Reti di telecomunicazioni 2024 Latini Luca

AUTORE:

Relazione dell’elaborato che realizza la soluzione della traccia numero 3 a cura di **Latini Luca** (matricola: **0001091148**)

RICHIESTA:

- **Descrizione**: Usare Wireshark per catturare e analizzare il traffico di rete durante una sessione di trasferimento file (es. FTP o HTTP). Gli studenti dovranno esaminare i pacchetti per comprendere il funzionamento del protocollo TCP/IP.
- **Obiettivi**: Identificare la sequenza di handshake TCP, analizzare la frammentazione dei pacchetti e il controllo di flusso, e riconoscere i campi chiave nei pacchetti.
- **Consegne richieste**: Report con screenshot delle catture di pacchetti, analisi delle metriche chiave (es. numero di pacchetti, tempi di latenza) e spiegazione di eventuali ritrasmissioni o problemi di connessione osservati.

Interpretazione di una Cattura Packet Capture: Protocollo TCP/IP e HTTP

```
493 21.325595    10.201.106.166    193.206.135.26    TCP      66 51384 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
494 21.331268    193.206.135.26    10.201.106.166    TCP      66 80 → 51384 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1250 SACK_PERM WS=128
495 21.331397    10.201.106.166    193.206.135.26    TCP      54 51384 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
496 21.331655    10.201.106.166    193.206.135.26    HTTP    178 GET /ncsi.txt HTTP/1.1
497 21.334963    193.206.135.26    10.201.106.166    TCP      54 80 → 51384 [ACK] Seq=1 Ack=125 Win=8222592 Len=0
498 21.338585    193.206.135.26    10.201.106.166    TCP      54 [TCP Window Update] 80 → 51384 [ACK] Seq=1 Ack=125 Win=64128 Len=0
499 21.338987    193.206.135.26    10.201.106.166    HTTP    233 HTTP/1.1 200 OK  (text/plain)
500 21.338987    193.206.135.26    10.201.106.166    TCP      54 80 → 51384 [FIN, ACK] Seq=180 Ack=125 Win=64128 Len=0
501 21.339054    10.201.106.166    193.206.135.26    TCP      54 51384 → 80 [ACK] Seq=125 Ack=181 Win=130816 Len=0
502 21.339348    10.201.106.166    193.206.135.26    TCP      54 51384 → 80 [FIN, ACK] Seq=125 Ack=181 Win=130816 Len=0
503 21.347916    193.206.135.26    10.201.106.166    TCP      54 80 → 51384 [ACK] Seq=181 Ack=126 Win=64128 Len=0
```

Scenario rappresentato nella cattura

1. **Handshake TCP iniziale (pacchetti 493-495):**
   - **Pacchetto 493:** Il client (IP 10.201.106.166), porta 51384, invia un pacchetto **SYN** al server (IP 193.206.135.26) sulla porta 80 per iniziare la comunicazione. Il pacchetto include:
     - `Seq=0`: Il numero di sequenza iniziale del client.
     - `Win=64240`: La dimensione della finestra di ricezione.
     - Opzioni TCP, come MSS=1460 (dimensione massima del segmento) e WS=256 (fattore di scala della finestra TCP).
   - **Pacchetto 494:** Il server risponde con un pacchetto **SYN, ACK**, accettando la connessione:
     - `Ack=1`: Conferma che ha ricevuto il numero di sequenza iniziale del client.
     - `MSS=1250`: Indica un valore differente di MSS per il server.
   - **Pacchetto 495:** Il client conferma la ricezione con un pacchetto **ACK**:
     - `Seq=1` e `Ack=1`: La comunicazione TCP viene stabilita.

   Questo scambio rappresenta il completamento del **handshake TCP a tre vie**.

1. **Richiesta HTTP (pacchetto 496):**
    - Il client invia una richiesta HTTP GET per il file `/ncsi.txt` con protocollo HTTP/1.1.
    - Questa richiesta suggerisce che il client sta cercando di testare la connessione internet. Questo tipo di file è spesso associato ai test di connettività di rete.
    - Il pacchetto include `Seq=1`, quindi i dati HTTP iniziano esattamente dove si era fermato l'handshake.

---

1. **Scambio di pacchetti TCP durante il trasferimento dei dati (pacchetti 497-500):**
    - **Pacchetto 497:** Il server invia un pacchetto **ACK** per confermare la ricezione della richiesta HTTP.
    - **Pacchetto 498:** Il server invia un messaggio di aggiornamento finestra TCP (**TCP Window Update**).
    - **Pacchetto 499:** Il server risponde con un messaggio HTTP 1.1 **200 OK**, indicando che la richiesta è stata elaborata con successo. Il corpo della risposta contiene probabilmente il contenuto del file `/ncsi.txt`:
        - Tipo di contenuto: `text/plain` (file di testo).
    - **Pacchetto 500:** Il client invia un pacchetto **ACK** di ricezione.

---

1. **Chiusura della connessione TCP (pacchetti 501-503):**
    - **Pacchetto 501:** Il client invia un pacchetto **FIN, ACK** per segnalare l'intenzione di chiudere la connessione TCP:
        - `Seq=125` e `Ack=181`: Il client ha terminato l'invio di dati e conferma che ha ricevuto tutti i dati dal server fino al numero di sequenza 180.
    - **Pacchetto 502:** Il server risponde con **FIN, ACK**, indicando che anche lui è pronto a chiudere la connessione.
    - **Pacchetto 503:** Il client invia un ultimo **ACK** al server per confermare la terminazione della comunicazione.

Questo scambio rappresenta una **chiusura TCP** , che garantisce la chiusura ordinata della connessione da entrambe le parti.

Analisi e interpretazione dei principali pacchetti

Pacchetto 493: Handshake TCP (SYN)

- **Descrizione:** Il client **10.201.106.166** avvia una connessione TCP al server **193.206.135.26** (porta 80), inviando un pacchetto SYN.
- **Caratteristiche principali:**
  - **Protocollo di livello 2 (Ethernet):** Si tratta di una comunicazione su Ethernet, con destinazione il MAC `00:09:0f:09:00:12`.
  - **Livello 3 (IPv4):**
    - Origine: 10.201.106.166.
    - Destinazione: 193.206.135.26.
    - **Flags:** `Don't fragment` per mantenere la trasmissione dei pacchetti non frammentata.
  - **Livello 4 (TCP):**
    - Porta di origine: 51384 (client).
    - Porta di destinazione: 80 (standard HTTP).
    - **Flag SYN:** Impostato per la richiesta di sincronizzazione iniziale.
    - **Opzioni TCP:**
      - **MSS (Maximum Segment Size):** 1460 bytes (indica la grandezza massima dei segmenti TCP).
      - **SACK Permitted (Selective Acknowledgement):** Indica che il client supporta SACK per la gestione degli errori.
      - **Window Scale:** 256 (utilizzato per gestire finestre TCP più ampie se necessario).

Pacchetto 494: Handshake TCP (SYN + ACK)

- **Descrizione:** Il server **193.206.135.26** risponde al pacchetto del client con SYN+ACK, accettando la connessione.
- **Caratteristiche principali:**
  - **Livello 3 (IPv4):**
    - Origine: 193.206.135.26.
    - Destinazione: 10.201.106.166.
    - **Time to Live (TTL):** Indicato in 56, che può fornire un'indicazione sul numero di hop rimanenti.
  - **Livello 4 (TCP):**
    - Porta di origine: 80.
    - Porta di destinazione: 51384 (client).
    - **Flag SYN, ACK:** La sequenza SYN viene riconosciuta e accettata dal server.
    - **Opzioni TCP:**
      - **MSS:** Supportato a 1250 bytes, leggermente inferiore rispetto al client.
      - **SACK Permitted.**
      - **Window Scale:** Presente.
    - **Ack Number:** 1, conferma di aver ricevuto il primo pacchetto del client.

Pacchetto 495: Handshake TCP (ACK)

- **Descrizione:** Il client **10.201.106.166** conferma la risposta SYN+ACK ricevuta dal server, completando così l'handshake TCP a tre vie.
- **Caratteristiche principali:**
  - **Livello 3 (IPv4):**
    - Simile agli altri pacchetti, con destinazione iniziale sul server.
  - **Livello 4 (TCP):**
    - **Flag ACK:** Indica che il client ha ricevuto la risposta del server e la connessione è ora aperta.
    - **Finestra TCP:** Viene pubblicizzata una dimensione del buffer molto ampia (131072 bytes, calcolata con un fattore di scala di 256).

Pacchetto 496: Richiesta HTTP GET

- **Descrizione:** Subito dopo il completamento dell'handshake TCP, il client invia una richiesta HTTP GET per il file **/ncsi.txt** al server.
- **Caratteristiche principali:**
  - **Protocollo HTTP (livello applicativo):**
    - Richiesta: `GET /ncsi.txt HTTP/1.1`.
    - Headers HTTP non inclusi, ma si tratta probabilmente di una richiesta standard di test di connettività (NCSI - Network Connectivity Status Indicator), tipica di sistemi Windows.
  - **Livello 4 (TCP):**
    - **PSH, ACK:** Il flag PSH indica che i dati contenuti devono essere trasferiti al layer applicativo del destinatario immediatamente, senza buffering.

Pacchetto 499: risposta HTTP dal server 193.206.135.26 al client 10.201.106.166

- **Livello 2: Ethernet II**
  - MAC sorgente: `cc:6b:1e:11:41:1b` (server/net device)
  - MAC destinazione: `00:09:0f:09:00:12` (gateway/firewall)
  - Tipo: IPv4 (`0x0800`)
- **Livello 3: IPv4**
  - IP sorgente: `193.206.135.26`
  - IP destinazione: `10.201.106.166`
  - Lunghezza totale: 164 bytes
  - TTL: 128
  - Flags: Don't Fragment (`0x2`)
- **Livello 4: TCP**
  - Porta sorgente: 80 (HTTP)
  - Porta destinazione: 51384
  - Flags: PSH, ACK
  - Payload TCP: 124 bytes
- **Livello 7: HTTP**
  - Status: `HTTP/1.1 200 OK`
  - Content-Type: `text/plain`
  - Body: Contiene 124 byte del file `/ncsi.txt`, usato per verifiche di connettività

Calcolo dei tempi di latenza

1. Tempo di handshake TCP

- Pacchetto 493 (SYN): `21.325595 s`.
- Pacchetto 494 (SYN, ACK): `21.331268 s`.
  - Latenza SYN → SYN-ACK: `21.331268 - 21.325595 = 0.005673 s` (5.673 ms).
- Pacchetto 495 (ACK): `21.331397 s`.
  - Tempo totale handshake: `21.331397 - 21.325595 = 0.005802 s` (5.802 ms).

2. Tempo tra handshake e richiesta HTTP

- Pacchetto 495 (ACK): `21.331397 s`.
- Pacchetto 496 (GET HTTP): `21.331655 s`.
  - Tempo di elaborazione richiesta: `21.331655 - 21.331397 = 0.000258 s` (0.258 ms).

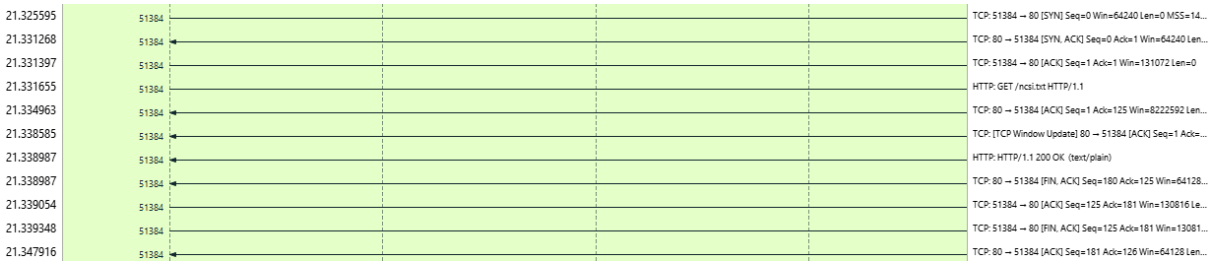
3. Tempo di risposta del server (latenza server-side)

- Pacchetto 496 (GET HTTP): `21.331655 s`.
- Pacchetto 499 (HTTP 200 OK): `21.338987 s`.
  - Tempo di elaborazione server: `21.338987 - 21.331655 = 0.007332 s` (7.332 ms).

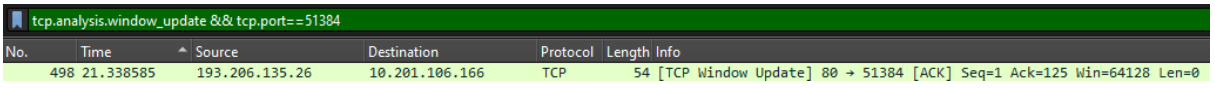
4. Tempo di chiusura TCP

- Pacchetto 501 (FIN, ACK): `21.339054 s`.
- Pacchetto 503 (ACK): `21.347916 s`.
  - Tempo di chiusura connessione: `21.347916 - 21.339054 = 0.008862 s` (8.862 ms).

Controllo del flusso



la seguente cattura ottenuta tramite wireshark dalla sezione Statistiche—>Grafico del flusso permette di analizzare e comprendere la sequenza temporale delle comunicazioni tra host.

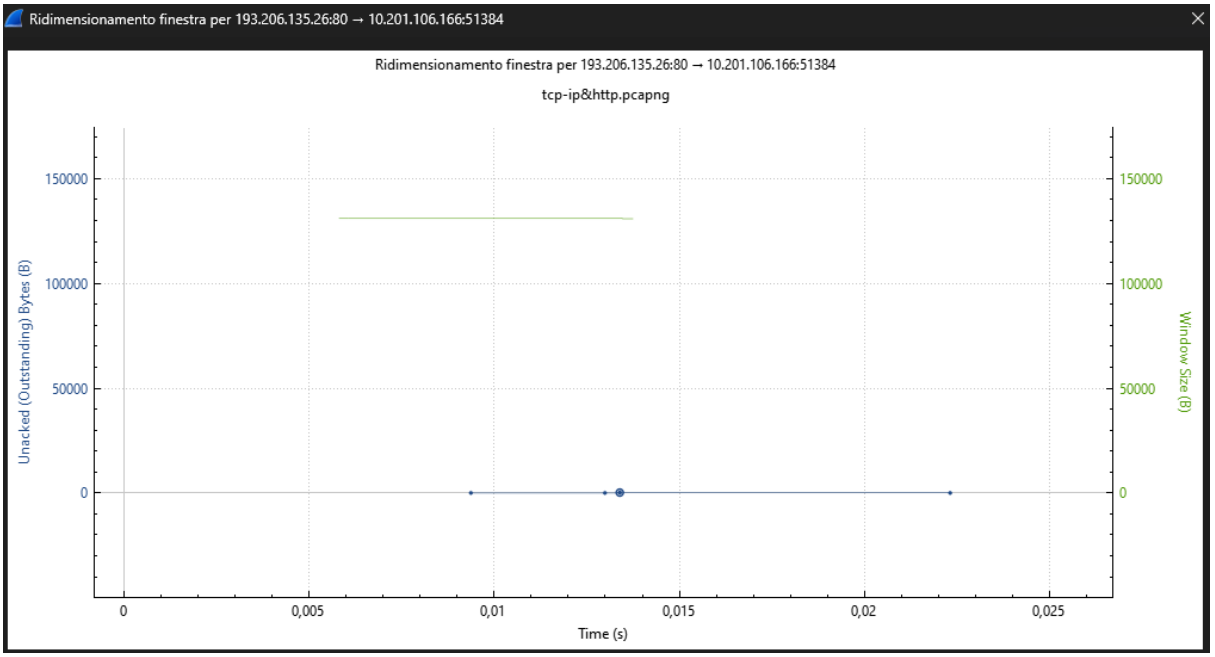


Inoltre tramite il filtro `tcp.analysis.window_update && tcp.port==51384` è possibile osservare che nel pacchetto 498 la Windows size sia diminuita passando da un valore di `64240` ad un valore di `64128` indicando che il buffer del destinatario si sta riempiendo man mano che riceve dati, e di conseguenza la finestra disponibile per il mittente si riduce.

La differenza è di 112 byte rappresenta i dati in transito che il buffer del destinatario ha accumulato. Se la finestra continuasse a ridursi fino a raggiungere 0, ciò implicherebbe che il buffer del destinatario è pieno e non può ricevere altri dati fino a quando non li elabora o li trasferisce. Questa riduzione della finestra serve a regolare la velocità di trasmissione del mittente, in modo che non invii più dati di quanti il destinatario possa gestire.

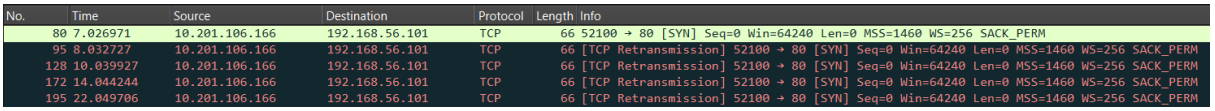
Analisi dettagliata del pacchetto 498:

- **Frame**: 54 byte (432 bit).
- **Ethernet**: indirizzi MAC `Fortinet_09:00:12` e `CloudNetwork_11:41:1b`, tipo IPv4.
- **IPv4**:
  - **Fonte**: 193.206.135.26, **Destinazione**: 10.201.106.166.
  - **TTL**: 56, protocollo TCP.
  - **Flags**: "Don't Fragment" attivo.
- **TCP**:
  - **Porta sorgente**: 80 (HTTP), **Porta destinazione**: 51384.
  - **Sequence Number**: 1, **Acknowledgment Number**: 125.
  - **Flag**: ACK (conferma della ricezione).
  - **Window size**: 501, **Window size scalato**: 64128 byte (capacità di ricezione aggiornata).



la seguente immagine ottenuta tramite wireshark nella sezione *Statistiche—>Grafici dei flussi TCP —>Ridimensionamento della finestra* permette di visualizzare l'andamento della **finestra di ricezione TCP** durante una sessione di comunicazione tra due host.

Analisi di un Tentativo di Connessione HTTP con Ritrasmissioni TCP Ripetitive



No.	Time	Source	Destination	Protocol	Length	Info
80	7.026971	10.201.106.166	192.168.56.101	TCP	66	52100 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
95	8.032727	10.201.106.166	192.168.56.101	TCP	66	[TCP Retransmission] 52100 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
128	10.039927	10.201.106.166	192.168.56.101	TCP	66	[TCP Retransmission] 52100 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
172	14.044244	10.201.106.166	192.168.56.101	TCP	66	[TCP Retransmission] 52100 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
195	22.049706	10.201.106.166	192.168.56.101	TCP	66	[TCP Retransmission] 52100 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM

Descrizione generale della cattura

Questa cattura mostra una serie di tentativi di connessione TCP da parte di un client (**10.201.106.166**) verso un server (**192.168.56.101**) sulla porta **80** (HTTP). Il primo **pacchetto SYN** non riceve risposta dal server, provocando **ritrasmissioni TCP ripetute** (segnalate come "Retransmission").

Analisi dettagliata

Metodologia di analisi: *L'analisi delle ritrasmissioni SYN è stata condotta utilizzando il filtro Wireshark* `tcp.flags.syn == 1 && tcp.flags.ack == 0`*. Questo filtro ha permesso di isolare i pacchetti SYN inviati dal client. Successivamente, l'analisi si è concentrata sull'individuazione di pacchetti SYN identici (stesso numero di sequenza) con tempi di arrivo significativamente diversi, indicando quindi le ritrasmissioni. Il tempo trascorso tra le ritrasmissioni è stato misurato per determinare il comportamento di backoff TCP."*

1. Primo Tentativo: Pacchetto SYN

- **Pacchetto iniziale (#80)**:
  - Il client invia un pacchetto SYN per iniziare un handshake TCP a tre vie.
  - Viene configurato con:
    - **Window Size:** 64240 (capacità del buffer TCP lato client).
    - **MSS (Maximum Segment Size):** 1460 byte.
    - **SACK (Selective Acknowledgment):** abilitato.
- **Problema:** Nessuna risposta dal server entro il timeout stimato.

---

2. Ritrasmissioni del Pacchetto SYN

Dopo il primo **pacchetto SYN (#80)**, il client tenta di ritrasmettere lo stesso pacchetto SYN in quattro occasioni differenti, come indicato dai pacchetti #95, #128, #172 e #195.

- **Caratteristiche delle ritrasmissioni:**
  - **Ritrasmissione 1 (#95):** Effettuata dopo **1.005756 secondi** (delta rispetto al pacchetto iniziale).
  - **Ritrasmissione 2 (#128):** Effettuata dopo ulteriori **2.007200 secondi** (delta dal precedente pacchetto).
  - **Ritrasmissione 3 (#172):** Effettuata dopo **4.004317 secondi** dal pacchetto precedente.
  - **Ritrasmissione 4 (#195):** Effettuata dopo **8.005462 secondi** dal precedente pacchetto.

3. Timeout cumulativo

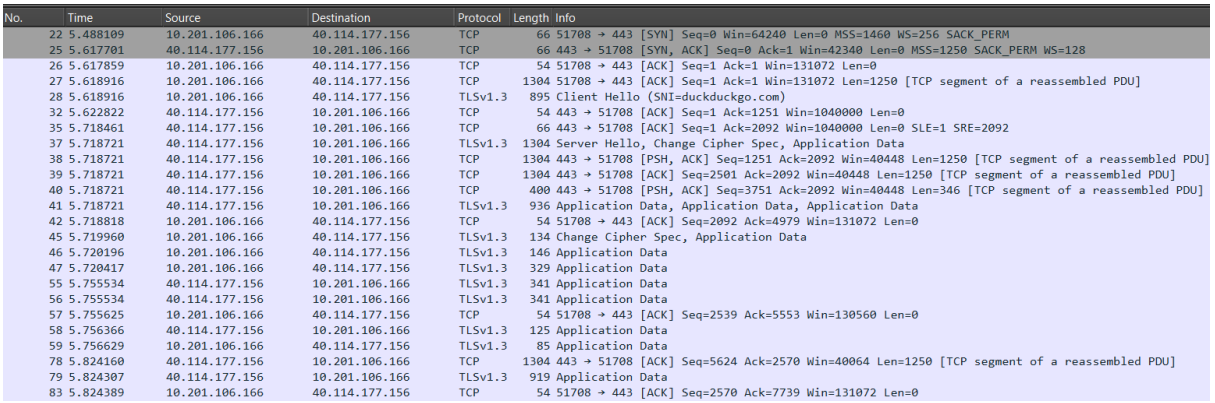
Nel complesso, il ritardo fra il primo pacchetto SYN (#80 - tempo 7.026971) e l'ultimo ritrasmesso (#195 - tempo 22.049706) è di circa **15.022735 secondi**, indicando un tentativo fallito di instaurare una connessione con il server.

Metriche di analisi temporale

- **Backoff TCP:**
  Il client utilizza un'implementazione del
  TCP Exponential Backoff per incrementare i tempi di attesa prima di ritrasmettere.
  - Ritrasmissione 1: 1.005756 secondi dopo il pacchetto iniziale.
  - Ritrasmissione 2: ~2 secondi dopo la precedente.
  - Ritrasmissione 3: ~4 secondi dopo la precedente.
  - Ritrasmissione 4: ~8 secondi dopo la precedente.

Questo comportamento è standard nel protocollo TCP.

Analisi del traffico TCP e TLS durante una connessione HTTPS



No.	Time	Source	Destination	Protocol	Length	Info
22	5.488109	10.201.106.166	40.114.177.156	TCP	66	51708 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
25	5.617701	40.114.177.156	10.201.106.166	TCP	66	443 → 51708 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1250 SACK_PERM WS=128
26	5.617859	10.201.106.166	40.114.177.156	TCP	54	51708 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
27	5.618916	10.201.106.166	40.114.177.156	TCP	1304	51708 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=1250 [TCP segment of a reassembled PDU]
28	5.618916	10.201.106.166	40.114.177.156	TLSv1.3	895	Client Hello (SNI=duckduckgo.com)
32	5.622822	40.114.177.156	10.201.106.166	TCP	54	443 → 51708 [ACK] Seq=1 Ack=1251 Win=1040000 Len=0
35	5.718461	40.114.177.156	10.201.106.166	TCP	66	443 → 51708 [ACK] Seq=1 Ack=2092 Win=1040000 Len=0 SLE=1 SRE=2092
37	5.718721	40.114.177.156	10.201.106.166	TLSv1.3	1304	Server Hello, Change Cipher Spec, Application Data
38	5.718721	40.114.177.156	10.201.106.166	TCP	1304	443 → 51708 [PSH, ACK] Seq=1251 Ack=2092 Win=40448 Len=1250 [TCP segment of a reassembled PDU]
39	5.718721	40.114.177.156	10.201.106.166	TCP	1304	443 → 51708 [ACK] Seq=2501 Ack=2092 Win=40448 Len=1250 [TCP segment of a reassembled PDU]
40	5.718721	40.114.177.156	10.201.106.166	TCP	400	443 → 51708 [PSH, ACK] Seq=3751 Ack=2092 Win=40448 Len=346 [TCP segment of a reassembled PDU]
41	5.718721	40.114.177.156	10.201.106.166	TLSv1.3	936	Application Data, Application Data, Application Data
42	5.718818	10.201.106.166	40.114.177.156	TCP	54	51708 → 443 [ACK] Seq=2092 Ack=4979 Win=131072 Len=0
45	5.719960	10.201.106.166	40.114.177.156	TLSv1.3	134	Change Cipher Spec, Application Data
46	5.720196	10.201.106.166	40.114.177.156	TLSv1.3	146	Application Data
47	5.720417	10.201.106.166	40.114.177.156	TLSv1.3	329	Application Data
55	5.755534	40.114.177.156	10.201.106.166	TLSv1.3	341	Application Data
56	5.755534	40.114.177.156	10.201.106.166	TLSv1.3	341	Application Data
57	5.755625	10.201.106.166	40.114.177.156	TCP	54	51708 → 443 [ACK] Seq=2539 Ack=5553 Win=130560 Len=0
58	5.756366	40.114.177.156	10.201.106.166	TLSv1.3	125	Application Data
59	5.756629	10.201.106.166	40.114.177.156	TLSv1.3	85	Application Data
78	5.824160	40.114.177.156	10.201.106.166	TCP	1304	443 → 51708 [ACK] Seq=5624 Ack=2570 Win=40064 Len=1250 [TCP segment of a reassembled PDU]
79	5.824307	40.114.177.156	10.201.106.166	TLSv1.3	919	Application Data
83	5.824389	10.201.106.166	40.114.177.156	TCP	54	51708 → 443 [ACK] Seq=2570 Ack=7739 Win=131072 Len=0

Descrizione generale della cattura

La cattura mostra una serie di pacchetti che rappresentano una connessione TCP stabilita tra un client (**10.201.106.166**) e un server (**40.114.177.156**) sulla porta **443** (HTTPS). La connessione include l'handshake TCP, la negoziazione TLS 1.3 e il trasferimento di dati cifrati

1. Handshake TCP

Pacchetto 22: SYN

Descrizione: Il client invia un pacchetto SYN per iniziare la connessione TCP.

Dettagli principali:

- **Src MAC:** CloudNetwork_11:41:1b (cc:6b:1e:11:41:1b)
- **Dst MAC:** Fortinet_09:00:12 (00:09:0f:09:00:12)
- **Source (porta sorgente)**: `10.201.106.166:51708`
- **Destination (porta di destinazione)**: `40.114.177.156:443`
- **Flags**: `SYN` (synchronize) - indica l'inizio della connessione.
- **Sequence Number**: `0` - il client inizia la comunicazione con questo numero di sequenza.
- **Window Size**: `64240` - indica la capacità di buffer disponibile sul client.
- **MSS (Maximum Segment Size)**: `1460` - suggerisce la dimensione massima di un segmento TCP per questa connessione.

Questo pacchetto avvia il processo di negoziazione TCP.

Pacchetto 25: SYN-ACK

- **Source (porta sorgente)**: `40.114.177.156:443`
- **Destination (porta di destinazione)**: `10.201.106.166:51708`
- **Flags**: `SYN, ACK` - il server accetta la richiesta e invia una conferma.
- **Sequence Number**: `0` - il server inizia il proprio numero di sequenza.
- **Acknowledgment Number**: `1` - conferma la ricezione del pacchetto SYN dal client.
- **Window Size**: `42340` - capacità del buffer lato server.

Il server risponde con un SYN-ACK, confermando che è pronto per stabilire la connessione.

Pacchetto 26: ACK

- **Source**: `10.201.106.166:51708`
- **Destination**: `40.114.177.156:443`
- **Flags**: `ACK` - il client conferma la ricezione del SYN-ACK.
- **Sequence Number**: `1` - il client incrementa il numero di sequenza.
- **Acknowledgment Number**: `1` - conferma il numero di sequenza del server.

Conclusione del three-way handshake: La connessione TCP è stata stabilita con successo e il client è pronto per inviare dati.

---

2. Handshake TLS v1.3

Il protocollo **TLS (Transport Layer Security)** è utilizzato per criptare i dati e garantire la sicurezza della connessione HTTPS. Vediamo i dettagli della negoziazione TLS:

Pacchetto 27: Client Hello

- **Protocollo**: Ethernet → IP → TCP → TLSv1.3.
- **Descrizione**: Questo pacchetto è il primo nella sequenza di handshake TLS. Il client invia le sue preferenze di sicurezza al server.
- **Informazioni principali**:
  - **Cipher Suite**: Una lista dei cifrari supportati dal client.

    - **Random**: Un valore casuale utilizzato per generare chiavi crittografiche.
    - **SNI (Server Name Indication)**: Specifica il nome del server richiesto .
    - **Extensions**: Informazioni aggiuntive per migliorare la sicurezza.
- **Lunghezza totale del frame:** 895 byte.
- **Dettagli IP e TCP:**
    - **Src IP:** 10.201.106.166 (client interno alla rete).
    - **Dst IP:** 40.114.177.156 (server remoto).
    - **Porta sorgente:** 51708.
    - **Porta destinazione:** 443 (HTTPS).
    - **Dimensione del payload TCP:** 841 byte.

Il messaggio **Client Hello** include:

- La lista di cifrari supportati dal client.
- La versione del protocollo TLS utilizzata
- Il "Random" del client (un valore casuale usato per la sicurezza nella negoziazione).
- Le estensioni come SNI (Server Name Indication), che permette al client di indicare il nome del server richiesto.

Pacchetto 32:

- **Tipo:** ACK
- **Descrizione:** Il server invia un ACK per confermare la ricezione del Client Hello.
- **Dettagli principali:**
    - **Src MAC:** (00:09:0f:09:00:12)
    - **Dst MAC:** (cc:6b:1e:11:41:1b)
    - **Src IP:** 40.114.177.156
    - **Dst IP:** 10.201.106.166
    - **Src Port:** 443
    - **Dst Port:** 51708
    - **Seq:** 1

- **Ack:** 1251
- **Flags:** ACK

Pacchetto 35:

- **Tipo:** ACK
- **Descrizione:** Il server invia un ulteriore ACK per confermare la ricezione completa del Client Hello.
- **Dettagli principali:**
  - **Src MAC:** (24:81:3b:13:48:2e)
  - **Dst MAC:** (cc:6b:1e:11:41:1b)
  - **Src IP:** 40.114.177.156
  - **Dst IP:** 10.201.106.166
  - **Src Port:** 443
  - **Dst Port:** 51708
  - **Seq:** 1
  - **Ack:** 2092
  - **Flags:** ACK

Pacchetto 37: Server Hello

- **Protocollo**: `TLSv1.3`.
- **Descrizione**: Il server risponde al `Client Hello` accettando una configurazione di sicurezza.
- **Informazioni principali**:
  - **Cipher Suite**: Il server seleziona il cifrario da utilizzare.
  - **Extensions**: Indica il supporto per parametri avanzati come chiavi condivise o compressione.
  - La versione del protocollo TLS accettata.
  - Il "Random" del server.
- **Lunghezza totale del frame:** 1304 byte.
- **Dettagli IP e TCP:**

- **Src IP:** 40.114.177.156 (server remoto).
- **Dst IP:** 10.201.106.166 (client).
- **Porta sorgente:** 443 (HTTPS).
- **Porta destinazione:** 51708 (porta del client).
- **Dimensione del payload TCP:** 1250 byte.

Pacchetto 38-41: Change Cipher Spec

- **Descrizione**: Questi pacchetti indicano che il server ha completato la negoziazione TLS e ha abilitato la cifratura per i messaggi successivi.
- **Dati importanti**:
  - **Server Finished**: Il server invia un hash crittografico per confermare che l'handshake è stato completato senza alterazioni.

Pacchetto 45: Client Finished

- **Descrizione**: Il client invia il proprio hash crittografico per confermare che accetta i parametri negoziati.
- **Protocols:** TCP, TLS
- **Transport Layer Security:** Dati applicativi cifrati

Da questo punto in poi, tutti i dati trasmessi sono cifrati.

---

3. Trasmissione di dati crittografati

I pacchetti seguenti trasportano dati crittografati utilizzando il protocollo TLS. Questi pacchetti sono identificati come **Application Data**:

Pacchetti 55-78: Application Data

- **Protocollo**: `TLSv1.3`.
- **Descrizione**: Questi pacchetti contengono i dati trasferiti tra il client e il server.
- **Informazioni tecniche**:
  - **Sequence Number**: Aggiornato per ogni pacchetto inviato.
  - **Acknowledgment Number**: Riconosce i dati ricevuti dall'altra parte.
  - **Window Size**: Aggiornato dinamicamente per ottimizzare il controllo di flusso.

Esempi di dati che potrebbero essere trasferiti:

- Richieste HTTPS (GET, POST, ecc.).
- Risposte del server, come pagine web o risorse.

---

4. Controllo di flusso e gestione della connessione

Durante tutta la cattura, il TCP gestisce il flusso dei dati:

- **Finestra di ricezione (Window Size)**:
  - Viene aggiornata dinamicamente per indicare quanto spazio è disponibile nel buffer del ricevitore.
- **Riconoscimenti (ACK)**:
  - Ogni pacchetto ricevuto è confermato con un `ACK`, garantendo l'affidabilità della trasmissione.

Non ci sono pacchetti persi o duplicati nella sequenza osservata, il che indica una trasmissione stabile.

---

5. Considerazioni chiave

- **Correttezza dell'handshake TCP**:
  - Segue la sequenza standard `SYN → SYN-ACK → ACK`.
- **Sicurezza dell'handshake TLS**:
  - È stata negoziata correttamente una connessione TLS v1.3.
- **Trasferimento dati**:
  - I pacchetti TLS contengono dati crittografati; il contenuto reale non è visibile senza la chiave di sessione.

Analisi dei tempi di latenza

Analizziamo i delta di tempo chiave per valutare la latenza nelle varie fasi:

Connessione TCP

- **Delta tra SYN (pacchetto 22) e SYN-ACK (pacchetto 25):**
  - Tempo: 5.617701−5.488109=0.129592secondi.

    5.617701−5.488109=0.129592 secondi5.617701 - 5.488109 = 0.129592 secondi.
  - **Descrizione:** Questo rappresenta il tempo necessario al server per rispondere alla richiesta iniziale di connessione TCP.
- **Delta tra SYN-ACK (pacchetto 25) e ACK (pacchetto 26):**
  - Tempo: 5.617859−5.617701=0.000158secondi.

    5.617859−5.617701=0.000158 secondi5.617859 - 5.617701 = 0.000158 secondi.
  - **Descrizione:** Questo tempo rappresenta la conferma del client.

Handshake TLS

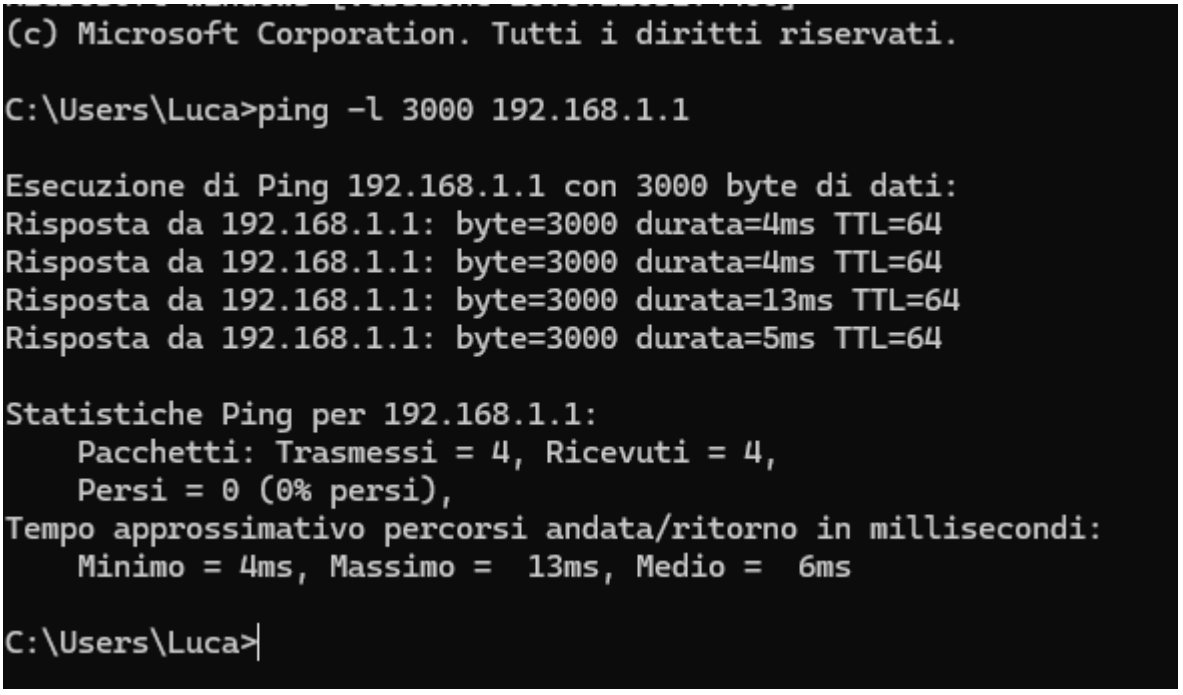
- **Delta tra ACK (pacchetto 26) e Client Hello (pacchetto 28):**
  - Tempo: 5.618916−5.617859=0.001057secondi.
  - **Descrizione:** Ritardo minimo, tipico per il passaggio immediato a TLS.
- **Delta tra Client Hello (pacchetto 28) e Server Hello (pacchetto 35):**
  - Tempo: 5.718461−5.618916=0.099545secondi.
  - **Descrizione:** Questo rappresenta il tempo di risposta del server per negoziare il TLS.

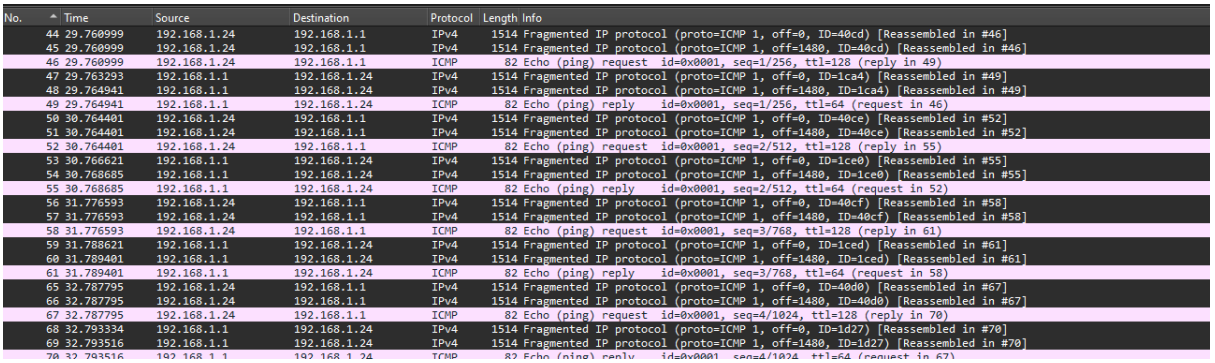
Cambio di cifrario e Application Data

- **Delta tra Server Hello (pacchetto 35) e Change Cipher Spec (pacchetto 45):**
  - Tempo: 5.719960−5.718461=0.001499secondi.
  - **Descrizione:** Minimo ritardo tra il completamento dell'handshake e l'inizio della comunicazione cifrata.
- **Delta tra Change Cipher Spec (pacchetto 45) e il primo Application Data (pacchetto 55):**
  - Tempo: 5.755534−5.719960=0.035574secondi.
  - **Descrizione:** Ritardo minimo nella trasmissione dei dati cifrati.

Analisi di una Comunicazione ICMP con Frammentazione IP

Per ottenere un esempio di frammentazione IP, ho eseguito il comando `ping -l 3000 192.168.1.1`. Questo ha generato pacchetti ICMP frammentati a livello IP, poiché la dimensione specificata (3000 byte) supera l'MTU standard, solitamente impostato a 1500 byte





Metodologia di analisi:

per ottenere la seguente cattura è stato applicato il filtro `ip.flags.mf == 1 || ip.frag_offset > 0` *è un filtro utilizzato in Wireshark per identificare pacchetti IP frammentati.* `ip.flags.mf == 1` *: Filtra i pacchetti che sono frammenti non ultimi, ovvero che hanno il flag "More Fragments" (MF) impostato su 1, indicando che ci sono altri frammenti dopo questo.* `ip.frag_offset > 0` *: Filtra i pacchetti che non sono i primi frammenti di una serie di frammenti, indicando che la posizione del frammento all'interno della sequenza è diversa da zero.*

Analisi Dettagliata dei Pacchetti

Pacchetto 44

- **Tipo:** Primo frammento di una richiesta ICMP frammentata (parte della sequenza).
- **Dimensioni:**
  - Lunghezza **1514 byte**, inclusi **20 byte di header IP**.
  - Payload ICMP: **1480 byte**.
- **Indirizzi:**
  - **Sorgente:** `192.168.1.24`
  - **Destinazione:** `192.168.1.1`
- **Fragmentazione:**
  - Questo è il **primo frammento** del pacchetto.
  - Il flag "More fragments" è impostato (indica che ci sono frammenti successivi).
  - Offset = 0.
- **Protocollo:** ICMP identificato al livello IP (Protocol Number: 1).
- **Note:** Parte del pacchetto completo che verrà riassemblato nel **frame 46**.

Pacchetto 45

- **Tipo:** Secondo frammento del pacchetto ICMP frammentato (stessa sequenza del pacchetto 44).
- **Dimensioni:**
  - Lunghezza **1514 byte** (totale identico al pacchetto 44).
  - Payload ICMP: **1480 byte**.
- **Indirizzi:**
  - **Sorgente:** `192.168.1.24`
  - **Destinazione:** `192.168.1.1`
- **Fragmentazione:**
  - **Secondo frammento** della sequenza di tre.
  - Offset = 1480 (indica l’inizio di questo frammento).
  - Il flag "More fragments" è impostato (indica che ne esiste un frammento successivo).
- **Protocollo:** ICMP nel livello IP.

Contribuisce al riassemblaggio completo nel **frame 46**.

Pacchetto 46

- **Tipo:** Terzo frammento di una richiesta ICMP, parte finale della sequenza frammentata (completa il riassemblaggio).
- **Dimensioni:**
  - Lunghezza **82 byte**, inclusi header Ethernet e IP.
  - Payload ICMP: **48 byte**.
- **Indirizzi:**
  - **Sorgente:** `192.168.1.24`
  - **Destinazione:** `192.168.1.1`

- **Fragmentazione:**
  - **Ultimo frammento** della sequenza.
  - Offset = 2960 (inizia alla posizione 2960 nel payload originale).
  - **Il flag "More fragments" non è impostato**, indicando la fine del pacchetto frammentato.
- **Protocollo:** ICMP, identificato dal livello IP.

Risposta ICMP (Pacchetti 47-49)

Pacchetto 47:

- **Tipo**: Primo frammento di una risposta ICMP frammentata
- **Dimensioni**: 1514 byte (12112 bit)
- **Indirizzi**:
  - **Sorgente**: 192.168.1.1
  - **Destinazione**: 192.168.1.24
- **Fragmentazione**:
  - **Flags**: More fragments (MF = 1)
  - **Fragment Offset**: 0 (primo frammento)
  - **Lunghezza totale**: 1500 byte (dati 1480 byte)
- **Protocollo**: ICMP (ping)

Pacchetto 48:

- **Tipo**: Secondo frammento del pacchetto ICMP frammentato (stessa sequenza del pacchetto 47)
- **Dimensioni**: 1514 byte (12112 bit)
- **Indirizzi**:
  - **Sorgente**: 192.168.1.1
  - **Destinazione**: 192.168.1.24
- **Fragmentazione**:
  - **Flags**: More fragments (MF = 1)
  - **Fragment Offset**: 1480 (secondo frammento)
  - **Lunghezza totale**: 1500 byte (dati 1480 byte)
- **Protocollo**: ICMP (ping)

Pacchetto 49:

- **Tipo**: Terzo frammento di una richiesta ICMP, parte finale della sequenza frammentata (completa il riassemblaggio).
- **Dimensioni**: 82 byte (656 bit)
- **Indirizzi**:
  - **Sorgente**: 192.168.1.1
  - **Destinazione**: 192.168.1.24
- **Fragmentazione**:
  - **Flags**: No more fragments (MF = 0)
  - **Fragment Offset**: 2960 (ultimo frammento)
  - **Lunghezza totale**: 68 byte (dati 48 byte)
- **Protocollo**: ICMP (ping)

I pacchetti 47, 48 e 49 rappresentano la risposta ICMP (un Echo Reply) dalla destinazione (192.168.1.1) alla sorgente (192.168.1.24). Anche in questo caso, la risposta è stata frammentata:

Pacchetto 47 e 48: Sono i primi due frammenti della risposta ICMP, con dimensioni e offset analoghi a quelli della richiesta iniziale (pacchetti 44 e 45).

Pacchetto 49: Ultimo frammento della risposta ICMP.

Similmente alla richiesta, il destinatario (192.168.1.24) riassembla questi tre frammenti per ricevere la risposta ICMP completa.

L'analisi approfondita si concentra sui pacchetti 44, 45 ,46 ,47,48 e 49 in quanto rappresentano un esempio del pattern ricorrente osservato nella cattura, con gli altri pacchetti che mostrano un comportamento molto simile sia in termini di frammentazione che di gestione delle richieste e risposte ICMP.

Guida all'utilizzo

All'interno della repository sono disponibili i file relativi alle tre catture analizzate, ciascuno associato a una specifica sessione di traffico di rete. Di seguito, vengono descritte le catture e le istruzioni per il loro utilizzo in Wireshark.

---

1. tcp-ip&http.pcapng

- **Descrizione del file:** Questo file contiene la cattura analizzata nella sezione *"Interpretazione di una Cattura Packet Capture: Protocollo TCP/IP e HTTP"*.
- **Filtro da applicare:** Per esaminare tutti i pacchetti analizzati nel paragrafo, applicare in Wireshark il seguente filtro permette di isolare i pacchetti relativi alla connessione tra il client e il server sulla porta TCP **51384**.

```
tcp.port == 51384
```

---

2. retrasmission.pcapng

- **Descrizione del file:** Questa cattura corrisponde all'analisi della sezione *"Analisi di un Tentativo di Connessione HTTP con Ritrasmissioni TCP Ripetitive"*. Rappresenta la situazione in cui un client tenta più volte di stabilire una connessione TCP senza successo.
- **Filtro da applicare:** Per individuare i pacchetti ritrasmessi, utilizzare il filtro seguente consente di isolare i pacchetti SYN inviati senza ricevere risposta.

```
tcp.flags.syn == 1 && tcp.flags.ack == 0
```

- **Nota aggiuntiva:** In questa analisi, ci si è focalizzati sui pacchetti connessi alla porta TCP **52100**. È quindi possibile affinare ulteriormente il filtro:

```
tcp.port == 52100
```

---

3. https-tcp&tls.pcapng

- **Descrizione del file:** Questo file è legato all'analisi descritta nella sezione *"Analisi del traffico TCP e TLS durante una connessione HTTPS"*. Rappresenta il traffico cifrato tra il client e il server che sfrutta il protocollo TLS 1.3 per garantire la sicurezza.
- **Filtro da applicare:** Per visualizzare tutti i pacchetti analizzati, utilizzare il seguente filtro permette di isolare il traffico HTTPS sulla porta TCP **51708**.

```
tcp.port == 51708
```

4.frammentazione.pcapng

- **Descrizione del file:** Questo file è legato all'analisi descritta nella sezione *"Analisi di una Comunicazione ICMP con Frammentazione IP"*.
- **Filtro da applicare:** Per visualizzare tutti i pacchetti frammentati e analizzati , utilizzare il seguente filtro:

```
ip.flags.mf == 1 || ip.frag_offset > 0
```

Indicazioni Generali

Nella relazione sono stati inclusi solo gli screenshot delle catture principali per garantire una presentazione chiara e concisa. Tuttavia, grazie al paragrafo "guida all'utilizzo" e all'applicazione dei filtri specifici, è possibile analizzare tutte le proprietà dettagliate dei pacchetti catturati, fornendo una visione completa del traffico analizzato.

- Per caricare i file `.pcapng`, è sufficiente selezionare la voce **File** > **Open** nel menu di Wireshark e navigare fino alla directory dei file di cattura.
- Dopo aver applicato i filtri specificati per ogni cattura, è possibile esaminare i dettagli dei pacchetti come descritto nelle rispettive analisi.

Con questa guida e i filtri indicati, sarà possibile replicare le analisi effettuate.