

# Network Security

## Introduzione alla network security

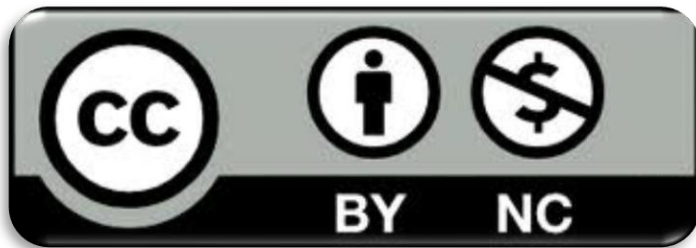


# License & Disclaimer

2

## License Information

This presentation is licensed under the  
Creative Commons BY-NC License



To view a copy of the license, visit:

<http://creativecommons.org/licenses/by-nc/3.0/legalcode>

## Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

# Obiettivi

3

- Imparare le basi teoriche per affrontare challenge di network security
- Imparare a monitorare e salvare il traffico di rete
- Essere in grado di analizzare il traffico di rete attraverso Wireshark
- Imparare le basi di Pyshark per automatizzare il parsing di un file pcap

# Indice

4

- Cenni teorici (Modello ISO/OSI e TCP/IP, IP, TCP e UDP, Porte)
- Salvare il traffico di rete
- Introduzione a Wireshark
- Wireshark: elementi nella GUI
- Wireshark: lavorare con i pacchetti
- Wireshark: seguire gli stream ed estrarre artefatti
- Pyshark: basi di scripting

# Modello ISO/OSI

5

- Sviluppato nel 1984 dall'International Organization for Standardization (ISO)
- Fornisce la logica per la comunicazione tra dispositivi all'interno di reti di calcolatori
- **Non** è lo standard implementato abitualmente su Internet

# OSI Layers

6



Una guida per lo sviluppo di protocolli di rete.

- Sette layers (strati)
- Basato su un approccio *divide et impera*: ogni problema viene risolto all'interno di un singolo layer
- Forte flessibilità: può adattarsi a nuovi protocolli e servizi di rete
- Non è necessario che le implementazioni si basino su tutti i livelli

# Modello TCP/IP

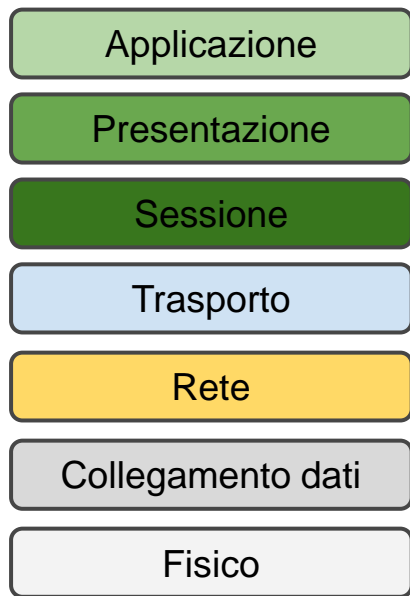
7

Il modello TCP/IP è un insieme di protocolli utilizzati per la comunicazione all'interno di reti.

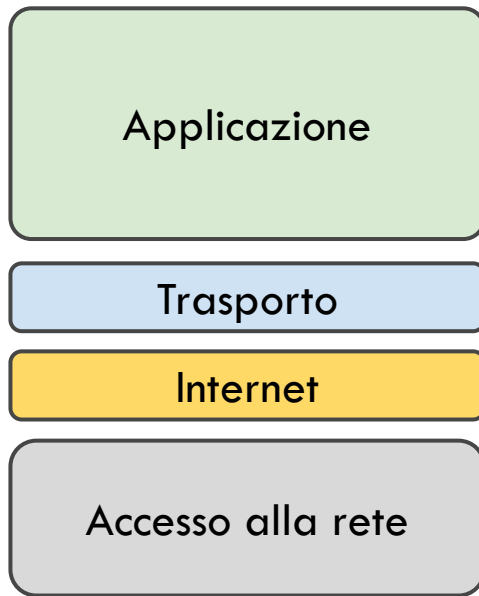
- È basato su quattro livelli
- È basato sui protocolli sviluppati e applicati su Internet (quindi è protocollo-dipendente)
- È lo standard utilizzato su Internet

# Modello TCP/IP

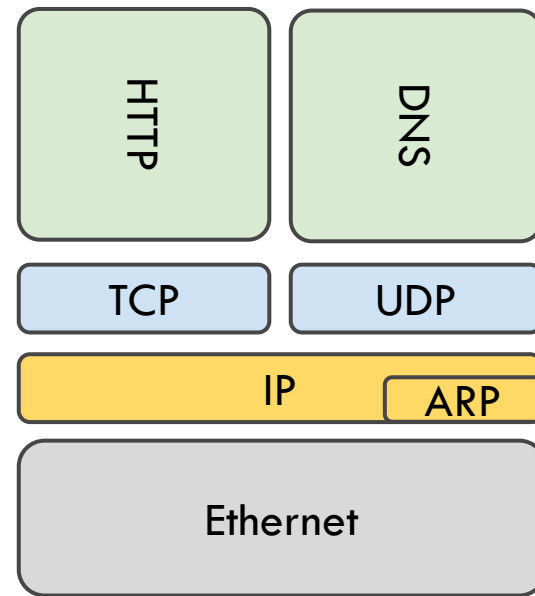
8



**ISO/OSI**



**TCP/IP**



**Protocolli standard**



# Internet Protocol (IP)

9

L'Internet Protocol (IP) è il protocollo utilizzato nel layer 3 (rete).

- È lo standard per l'instradamento di pacchetti all'interno delle reti
- Incapsula i dati e li passa in forma di pacchetti
- Ogni pacchetto include:
  - l'indirizzo IP della sorgente
  - l'indirizzo IP del destinatario

# Indirizzamento IP

10

- Gli indirizzi IP, nella versione 4, sono:
  - Formati da 32 bits
  - Raggruppati in quattro ottetti da 8 bit
  - Rappresentati da un numero decimale, ogni ottetto è separato dagli altri attraverso un punto

11000000 10101000 01100100 11001000

192 . 168 . 100 . 200

# Indirizzi IP

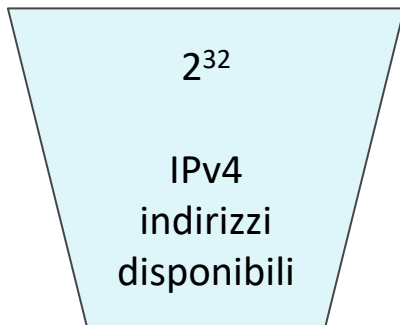
11



**192.168.100.200**  
(indirizzo dell'host)



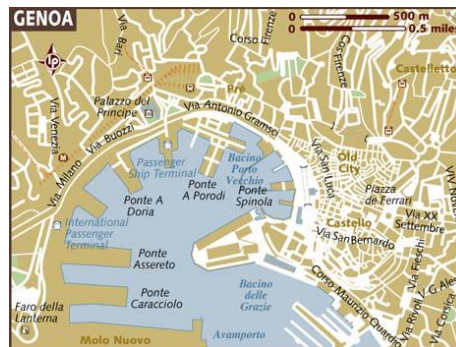
**192.168.100**  
(indirizzo di rete)



**35**  
(civico)



**via Dodecaneso**  
(via)



# TCP vs UDP

12

- TCP e UDP sono i protocolli più utilizzati nel layer 4
  - TCP prima crea una connessione, poi invia il contenuto della comunicazione. UDP trasmette direttamente dati
  - Entrambi controllano errori attraverso checksum, ma UDP non fa nessuna azione a riguardo. TCP invece è in grado di correggere gli errori
  - TCP ordina i pacchetti, UDP invece non ha un ordine predefinito per i pacchetti
  - I pacchetti TCP sono più grandi in termini di spazio rispetto ai pacchetti UDP

# Porte

13

- Il layer 4 si occupa anche della comunicazione tra processi remoti
- Questi processi sono identificati attraverso porte così definite:
  - 16-bit unsigned integer (0-65535, 0 riservato)
  - **Well-known ports** (0-1023): usate da processi di Sistema particolarmente rilevanti
  - **Registered ports** (1024-49151): assegnate da IANA a particolari applicazioni previa registrazione
  - **Ephemeral ports** (49152-65535): porte dinamiche o per servizi private.
- Nonostante questa convenzione, qualsiasi servizio può ascoltare su qualsiasi porta!

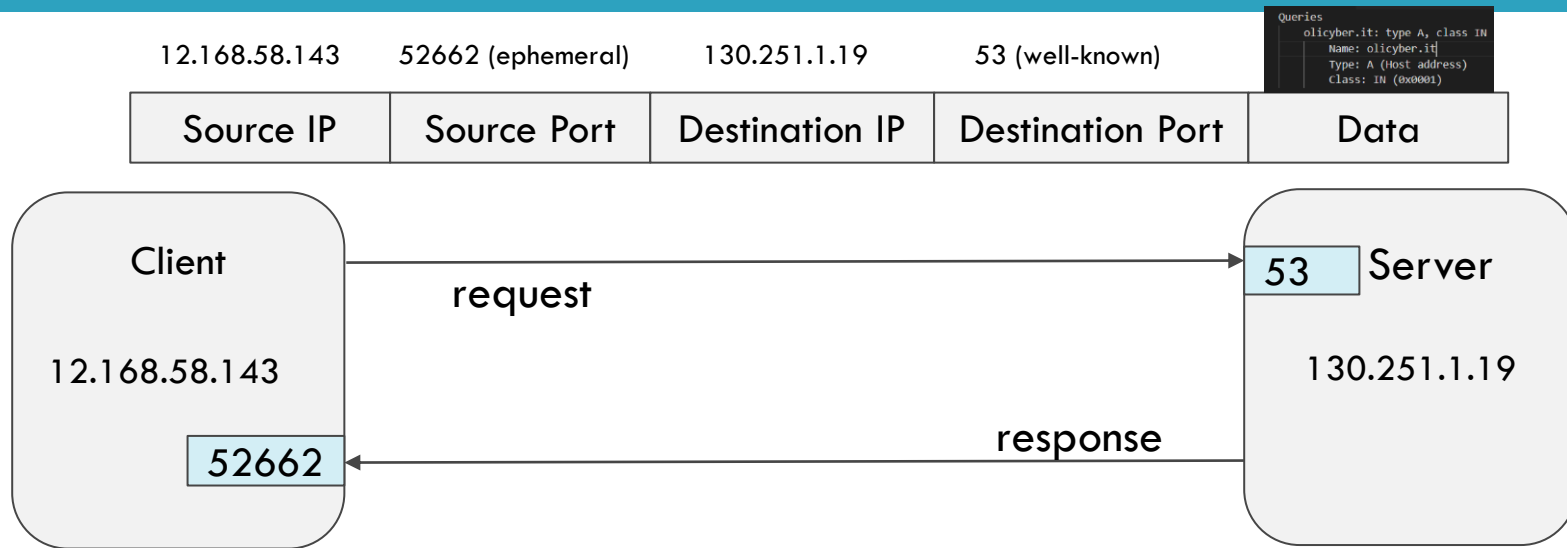
# Il modello client-server

14

- Il modello client-server è il paradigma più utilizzato per la comunicazione all'interno di reti
  - Si ha una relazione per cui un programma (client) richiede un servizio o una risorsa a un altro programma (server)
  - Il client deve conoscere l'indirizzo del server
  - Il server non per forza deve conoscere l'indirizzo (o l'esistenza) del client prima della connessione

# Il modello client-server (esempio DNS)

15



Data	Destination Port	Destination IP	Source Port	Source IP
<pre>Answers [...] olicyber.it: type A, class IN Name: olicyber.it Type: A (Host address) Class: IN (0x0001) Addr: 104.21.89.84 [...]</pre>	52662 (ephemeral)	12.168.58.143	53 (well-known)	130.251.1.19

# Salvare il traffico di rete

16

- Per poter analizzare il traffico di rete passato è necessario prima salvarlo (dump del traffico)
- Tcpcmdump (<https://github.com/the-tcpdump-group/tcpdump>) è un esempio di tool che permette di visualizzare e salvare il traffico di rete
- Generalmente il traffico viene salvato all'interno di file con formato Packet Capture dall'estensione **.pcap**



# Wireshark

17

- Wireshark è un tool che permette di catturare traffico da una rete (sniffer) e analizzarlo
  - L'analisi può essere effettuata real-time oppure usando un file precedentemente salvato
  - I pacchetti sono composti da dati *generic*, essi andranno poi valutati livello per livello per estrapolarne informazioni
- Disponibile sia su Linux che Windows:
  - <https://www.wireshark.org/>

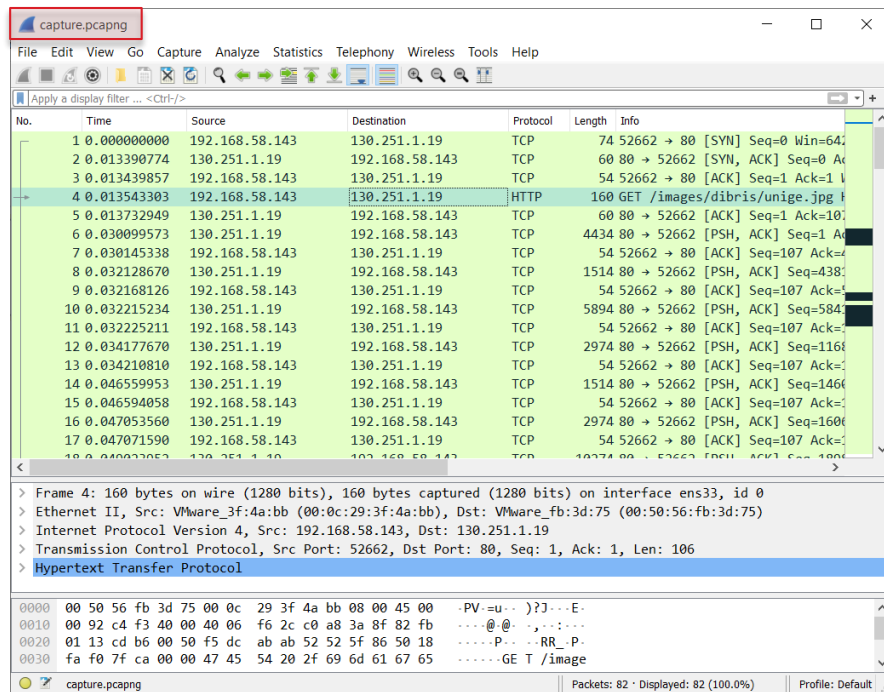
# Challenges di network

18

- Nella maggior parte delle sfide di network security vengono forniti ai giocatori dei file PCAP
- Per risolvere queste challenge i giocatori devono saper analizzare questi file per
  - Trovare la flag direttamente tra i byte
  - Rispondere a delle domande relative al traffico analizzato
- Wireshark è uno strumento utile per risolvere questo tipo di sfide

# Wireshark GUI

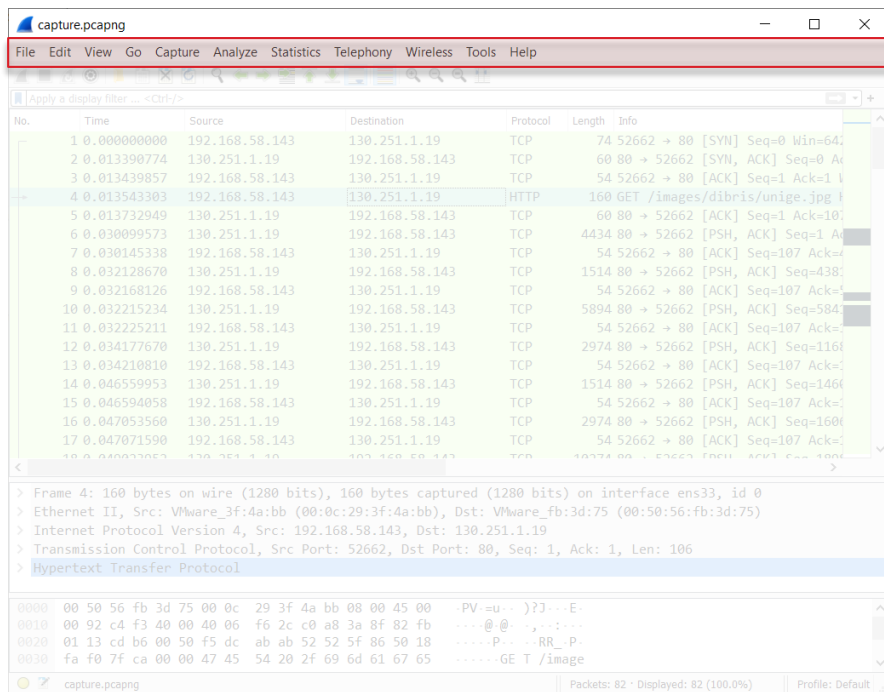
19



- Wireshark ha una interfaccia grafica (Graphical User Interface GUI)
- È possibile aprire un file .pcap da analizzare dal menu *File* oppure utilizzando il comando open (CTRL+o)  
Esso apparirà nella schermata principale
- Utilizzando una versione di Wireshark in inglese è più facile cercare aiuto e informazioni su Internet

# Wireshark GUI: menu

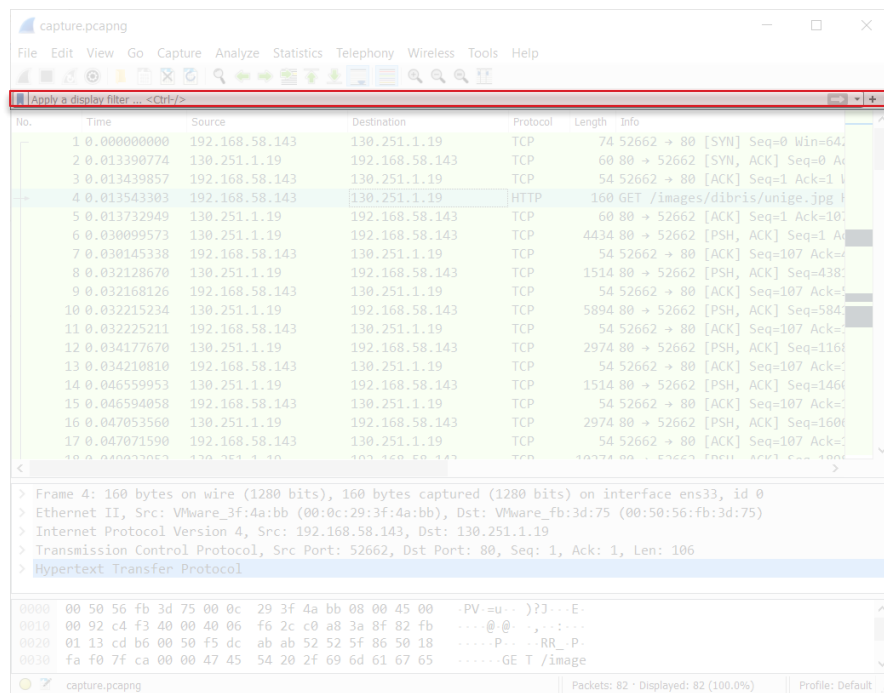
20



- Il **menu** è utilizzato per effettuare azioni
- Le azioni di maggiore interesse sono:
  - **File**: aprire o raggruppare file, salvare, stampare o esportare dati
  - **Edit**: trovare un pacchetto, evidenziare flussi, gestire le configurazioni
  - **View**: controllare come vengono visualizzati i pacchetti (colore, font ...)
  - **Go**: andare ad un determinato pacchetto
  - **Analyze**: manipolare, filtrare, attivare o disattivare il focus su determinati protocolli, seguire i flussi (stream)
  - **Statistics**: visualizzare diverse statistiche quali gli indirizzi IP coinvolti nella comunicazione, numero di pacchetti scambiati etc. Utili per farsi un'idea iniziale sul tipo di traffico contenuto in quel file aperto

# Wireshark GUI: filter toolbar

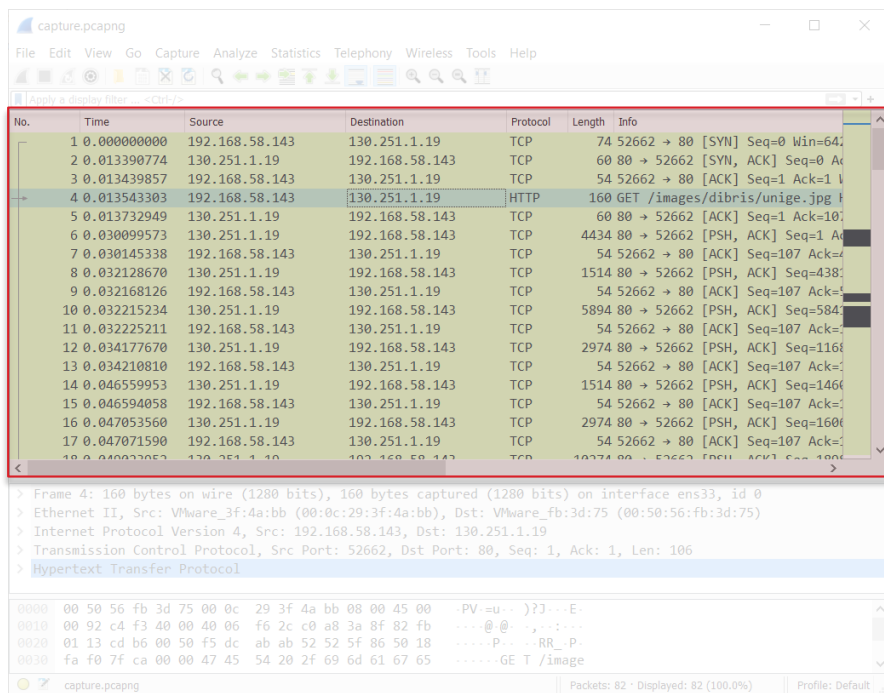
21



- Il menu dei filtri permette di modificare e applicare rapidamente dei filtri sui pacchetti
  - Gestire o salvare filtri salvati
  - Reset dei filtri
  - Applicare il filtro corrente
  - Selezionare un filtro da una lista di filtri usati di recente
  - Aggiungere un nuovo filter button (shortcut per applicare un determinato filtro)

# Wireshark GUI: lista dei pacchetti

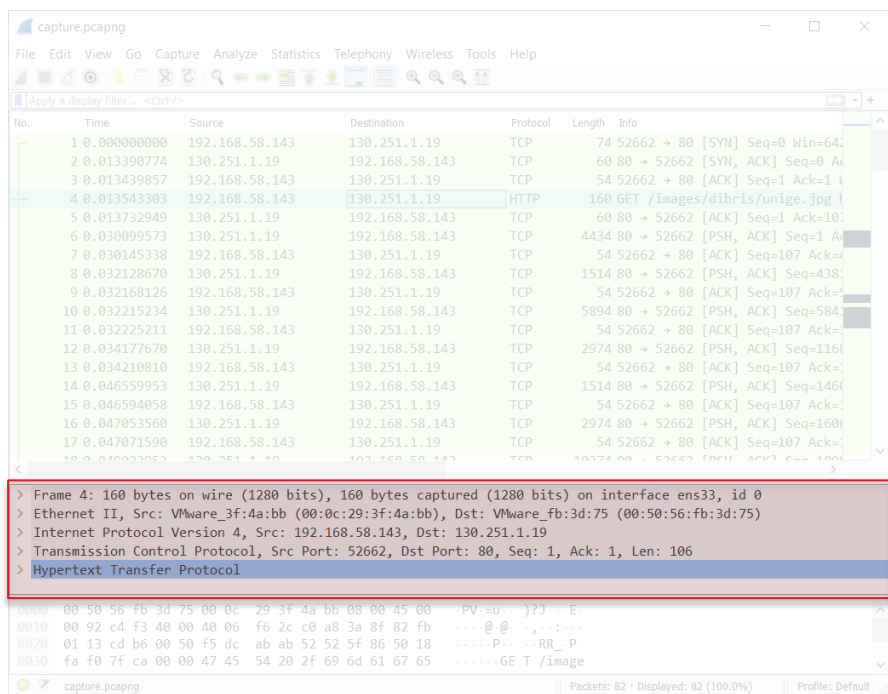
22



- Il pannello centrale mostra la lista di tutti i pacchetti catturati
- Ogni linea corrisponde a un pacchetto catturato
- Selezionando un pacchetto (singolo click) vengono visualizzati i dettagli del pacchetto all'interno della sottostante sezione *packet details* e *packet bytes*
- Si può cliccare sulle colonne per ordinare i pacchetti

# Wireshark GUI: packet details

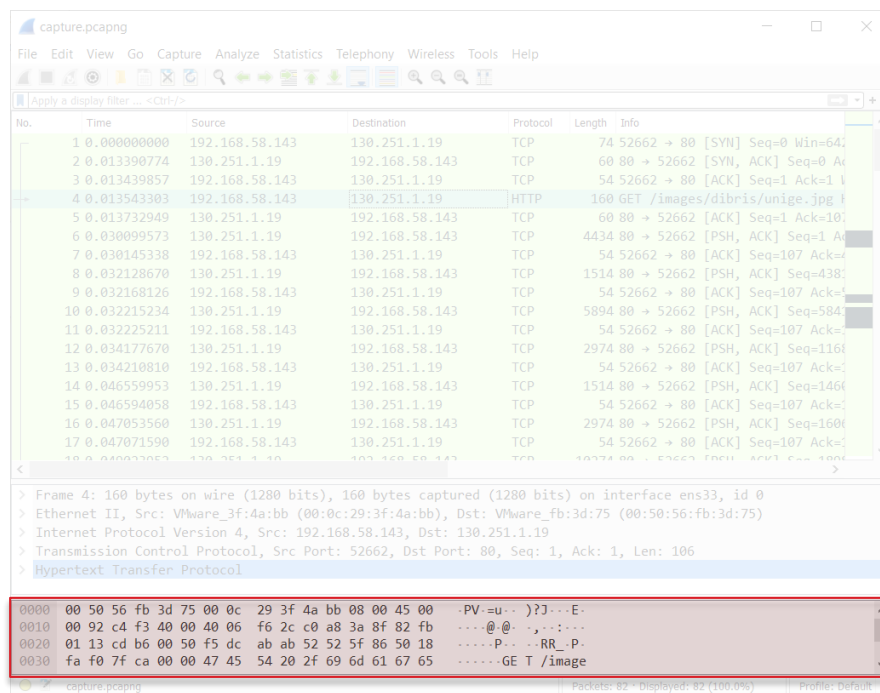
23



- Il pannello sottostante alla lista dei pacchetti mostra i dettagli del pacchetto selezionato
- In particolare mostra i protocolli e i campi del pacchetto con una struttura ad albero  
Ciascun ramo corrisponde ad un protocollo e può essere espanso per visualizzare i corrispondenti dati contenuti all'interno

# Wireshark GUI: packet bytes

24



- Il pannello contenente i *packet bytes* mostra i dati del pacchetto selezionato in stile *hexdump*
- Ciascuna linea contiene:
  - L'offset dei dati
  - 16 byte rappresentati in base esadecimale
  - 16 caratteri ASCII (i caratteri non stampabili vengono rappresentati con un punto ".")



# La lista dei pacchetti

25

Pacchetti  
collegati

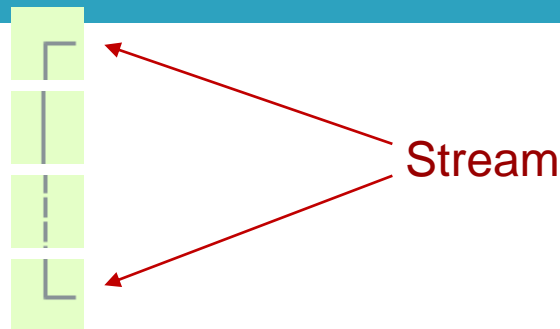
No.	Time	Source	Destination	Protocol	Length	Info	colonne
1	0.000000000	192.168.58.143	130.251.1.19	TCP	74	52662 → 80 [SYN] Seq=0 Win=64240 Len=0	
2	0.013390774	130.251.1.19	192.168.58.143	TCP	60	80 → 52662 [SYN, ACK] Seq=0 Ack=1 Win=6	
3	0.013439857	192.168.58.143	130.251.1.19	TCP	54	52662 → 80 [ACK]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

1. **No.** Numero del pacchetto all'interno del file. Anche se vengono applicati dei filtri questo numero non cambia.
2. **Time** Timestamp del pacchetto (per cambiare formato andare su View → Time display format)
3. **Source** Indirizzo IP del mittente
4. **Destination** Indirizzo IP del destinatario
5. **Protocol** Nome del protocollo
6. **Length** Lunghezza del pacchetto
7. **Info** Informazioni riguardo il contenuto del pacchetto

# Simboli per pacchetti collegati (stesso flusso/stream)

26

- Primo pacchetto del flusso
- Parte del flusso selezionato
- **Non** parte del flusso selezionato
- Ultimo pacchetto del flusso
- Richiesta
- Risposta
- Il pacchetto selezionato è una conferma di ricezione di questo pacchetto
- Il pacchetto selezionato è un duplicato di conferma di ricezione di questo pacchetto
- Il pacchetto selezionato ha a che fare con questo pacchetto (as esempio parte del contenuto)



## 27

- Wireshark - Coloring Rules - Default

Name	Filter
<input checked="" type="checkbox"/> Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update
<input checked="" type="checkbox"/> HSRP State Change	hsrp.state != 8 && hsrp.state != 16
<input checked="" type="checkbox"/> Spanning Tree Topology Change	stp.type == 0x80
<input checked="" type="checkbox"/> OSPF State Change	ospf.msg != 1
<input checked="" type="checkbox"/> ICMP errors	icmp.type eq 3    icmp.type eq 4    icmp.type eq 5    icmp.type eq 11    icmp.v6.type eq 1
<input checked="" type="checkbox"/> ARP	arp
<input checked="" type="checkbox"/> ICMP	icmp    icmpv6
<input checked="" type="checkbox"/> TCP RST	tcp.flags.reset eq 1
<input checked="" type="checkbox"/> SCTP ABORT	sctp.chunk_type eq ABORT
<input checked="" type="checkbox"/> TTL low or unexpected	((ip.dst == 224.0.0.0/4 && ip.ttl < 5 && ip.mh && !ospf)    (ip.dst == 224.0.0.0/4 && ip.mh && !ospf))
<input checked="" type="checkbox"/> Checksum Errors	eth.fcs.status == "Bad"    ip.checksum.status == "Bad"    tcp.checksum.status == "Bad"    udp.checksum.status == "Bad"
<input checked="" type="checkbox"/> SMB	smb    nbss    nbns    nbpx    ipxsap    netbios
<input checked="" type="checkbox"/> HTTP	http    tcp.port == 80    http2
<input checked="" type="checkbox"/> IPX	ipx    spx
<input checked="" type="checkbox"/> DCE/RPC	dcerpc
<input checked="" type="checkbox"/> Routing	hsrp    eigrp    ospf    bgp    cdp    vrrp    carp    gvrp    igmp    ismp
<input checked="" type="checkbox"/> TCP SYN/FIN	tcp.flags & 0x02    tcp.flags.fin == 1
<input checked="" type="checkbox"/> TCP	tcp
<input checked="" type="checkbox"/> UDP	udp
<input checked="" type="checkbox"/> Broadcast	eth[0] & 1

Double click to edit. Drag to move. Rules are processed in order until a match is found.

OK Cancel Import... Export... Help

# Filtri

28

- Wireshark fornisce un linguaggio per gestire i filtri
- Con i filtri è possibile controllare quali pacchetti andare a visualizzare
- I filtri possono essere utilizzati per:
  - Visualizzare solo pacchetti di un determinato protocollo
  - Cercare pacchetti con un campo con un determinato valore
  - [...]
- I filtri possono essere combinati formando espressioni complesse, utilizzando operatori logici e parentesi

# Creare un filtro

29

1. Help → Manual Pages → Wireshark Filters
2. Expression builder: click con il tasto destroy sulla toolbar → Display Filter Expression...
3. Selezionare il campo sul pannello *packet details*:
  1. Apply as filter: filtra la lista dei pacchetti con solo quelli che soddisfano l'espressione
  2. Prepare as filter: scrive l'espressione ma essa non viene ancora applicata alla lista dei pacchetti

The screenshot shows the Wireshark interface. On the left, the 'packet details' pane is expanded to show the 'Transmission Control Protocol' section. The 'Source Address' field is highlighted with a red box and labeled 'Campo protocollo'. A red arrow points from this field to the 'Display Filter Expression' bar on the right, which is labeled 'Prepare a Filter'. The filter expression 'ip.src == 145.254.160.237' is entered in the bar. Below the filter bar, a table with columns 'No.', 'Time', and 'Source' is visible. At the bottom left, the 'Olicyber' logo is present. At the bottom right, the 'CYBERSECURITY NATIONAL LAB' logo is present. The date 'Rel. 14.01.2023' is also visible.

Campo protocollo

Prepare a Filter

ip.src == 145.254.160.237

No. Time Source

Source Address (ip.src), 4 bytes Filter field

Rel. 14.01.2023

# Filtri utili

30

- `ip.src` / `ip.dst` → filtra rispettivamente per l'indirizzo di origine e di destinazione
- Per filtrare per protocollo basta scrivere il nome che compare nella rispettiva colonna, tutto in lettere minuscole
- `protocol.port` → filtra per porta sul protocollo specificato, sostituire 'protocol' con il protocollo desiderato in letter minuscole
- `frame contains "stringa"` → filtra tutti i pacchetti che contengono "stringa"
- `frame.len` → filtra per la lunghezza (dimensione) del pacchetto, espressa in bytes
- Si riportano link della documentazione per approfondire:
  - <https://wiki.wireshark.org/DisplayFilters>
  - <https://www.wireshark.org/docs/man-pages/wireshark-filter.html>

# Seguire stream

31

- **Seguire uno stream** mostra una diversa visualizzazione del traffico di rete: anziché visualizzare un pacchetto singolo, vengono visualizzati i dati trasmessi tra mittente e destinatario
- Quando viene visualizzato uno stream, un filtro relativo allo stream corrente viene applicato. Solo i pacchetti di quello stream verranno visualizzati

7	9.025432	72.163.7.54	192.168.1.135	ETD	07 Response	220-\tCisco Syste
8	9.025433	72.163.7.54	192.168.	Mark/Unmark Packet	Ctrl+M	220-
9	9.025434	72.163.7.54	192.168.	Ignore/Unignore Packet	Ctrl+D	220- \t\t\t\t\t\t
10	9.025434	72.163.7.54	192.168.	Set/Unset Time Reference	Ctrl+T	220-\tPhone: +1.8
11	9.025435	72.163.7.54	192.168.	Time Shift...	Ctrl+Shift+T	220-
12	9.025435	72.163.7.54	192.168.	Packet Comment...	Ctrl+Alt+C	220- Local time
13	9.025435	72.163.7.54	192.168.			220-
14	9.025532	192.168.1.135	72.163.7	Edit Resolved Name		[ACK] Seq=1 Ack=
15	9.025860	72.163.7.54	192.168.	Apply as Filter		220-\tThis system
16	9.037860	72.163.7.54	192.168.	Prepare a Filter		220-\t- FILES.CI
17	9.037862	72.163.7.54	192.168.	Conversation Filter		220-
18	9.037863	72.163.7.54	192.168.	Colorize Conversation		220-\tPlease read
19	9.037864	72.163.7.54	192.168.	SCTP		220-\tWARNING! -
20	9.037864	72.163.7.54	192.168.	Follow		220-\t+DASCOMB AR
21	9.037865	72.163.7.54	192.168.			TCP Stream
22	9.037866	72.163.7.54	192.168.	Copy		UDP Stream
Frame 7: 97 bytes on wire (776 bits), 97 bytes cap						SSL Stream
Ethernet II Src: Amtec 32:a1:59 (00:60:3b:32:a1:5						HTTP Stream
						Protocol Preferences
						Decade &c

# Seguire uno stream (esempio)

32

- Telnet è un protocollo di tipo client-server che può essere utilizzato per aprire la linea di comando su un host remoto
- In **blu** vengono visualizzati i dati dal server al client (ad esempio il prompt di login)
- In **rosso** vengono visualizzati i dati dal client al server (ad esempio il client che invia la password al server)
- I caratteri non stampabili vengono rappresentati con il punto “.”

```
.....!..".'.#..%..%.....!..".'.P.....b.....B.....
.....".'.#..&..$..&..$.....#.....9600,9600....#bam.zing.org:
0.0.....DISPLAY.bam.zing.org:0.0.....xterm-color.....
OpenBSD/i386 (oof) (tty1)

login: ..".....ffaakke
Password:user

Last login: Thu Dec  2 21:32:59 on tty1 from bam.zing.org
Warning: no Kerberos tickets issued.
OpenBSD 2.6-beta (OOF) #4: Tue Oct 12 20:42:32 CDT 1999

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

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

$ llss
$ llss --aa
.
.      .cshrc  .login  .mailrc  .profile  .rhosts
$ //ssbbiinn//ppiinnngg  wwwwww.yyaahhoooo..ccoomm

PING www.yahoo.com (204.71.200.74): 56 data bytes
64 bytes from 204.71.200.74: icmp_seq=0 ttl=239 time=73.569 ms
64 bytes from 204.71.200.74: icmp_seq=1 ttl=239 time=71.099 ms
64 bytes from 204.71.200.74: icmp_seq=2 ttl=239 time=68.728 ms
64 bytes from 204.71.200.74: icmp_seq=3 ttl=239 time=73.122 ms
64 bytes from 204.71.200.74: icmp_seq=4 ttl=239 time=71.276 ms
64 bytes from 204.71.200.74: icmp_seq=5 ttl=239 time=75.831 ms
64 bytes from 204.71.200.74: icmp_seq=6 ttl=239 time=70.191 ms
64 bytes from 204.71.200.74: icmp_seq=7 ttl=239 time=74.528 ms
64 bytes from 204.71.200.74: icmp_seq=8 ttl=239 time=74.514 ms
64 bytes from 204.71.200.74: icmp_seq=9 ttl=239 time=75.188 ms
64 bytes from 204.71.200.74: icmp_seq=10 ttl=239 time=72.925 ms
....C
--- www.yahoo.com ping statistics ---
13 packets transmitted, 11 packets received, 15% packet loss
round-trip min/avg/max = 68.728/72.807/75.831 ms
$ eexxiitt
```

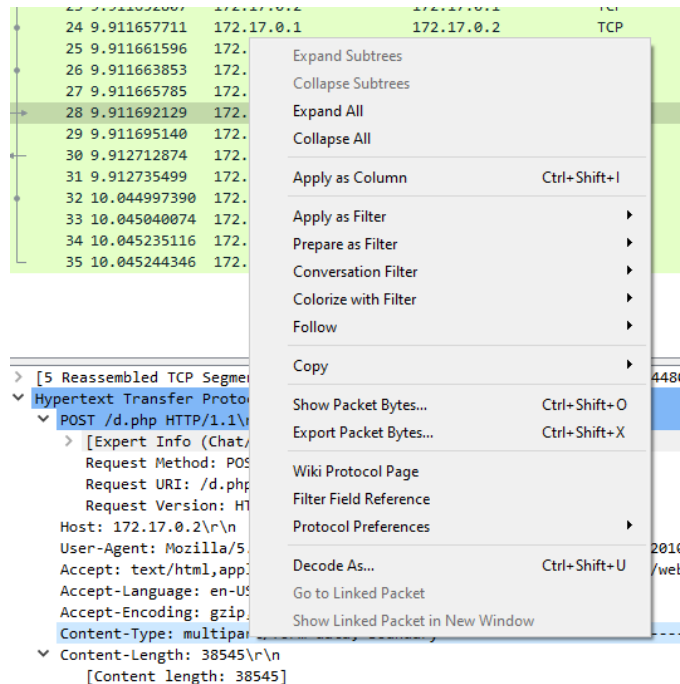


# Estrarre artefatti dagli stream: esempio

33

## ➤ Estrarre e salvare un file JPEG scaricato usando HTTP

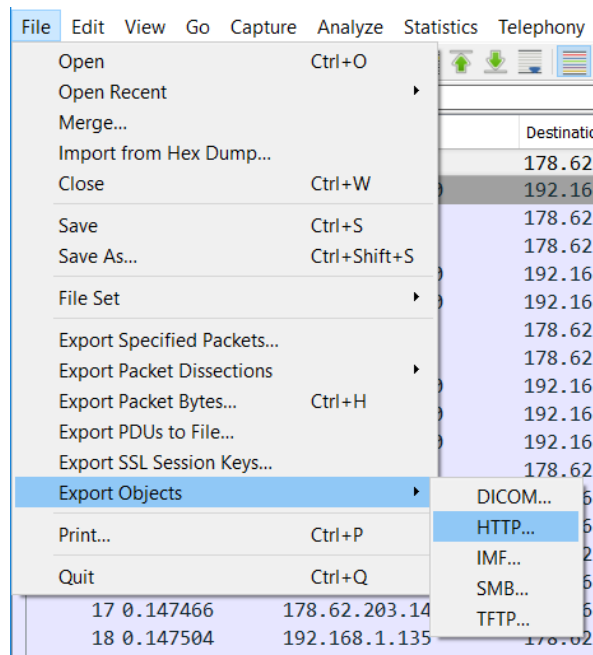
1. Selezionare il pacchetto
2. Andare sul packet bytes del pacchetto
3. Cliccare con il tasto destro sul campo contenente l'artefatto
4. Cliccare su *Export packet Bytes*
5. Salvare il file



# Estrarre artefatti: esempio 2

34

- File → Export Objects
- Questa features analizza gli stream di alcuni protocolli e ricostruisce alcuni oggetti come le pagine HTML, immagini etc...
- Questi file possono essere esportati e salvati su disco



# Pyshark

35

- Pyshark è un wrapper di tshark (versione di Wireshark per riga di comando) per Python
- Utile per automatizzare operazioni sui molti pacchetti
- Installazione:

```
sudo apt install tshark #se non già presente  
pip3 install pyshark
```

# Pyshark: utilizzo (1)

36

- Per prima cosa è necessario importare la libreria `import pyshark` e caricare il file pcap:

```
cap = pyshark.FileCapture('/path/to/pcap/file.pcap')
```

- La variabile `cap` è una lista che contiene tutti i pacchetti presenti all'interno del file di cattura. Quindi, per accedere a un pacchetto basta accedere all'elemento della lista `cap[n]`. Attenzione: in Wireshark il primo pacchetto è il numero 1, mentre in Pyshark è lo 0, quindi bisogna sottrarre 1 al numero del pacchetto di Wireshark per ottenere il corrispettivo in Pyshark (e aggiungere 1 nel caso opposto)

- Con un semplice for-loop è possibile ciclare tra tutti i pacchetti della cattura: `for packet in cap:`

# Pyshark: utilizzo (2)

37

- I pacchetti sono divisi in layer. È necessario accedere al layer appropriato prima di selezionare il campo desiderato:
  - `packet.ip.dst` → accede al layer IP del pacchetto e seleziona il campo dell'IP di destinazione
  - `packet.tcp.payload` → accede al layer TCP del pacchetto e seleziona il campo del payload
- Per verificare se un layer è presente nel pacchetto si può utilizzare il nome del layer: `if 'IP' in packet:`
- Per visualizzare tutti possibili campi, si può usare l'attributo `packet.layer.field_names` (es: `packet.ip.field_names`)

# Pyshark: esempio

38

Semplice script per stampare a schermo (come bytestring) il payload di tutti i pacchetti tcp di un file:

```
import pyshark
cap = pyshark.FileCapture('/path/to/pcap/file.pcap')

for packet in cap:
    # Controllo se il pacchetto è TCP (controllo è case insensitive)
    if 'tcp' in packet:
        try:
            print(packet.tcp.payload.binary_value)
        except:
            continue
```

# Network Security

## Introduzione alla network security

