq=1 Ack=54 Win=2047 Len=0 TSval=775648489 TSecr=67719595
5 0.000092 192.168.1.178 → 216.58.208.174 TCP 66 62275 → 443 [ACK] Seq=1 Ack=85 Win=2046 Len=0 TSval=775648489 TSecr=67719595
6 0.000092 192.168.1.178 → 216.58.208.174 TCP 66 62275 → 443 [ACK] Seq=1 Ack=124 Win=2046 Len=0 TSval=775648489 TSecr=67719596
7 0.001305 192.168.1.178 → 216.58.208.174 TLSv1.2 105 Application Data
8 0.006130 216.58.208.174 → 192.168.1.178 TLSv1.2 105 [TCP Spurious Retransmission], Application Data
9 0.006190 192.168.1.178 → 216.58.208.174 TCP 78 [TCP Dup ACK 6#1] 62275 → 443 [ACK] Seq=40 Ack=124 Win=2048 Len=0 TSval=775648494 TSecr=67719687 SLE=85
SRE=124
10 0.018511 216.58.208.174 → 192.168.1.178 TCP 66 443 → 62275 [ACK] Seq=124 Ack=40 Win=5715 Len=0 TSval=67719699 TSecr=775648490
11 0.056801 192.168.1.178 → 35.186.224.47 TLSv1.2 101 Application Data
12 0.075488 35.186.224.47 → 192.168.1.178 TCP 66 443 → 62211 [ACK] Seq=1 Ack=36 Win=245 Len=0 TSval=4130070075 TSecr=775648544
13 0.086684 35.186.224.47 → 192.168.1.178 TLSv1.2 97 Application Data
14 0.086744 192.168.1.178 → 35.186.224.47 TCP 66 62211 → 443 [ACK] Seq=36 Ack=32 Win=2047 Len=0 TSval=775648573 TSecr=4130070086

^C14 packets captured
→ ~
```

# Tshark

---

- Analizzatore di pacchetti a linea di comando
  - tshark -- color



The screenshot shows a macOS terminal window titled "1. maxrak@MacBook-Pro-di-Massimiliano: ~ (zsh)". The window contains a list of network packets captured by Tshark. The packets are color-coded using the --color option:

- Packets 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, and 171 are highlighted in light blue.
- Packets 158, 162, 163, 168, and 169 are highlighted in yellow.
- Packets 170 and 171 are highlighted in light orange.

The Tshark output includes the following details:

- Packet 149: 6.774266 192.168.1.172 → 224.0.0.251 MDNS 70 Standard query 0x0000 A wpad.local, "QM" question
- Packet 150: 6.774462 fe80::6dea:d100:50bb:86a8 → ff02::fb MDNS 90 Standard query 0x0000 A wpad.local, "QM" question
- Packet 151: 6.774469 fe80::6dea:d100:50bb:86a8 → ff02::1:3 LLMNR 84 Standard query 0xb6f9 A wpad
- Packet 152: 6.774819 192.168.1.172 → 224.0.0.252 LLMNR 64 Standard query 0xb6f9 A wpad
- Packet 153: 6.938263 192.168.1.178 → 224.0.0.251 MDNS 79 Standard query 0x0307 PTR \_arduino.\_tcp.local, "QM" question
- Packet 154: 6.940087 173.194.76.189 → 192.168.1.178 TLSv1.2 119 Application Data
- Packet 155: 6.940130 192.168.1.178 → 173.194.76.189 TCP 66 62271 → 443 [ACK] Seq=1 Ack=54 Win=2047 Len=0 TStamp=775551160 TSect=3007458996
- Packet 156: 7.183658 fe80::6dea:d100:50bb:86a8 → ff02::1:3 LLMNR 84 Standard query 0xb6f9 A wpad
- Packet 157: 7.183663 192.168.1.172 → 224.0.0.252 LLMNR 64 Standard query 0xb6f9 A wpad
- Packet 158: 7.491030 192.168.1.172 → 192.168.1.255 NBNS 92 Name query NB WPAD<00>
- Packet 159: 7.798245 192.168.1.172 → 224.0.0.251 MDNS 70 Standard query 0x0000 A wpad.local, "QM" question
- Packet 160: 7.798252 fe80::6dea:d100:50bb:86a8 → ff02::fb MDNS 90 Standard query 0x0000 A wpad.local, "QM" question
- Packet 161: 7.799232 ZyxelCom\_bb:69:c8 → IEEE-1905.1-Control ieee1905 554 Vendor specific
- Packet 162: 8.207860 192.168.1.172 → 192.168.1.255 NBNS 92 Name query NB WPAD<00>
- Packet 163: 8.720081 192.168.1.172 → 192.168.1.255 NBNS 92 Name query NB WPAD<00>
- Packet 164: 8.720088 192.168.1.172 → 224.0.0.251 MDNS 70 Standard query 0x0000 A wpad.local, "QM" question
- Packet 165: 8.720345 fe80::6dea:d100:50bb:86a8 → ff02::fb MDNS 90 Standard query 0x0000 A wpad.local, "QM" question
- Packet 166: 8.720352 fe80::6dea:d100:50bb:86a8 → ff02::1:3 LLMNR 84 Standard query 0xa8a85 A wpad
- Packet 167: 8.720495 192.168.1.172 → 224.0.0.252 LLMNR 64 Standard query 0xa8a85 A wpad
- Packet 168: 8.931833 ZyxelCom\_bb:69:c8 → Apple\_85:53:e5 ARP 60 Who has 192.168.1.178? Tell 192.168.1.1
- Packet 169: 8.931883 Apple\_85:53:e5 → ZyxelCom\_bb:69:c8 ARP 42 192.168.1.178 is at 78:4f:43:85:53:e5
- Packet 170: 9.129412 fe80::6dea:d100:50bb:86a8 → ff02::1:3 LLMNR 84 Standard query 0xa8a85 A wpad
- Packet 171: 9.129419 192.168.1.172 → 224.0.0.252 LLMNR 64 Standard query 0xa8a85 A wpad

171 packets captured

# Tshark

---

- Analizzatore di pacchetti a linea di comando
  - tshark – color |grep TLS

```
1. tshark --color | grep --color=auto --exclude-dir={.bzr,CVS,.git,.hg,.svn} TLS (dumpcap)
X ..cerca/Covid19 (... ¶1) X ..anno/current (z... ¶2) X ..uatic2020/d... ¶3 X ..ca/LDSscaling (... ¶4) X ..azione PKI/C... ¶5 X ~/tmp/wip2 (z... ¶6) X tshark (dumpcap) ¶7
→ ~ tshark --color |grep TLS
Capturing on 'Wi-Fi: en0'
93 8 1.219822 192.168.1.178 → 193.206.103.194 TLSv1 583 Client Hello
11 1.222167 192.168.1.178 → 193.206.103.194 TLSv1 583 Client Hello
14 1.223112 192.168.1.178 → 193.206.103.194 TLSv1 583 Client Hello
16 1.231781 193.206.103.194 → 192.168.1.178 TLSv1.2 203 Server Hello, Change Cipher Spec, Encrypted Handshake Message
18 1.231984 192.168.1.178 → 193.206.103.194 TLSv1.2 117 Change Cipher Spec, Encrypted Handshake Message
19 1.232697 192.168.1.178 → 193.206.103.194 TLSv1.2 498 Application Data
21 1.233868 193.206.103.194 → 192.168.1.178 TLSv1.2 203 Server Hello, Change Cipher Spec, Encrypted Handshake Message
23 1.234130 192.168.1.178 → 193.206.103.194 TLSv1.2 117 Change Cipher Spec, Encrypted Handshake Message
24 1.234800 192.168.1.178 → 193.206.103.194 TLSv1.2 539 Application Data
26 1.235271 193.206.103.194 → 192.168.1.178 TLSv1.2 203 Server Hello, Change Cipher Spec, Encrypted Handshake Message
28 1.235540 192.168.1.178 → 193.206.103.194 TLSv1.2 117 Change Cipher Spec, Encrypted Handshake Message
29 1.236037 192.168.1.178 → 193.206.103.194 TLSv1.2 666 Application Data
32 1.246462 193.206.103.194 → 192.168.1.178 TLSv1.2 540 Application Data
35 1.259463 193.206.103.194 → 192.168.1.178 TLSv1.2 1434 Application Data
41 1.259544 193.206.103.194 → 192.168.1.178 TLSv1.2 391 Application Data
54 1.270967 193.206.103.194 → 192.168.1.178 TLSv1.2 1434 Application Data [TCP segment of a reassembled PDU]
65 1.272021 193.206.103.194 → 192.168.1.178 TLSv1.2 1434 Application Data [TCP segment of a reassembled PDU]
71 1.272028 193.206.103.194 → 192.168.1.178 TLSv1.2 1434 Application Data [TCP segment of a reassembled PDU]
73 1.272029 193.206.103.194 → 192.168.1.178 TLSv1.2 1053 Application Data, Application Data
82 1.516640 216.58.208.174 → 192.168.1.178 TLSv1.2 119 Application Data
83 1.516646 216.58.208.174 → 192.168.1.178 TLSv1.2 97 Application Data
84 1.516647 216.58.208.174 → 192.168.1.178 TLSv1.2 105 Application Data
88 1.517187 192.168.1.178 → 216.58.208.174 TLSv1.2 105 Application Data
```

Perchè non è a  
colori?

# netstat

---

- Lo stato delle connessioni del computer locale
- netstat

```
kaos@kaos:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 proxy.surfnet.iac:33297 mg-in-f125.:xmpp-client ESTABLISHED
tcp      0      0 proxy.surfnet.iac:33995 bylmsg5176516.phx.:1863 ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags     Type      State           I-Node Path
unix    3      [ ]     DGRAM                    13826   /dev/log
unix    2      [ ]     DGRAM                    5109    @/org/kernel/udev/udevd
unix    2      [ ]     DGRAM                   10443
@/org/freedesktop/hal/udev_event
unix    3      [ ]     STREAM     CONNECTED      34721   /tmp/orbit-kaos/linc-4876-0-
ebf915e300c0
unix    3      [ ]     STREAM     CONNECTED      34720
```

# netstat

---

- Lo stato delle connessioni del computer locale
- netstat

```
kaos@kaos:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 proxy.surfnet.iac:33297 mg-in-f125.:xmpp-client ESTABLISHED
tcp      0      0 proxy.surfnet.iac:33995 bylmsg5176516.phx.:1863 ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type      State           I-Node Path
unix     3      [ ]     DGRAM            13826   /dev/log
unix     2      [ ]     DGRAM            5109    @/org/kernel/udev/udevd
unix     2      [ ]     DGRAM            10443
@/org/freedesktop/hal/udev_event
unix     3      [ ]     STREAM   CONNECTED      34721   /tmp/orbit-kaos/linc-4876-0-
ebf915e300c0
unix     3      [ ]     STREAM   CONNECTED      34720
```

# netstat

---

- Lo stato delle connessioni del computer locale
- netstat

```
kaos@kaos:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 proxy.surfnet.iac:33297 mg-in-f125.:xmpp-client ESTABLISHED
tcp      0      0 proxy.surfnet.iac:33995 bylmsg5176516.phx.:1863 ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type      State           I-Node Path
unix    3      [ ]     DGRAM           13826   /dev/log
unix    2      [ ]     DGRAM           5109    @/org/kernel/udev/udevd
unix    2      [ ]     DGRAM           10443
@/org/freedesktop/hal/udev_event
unix    3      [ ]     STREAM     CONNECTED      34721   /tmp/orbit-kaos/linc-4876-0-
ebf915e300c0
unix    3      [ ]     STREAM     CONNECTED      34720
```

# netstat

---

- Lo stato delle connessioni del computer locale
- netstat

```
kaos@kaos:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address      State
tcp      0      0 proxy.surfnet.iac:33297 mg-in-f125.:xmpp-client ESTABLISHED
tcp      0      0 proxy.surfnet.iac:33995 bylmsg5176516.phx.:1863 ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags     Type      State           I-Node Path
unix    3      [ ]     DGRAM                    13826  /dev/log
unix    2      [ ]     DGRAM                    5109   @/org/kernel/udev/udevd
unix    2      [ ]     DGRAM                    10443 
@/org/freedesktop/hal/udev_event
unix    3      [ ]     STREAM     CONNECTED      34721  /tmp/orbit-kaos/linc-4876-0-
ebf915e300c0
unix    3      [ ]     STREAM     CONNECTED      34720
```

# Netstat - opzioni

---

- -a: che permette di vedere anche lo stato dei socket non attivi
- -t: che permette di vedere lo stato dei socket tcp
- -u: che permette di vedere lo stato dei socket udp
- -l: restituirà solo la lista dei servizi in ascolto (listening)
- -r: che permette di vedere la tabella di routing (restituisce lo stesso output del comando [route](#))
- -n: mostra gli indirizzi numerici (invece di quelli simbolici) degli host e delle porte
- -p: che permette di vedere il nome del programma e il relativo [PID](#) che ha instaurato una connessione (deve essere dato dall'utente root)
- -o: permette di vedere i PID (identificatori processo)

# Esercizio

---

- Installare Apache Web Server
- Attivare Httpd
- Verificare la pagina di default

# Wireshark - Capture

---

- Avvia Wireshark
  - Seleziona NIC
  - Avvia Capture
- Avvia Browser
  - Vai su <http://localhost>
- Stop Wireshark
- Esercizio analizza i pacchetti catturati e individua la navigazione effettuata.

GNS3 reproduce in modo simulato degli ambienti di rete usando le vere immagini dei componenti di rete.

L'obiettivo:

1) Progettazione

UNA VOLTA AVVIA TA LA RETE

- VPC non è configurato. Come lo configuro?
  - Show IP
- Scrivo "IP 10.0.1.1", cerca duplicati e gli mette netmask di default.
  - Non c'è gateway.