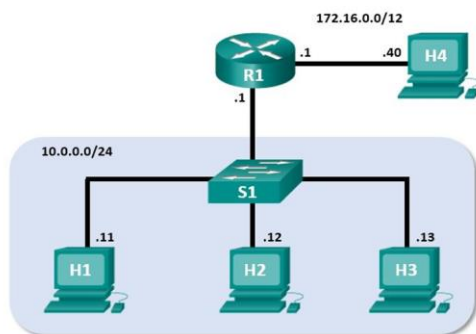


PRATICA S11L2

Usare Wireshark per Osservare l'Handshake a 3 Vie TCP

Topologia Mininet



Risorse Richieste: Macchina virtuale CyberOps Workstation

Obiettivi

- Parte 1: Preparare gli Host per Catturare il Traffico
- Parte 2: Analizzare i Pacchetti usando Wireshark
- Parte 3: Visualizzare i Pacchetti usando tcpdump

Contesto / Scenario

In questo laboratorio, userai Wireshark per catturare ed esaminare i pacchetti generati tra il browser del PC che utilizza il protocollo HTTP (HyperText Transfer Protocol) e un server web, come www.google.com. Quando un'applicazione, come HTTP o FTP (File Transfer Protocol), si avvia per la prima volta su un host, TCP utilizza l'handshake a tre vie per stabilire una sessione TCP affidabile tra i due host. Ad esempio, quando un PC utilizza un browser web per navigare in internet, viene avviato un handshake a tre vie e viene stabilita una sessione tra l'host del PC e il server web. Un PC può avere più sessioni TCP attive simultaneamente con vari siti web.

Preparare gli Host per Catturare il Traffico

Dopo aver avviato la macchina virtuale **CyberOps**, ho inizializzato l'ambiente **Mininet** per predisporre gli host alla cattura del traffico di rete, eseguendo il seguente comando:

- `sudo lab.support.files/scripts/cyberops_topo.py`

```
[analyst@secOps ~]$ sudo lab.support.files/scripts/cyberops_topo.py
[sudo] password for analyst:

CyberOPS Topology:

      -----
      | R1 |-----| H4 |
      -----
      |
      |
      -----
|-----| S1 |-----| | |
|       |   |       |
|       |   |       |
|       |   |       |
|-----|   |-----|
| H1 |   | H2 |   | H3 |
|-----|   |-----|

*** Adding internal links
*** Creating network
*** Adding hosts:
H1 H2 H3 H4 R1
*** Adding switches:
s1
*** Adding links:
(H1, s1) (H2, s1) (H3, s1) (H4, R1) (s1, R1)
*** Configuring hosts
H1 H2 H3 H4 R1 Enabling IP forwarding on R1

*** Starting controller

*** Starting 1 switches
s1 ...
*** Routing Table on Router:

Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
10.0.0.0         0.0.0.0         255.255.255.0   U        0    0    0 R1-eth1
172.16.0.0       0.0.0.0         255.240.0.0     U        0    0    0 R1-eth2

*** Starting CLI:
mininet>
```

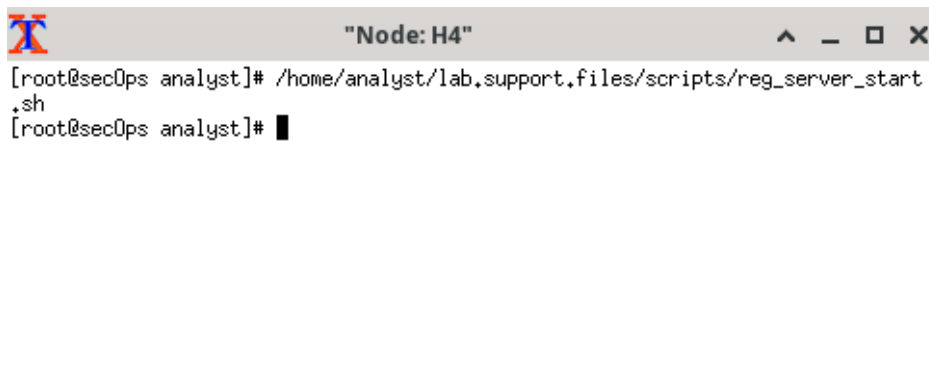
Una volta avviato l'ambiente **Mininet**, ho aperto le console degli **host H1 e H4** tramite i seguenti comandi:

- xterm H1
- xterm H4

```
*** Starting CLI:
mininet> xterm H1
mininet> xterm H4
mininet> █
```

Dalla console dell'host H4, ho avviato il server web eseguendo il seguente script:

- /home/analyst/lab.support.files/scripts/reg_server_start.sh

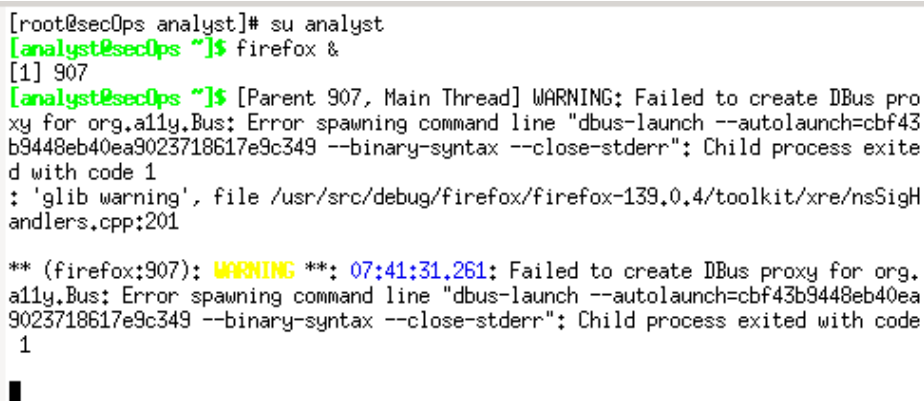


```

Node: H4
[root@secOps analyst]# /home/analyst/lab.support.files/scripts/reg_server_start
.sh
[root@secOps analyst]# █
```

Poiché per motivi di sicurezza non è consentito eseguire **Firefox** come utente root, sull'host **H1** ho utilizzato il comando **su** per effettuare lo switch dall'utente root all'account **analyst**:

- su analyst



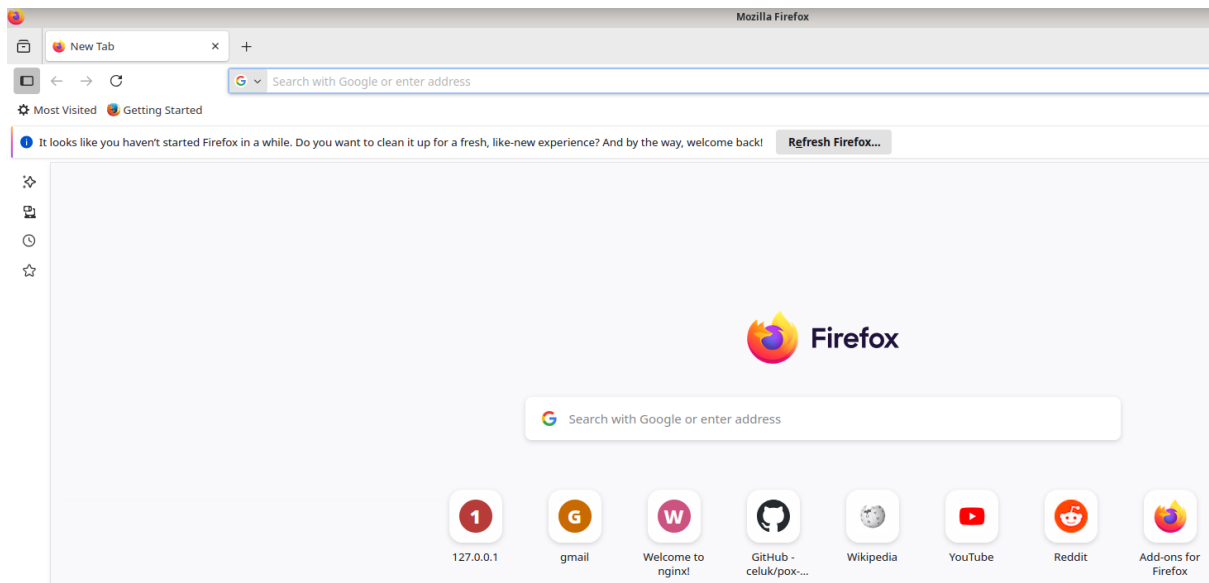
```

[root@secOps analyst]# su analyst
[analyst@secOps ~]$ firefox &
[1] 907
[analyst@secOps ~]$ [Parent 907, Main Thread] WARNING: Failed to create DBus pro
xy for org.a11y.Bus; Error spawning command line "dbus-launch --autolaunch=cbf43
b9448eb40ea9023718617e9c349 --binary-syntax --close-stderr"; Child process exite
d with code 1
; 'glib warning', file /usr/src/debug/firefox/firefox-139.0.4/toolkit/xre/nsSigh
andlers.cpp:201

** (firefox:907): WARNING **: 07:41:31.261: Failed to create DBus proxy for org.
a11y.Bus; Error spawning command line "dbus-launch --autolaunch=cbf43b9448eb40ea
9023718617e9c349 --binary-syntax --close-stderr"; Child process exited with code
1
█
```

Dopo aver effettuato lo switch all'utente **analyst** sull'host **H1**, ho avviato il browser web **Firefox** tramite il seguente comando:

- `firefox &`



Dopo aver aperto la finestra di Firefox sull'host H1, ho avviato una sessione di cattura del traffico con **tcpdump**, inviando l'output al file **capture.pcap**. Ho utilizzato l'opzione **-v** per monitorare l'avanzamento in tempo reale e ho limitato la cattura a 50 pacchetti tramite l'opzione **-c 50**:

- `sudo tcpdump -i H1-eth0 -v -c 50 -w /home/analyst/capture.pcap`

```
[analyst@secOps ~]$ sudo tcpdump -i H1-eth0 -v -c 50 -w /home/analyst/capture.pcap
cap
[sudo] password for analyst:
tcpdump: listening on H1-eth0, link-type EN10MB (Ethernet), snapshot length 2621
44 bytes
50 packets captured
54 packets received by filter
0 packets dropped by kernel
[analyst@secOps ~]$
```

Una volta avviata la cattura, ho navigato rapidamente all'indirizzo IP **172.16.0.40** tramite il browser **Firefox**, in modo da generare traffico utile per l'analisi.

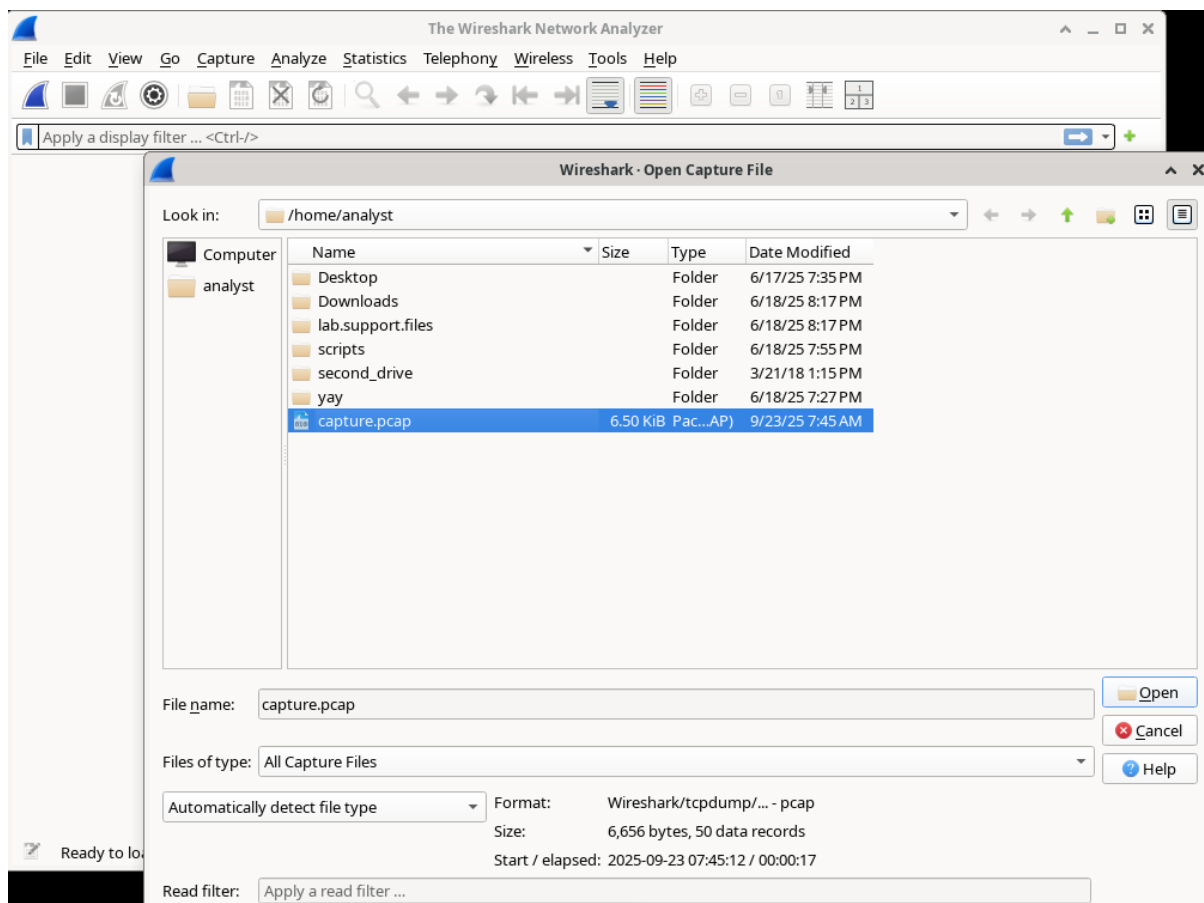
Analizzare i Pacchetti usando Wireshark

Sull'host H1 ho avviato **Wireshark** per l'analisi del traffico di rete. Al prompt relativo all'esecuzione come superutente, ho confermato cliccando su "OK" per proseguire con i privilegi richiesti.

- `wireshark-gtk &`

```
[analyst@secOps ~]$ wireshark-gtk &
[2] 1438
[analyst@secOps ~]$ qt.multimedia.symbolsresolver: Couldn't load pipewire-0.3 library
qt.multimedia.symbolsresolver: Couldn't resolve pipewire-0.3 symbols
** (wireshark:1438) 07:49:08.268676 [GUI WARNING] -- Session Dbus not running.
** (wireshark:1438) 07:49:08.269497 [GUI WARNING] -- Application will not react to setting changes.
Check your Dbus installation.
** (wireshark:1438) 07:49:10.313760 [Capture WARNING] /usr/src/debug/wireshark/wireshark-4.4.7/ui/capture.c:1019 -- capture_interface_stat_start(): Couldn't run dumpcap in child process: Permission denied
```

All'interno di Wireshark, ho cliccato su File > Open per aprire il file di cattura precedentemente salvato. Ho selezionato il file **capture.pcap**.



All'interno di Wireshark, ho applicato un filtro tcp alla cattura per isolare il traffico di interesse. In questo scenario, i primi tre frame corrispondono al classico handshake a tre vie TCP tra client e server, fondamentale per l'instaurazione della connessione.

No.	Time	Source	Destination	Protocol	Length	Info
24	12.053452	10.0.0.11	172.16.0.40	TCP	74	48446 → 80 [SYN, ACK] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=0 TSval=2124196538 TSecr=0 WS=512
25	12.053458	172.16.0.40	10.0.0.11	TCP	74	80 → 48446 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0 MSS=1460 SACK_PERM=0 TSval=3942472270 TSecr=2124196538 WS=512
26	12.053459	10.0.0.11	172.16.0.40	TCP	66	48446 → 80 [ACK] Seq=1 Ack=1 Win=42496 Len=0 TSval=2124196538 TSecr=3942472270
27	12.053589	10.0.0.11	172.16.0.40	HTTP	397	GET / HTTP/1.1

Esaminare le informazioni all'interno dei pacchetti, inclusi indirizzi IP, numeri di porta TCP e flag di controllo TCP.

Nel mio esempio, il frame 1 rappresenta l'inizio dell'**handshake a tre vie** tra il PC e il server sull'host H4. Dalla finestra principale di **Wireshark**, ho selezionato il primo pacchetto nel riquadro superiore (elenco dei pacchetti).

Nel riquadro dei dettagli del pacchetto, ho cliccato sulla freccia accanto a **“Transmission Control Protocol”** per espandere la sezione e analizzare le informazioni TCP. Ho individuato la porta di origine e quella di destinazione per confermare l'inizio della comunicazione.

Successivamente, ho espanso la sezione **“Flags”** cliccando sulla freccia a sinistra. Un valore pari a 1 indica che il flag è attivo: in questo pacchetto ho localizzato il flag impostato, che conferma la fase iniziale dell'handshake.

No.	Time	Source	Destination	Protocol	Length	Info
24	12.053452	10.0.0.11	172.16.0.40	TCP	74	48446 → 80 [SYN, ACK] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=0 TSval=2124196538 TSecr=0 WS=512
25	12.053458	172.16.0.40	10.0.0.11	TCP	74	80 → 48446 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0 MSS=1460 SACK_PERM=0 TSval=3942472270 TSecr=2124196538 WS=512
26	12.053459	10.0.0.11	172.16.0.40	TCP	66	48446 → 80 [ACK] Seq=1 Ack=1 Win=42496 Len=0 TSval=2124196538 TSecr=3942472270
27	12.053589	10.0.0.11	172.16.0.40	HTTP	397	GET / HTTP/1.1
28	12.053521	172.16.0.40	10.0.0.11	TCP	66	80 → 48446 [ACK] Seq=332 Win=43520 Len=0 TSval=3942472270 TSecr=2124196538
29	12.053588	172.16.0.40	10.0.0.11	TCP	384	80 → 48446 [PSH, ACK] Seq=1 Ack=332 Win=43520 Len=238 TSval=3942472270 TSecr=2124196538 [TCP PDU reassembled in 31]
30	12.053592	10.0.0.11	172.16.0.40	TCP	66	48446 → 80 [ACK] Seq=332 Ack=239 Win=42496 Len=0 TSval=2124196538 TSecr=3942472270
31	12.053611	172.16.0.40	10.0.0.11	HTTP	681	HTTP/1.1 200 OK (text/html)
32	12.053613	10.0.0.11	172.16.0.40	TCP	66	48446 → 80 [ACK] Seq=332 Ack=854 Win=41984 Len=0 TSval=2124196538 TSecr=3942472270
45	12.231111	10.0.0.11	172.16.0.40	HTTP	411	GET /favicon.ico HTTP/1.1
46	12.231282	172.16.0.40	10.0.0.11	HTTP	374	HTTP/1.1 404 Not Found (text/html)

Frame 24: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

Ethernet II, Src: 12:8c:c7:61:4a:40 (12:8c:c7:61:4a:40), Dst: ca:73:47:c9:6f:62 (ca:73:47:c9:6f:62)

Internet Protocol Version 4, Src: 10.0.0.11, Dst: 172.16.0.40

Transmission Control Protocol, Src Port: 48446, Dst Port: 80, Seq: 0, Len: 0

Source Port: 48446

Destination Port: 80

[Stream Index: 0]

[Stream Packet Number: 1]

[Conversation completeness: Incomplete, DATA (15)]

[TCP Segment Len: 0]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 2021415842

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 0

Acknowledgment number (raw): 0

1010 = Header Length: 40 bytes (10)

Flags: 0x002 [SYN]

0000 = Reserved: Not set

...0 = Accurate ECH: Not set

....0 = Congestion Window Reduced: Not set

....0 = ECH-Echo: Not set

...0 = Urgent: Not set

....0 = Acknowledgment: Not set

....0 = Push: Not set

...0 = Reset: Not set

.....1 = Syn: Set

[Expert Info (Chat/Sequence): Connection establish request (SYN): server port 80]

....0 = Fin: Not set

[TCP Flags:S.]

Window: 47368

0000 ca 73 47 c9 6f 62 12 8c c7 61 4a 40 00 00 45 00 sg ob ...aJ0 E

0010 00 3c 3a 34 40 00 40 06 4a 45 0a 00 00 0b ac 10 <:00 JE ...

0020 00 28 bd 3e 00 50 79 7c 5b a2 00 00 00 00 a0 82 (-> Px) [.....

0030 a5 64 b6 71 00 00 02 00 05 b4 04 02 00 0a 7e 9c d q [.....

0040 aa ba 00 00 00 00 01 03 03 09 [.....

- Qual è il numero di porta TCP di origine?

Il numero di porta TCP di origine per questo pacchetto è 48446, indicando il punto di partenza della connessione dal client verso il server.

- Come classificheresti la porta di origine?

Classificherei la porta TCP di origine come effimera, trattandosi di un numero elevato (48446) assegnato dinamicamente dal sistema operativo per avviare la connessione. Questo tipo di porta è tipico delle comunicazioni client verso server.

- **Qual è il numero di porta TCP di destinazione?**

Il numero di porta TCP di destinazione per questo pacchetto è 80, che corrisponde alla porta standard utilizzata per le connessioni HTTP verso server web. Questo conferma che il traffico è diretto al servizio web attivo sull'host H4.

- **Come classificherei la porta di destinazione?**

Classificherei la porta TCP di destinazione come "porta nota", in quanto corrisponde al numero 80, assegnato ufficialmente al protocollo HTTP. È una delle porte standard utilizzate per il traffico web, riconosciuta universalmente dai sistemi e dai servizi di rete.

- **Quale flag è impostato?**

Il flag TCP impostato in questo pacchetto è il SYN, che segnala l'inizio dell'handshake a tre vie. Questo flag indica che il client sta tentando di stabilire una connessione con il server.

- **A quale valore è impostato il numero di sequenza relativo?**

Il valore del numero di sequenza TCP in questo pacchetto è impostato su 0, come previsto nel primo segmento dell'handshake a tre vie. Questo valore iniziale viene scelto dal client per avviare la numerazione dei byte trasmessi nella connessione.

Dopo aver analizzato il **primo frame** dell'handshake TCP tra il PC e il server su H4, ho ripetuto la stessa procedura per i **due pacchetti successivi**, completando così l'esame dell'intero scambio a tre vie. Per ciascun frame, ho selezionato il pacchetto, espanso la sezione "Transmission Control Protocol" per verificare le porte di origine e destinazione, e ho ispezionato i flag impostati per confermare le fasi di **SYN, SYN-ACK e ACK**.

- **Quali sono i valori delle porte di origine e destinazione?**

In questo pacchetto, i valori delle porte TCP sono rispettivamente 80 per la porta di origine e 48446 per la porta di destinazione. Questo indica che la risposta proviene dal server web (porta nota 80) ed è diretta al client, che aveva aperto una porta effimera (48446) per ricevere i dati.

- **Quali flag sono impostati?**

In questo pacchetto, i flag TCP impostati sono SYN e ACK, indicando la seconda fase dell'handshake a tre vie. Il server sta riconoscendo la richiesta di connessione del client (SYN) e, allo stesso tempo, conferma la ricezione con un acknowledgment (ACK).

- In questo pacchetto, il valore del numero di sequenza è impostato su 0, mentre il numero di acknowledgment è 1. Questi valori confermano la seconda fase dell'handshake TCP: il server risponde al SYN del client con un segmento SYN-ACK, riconoscendo la richiesta e proponendo la propria sequenza.*

- **Quale flag è impostato?**

24	12.053409	10.0.0.11	172.16.0.40	TCP	74	48446 -> 80 [SYN] Seq=0 Min=42340 Len=0 MSS=1460 SACK_PERM TSval=2124196538 TScrcr=0 WS=512	
25	12.053450	172.16.0.40	10.0.0.11	TCP	74.80 -> 48446 [SYN, ACK] Seq=0 Ack=1 Win=43468 Len=0 MSS=1460 SACK_PERM TSval=3942472270 TScrcr=2124196538 WS=512		
26	12.053450	10.0.0.11	172.16.0.40	TCP	66 48446->80 [ACK] Seq=1 Ack=1 Min=42406 Len=0 TSval=2124196538 TScrcr=3942472270		
27	12.053567	10.0.0.11	172.16.0.40	HTTP	397 GET / HTTP/1.1		
28	12.053521	172.16.0.40	10.0.0.11	TCP	66 80 -> 48446 [ACK] Seq=1 Ack=332 Win=43520 Len=0 TSval=3942472270 TScrcr=2124196538		
29	12.053588	172.16.0.40	10.0.0.11	TCP	384 80 -> 48446 [PSH, ACK] Seq=1 Ack=332 Win=43520 Len=238 TSval=3942472270 TScrcr=2124196538 [TCP PDU reassembled in 3]		
30	12.053592	10.0.0.11	172.16.0.40	TCP	66 48446 -> 80 [ACK] Seq=332 Ack=239 Win=42496 Len=0 TSval=2124196538 TScrcr=3942472270		
31	12.053611	172.16.0.40	10.0.0.11	HTTP	681 HTTP/1.1 200 OK (text/html)		
32	12.053613	10.0.0.11	172.16.0.40	TCP	66 48446 -> 80 [ACK] Seq=332 Ack=854 Win=41984 Len=0 TSval=2124196538 TScrcr=3942472270		
45	12.231111	10.0.0.11	172.16.0.40	HTTP	411 GET /favicon.ico HTTP/1.1		
46	12.231202	172.16.0.40	10.0.0.11	HTTP	374 HTTP/1.1 404 Not Found (text/html)		

* Frame 26: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 Ethernet II, Src: 12:c:b7:c1:a4:a0 (12:8c:f7:61:a4:a0), Dst: ca:73:47:c9:6f:62 (ca:73:47:c9:6f:62)
 Internet Protocol Version 4, Src: 10.0.0.11, Dst: 172.16.0.40
 Transmission Control Protocol, Src Port: 48446, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
 Source Port: 48446
 Destination Port: 80
 Stream Index: 8
 [Stream Packet Number: 3]
 [Conversation completeness: Incomplete, DATA (15)]
 [TCP Segment Len: 0]
 Sequence Number: 1 (relative sequence number)
 Sequence Number (raw): 2021415843
 [Next Sequence Number: 1 (relative sequence number)]
 Acknowledgment Number: 1 (relative ack number)
 Acknowledgment number (raw): 2789479288
 1000 = Header Length: 32 bytes (8)
 * Flags: 0x010 [ACK]
 0000 = Reserved: Not set
0..... = Accurate ECN: Not set
0..... = Congestion Window Reduced: Not set
0..... = ECN-Echo: Not set
0..... = Urgent: Not set
0..... = Acknowledgment: Set
0..... = Push: Not set
0..... = Reset: Not set
0..... = SYN: Not set
0..... = FIN: Not set
 [TCP Flags: ...-A-...]
 Window: 83
 If (related window size == 478061)
 7 Transmission control protocol
 Packets: 50, Discarded: 17 (2.9%)
 Profile: Default

Visualizzare i pacchetti usando tcpdump

Per esplorare ulteriormente il contenuto del file capture.pcap e filtrare le informazioni di mio interesse, ho aperto una nuova finestra di terminale e digitato:

- man tcpdump

```
tcpdump(1)      General Commands Manual      tcpdump(1)

NAME
  tcpdump - dump traffic on a network

SYNOPSIS
  tcpdump [ -AbdDefhHjKlLnOpqStuvXx# ] [ -B buffer_size ]
          [ -c count ] [ --count ] [ -C file_size ]
          [ -E spi@ipaddr algo:secret.... ]
          [ -F file ] [ -G rotate_seconds ] [ -i interface ]
          [ --immediate-mode ] [ -j timestamp ] [ -M module ]
          [ -N secret ] [ --number ] [ --print ] [ -Q in/out/inout ]
          [ -r file ] [ -s snaplen ] [ -T type ] [ --version ]
          [ -v file ] [ -w file ] [ -W filecount ] [ -y datalinktype ]
          [ -z postrotate-command ] [ -Z user ]
          [ --time-stamp-precision=timestamp_precision ]
          [ --micro ] [ --nano ]
          [ expression ]

DESCRIPTION
  Tcpdump prints out a description of the contents of packets on a network
  interface that match the Boolean expression (see pcap-filter(7) for the
  expression syntax); the description is preceded by a time stamp,
  printed, by default, as hours, minutes, seconds, and fractions of a second
  since midnight. It can also be run with the -w flag, which causes
  it to save the packet data to a file for later analysis, and/or with the
  -r flag, which causes it to read from a saved packet file rather than to
  read packets from a network interface. It can also be run with the -v
  flag, which causes it to read a list of saved packet files. In all
  cases, only packets that match expression will be processed by tcpdump.

  Tcpdump will, if not run with the -c flag, continue capturing packets
  until it is interrupted by a SIGINT signal (generated, for example, by
  typing your interrupt character, typically control-C) or a SIGTERM signal
  (typically generated with the kill(1) command); if run with the -c
  flag, it will capture packets until it is interrupted by a SIGINT or
  SIGTERM signal or the specified number of packets have been processed.

  When tcpdump finishes capturing packets, it will report counts of:

Manual page tcpdump(1) line 1 (press h for help or q to quit)
```

Consultando le pagine manuali (man pages) del sistema Linux, ho navigato tra le opzioni disponibili per tcpdump, così da individuare i parametri utili per analizzare selettivamente il traffico catturato. In alcuni casi, ho premuto INVIO per superare l'intestazione e visualizzare il prompt di lettura.

- Cosa fa l'opzione -r?

Questo comando mi permette di visualizzare i pacchetti contenuti nel file capture.pcap, applicare filtri, e ispezionare il traffico registrato come se fosse in diretta.

Nello stesso terminale, ho aperto il file di cattura per visualizzare i primi tre pacchetti TCP registrati. Ho utilizzato il seguente comando:

- tcpdump -r /home/analyst/capture.pcap tcp -c 3

```
[analyst@secOps ~]$ tcpdump -r /home/analyst/capture.pcap tcp -c 3
reading from file /home/analyst/capture.pcap, link-type EN10MB (Ethernet), snapshot length 262144
07:45:24.617168 IP 10.0.0.11.48446 > 172.16.0.40.http: Flags [S], seq 2021415842, win 42340, options [mss 1460,sackOK,TS val 2124196538 ecr 0,nop,wscale 9], length 0
07:45:24.617209 IP 172.16.0.40.http > 10.0.0.11.48446: Flags [S.], seq 2701478287, ack 2021415843, win 43440, options [mss 1460,sackOK,TS val 3942472270 ecr 2124196538,nop,wscale 9], length 0
07:45:24.617218 IP 10.0.0.11.48446 > 172.16.0.40.http: Flags [.], ack 1, win 83, options [nop,nop,TS val 2124196538 ecr 3942472270], length 0
[analyst@secOps ~]$
```

1. Ci sono centinaia di filtri disponibili in Wireshark. Una rete di grandi dimensioni potrebbe avere numerosi filtri e molti tipi diversi di traffico. Elenca tre filtri che potrebbero essere utili a un amministratore di rete.

- **ip.addr == 192.168.1.1** Questo filtro mostra tutti i pacchetti in cui l'indirizzo IP specificato è coinvolto, sia come sorgente che come destinazione. Utile per monitorare il traffico da/verso un host specifico, rilevare anomalie o verificare la comunicazione tra dispositivi.
- **tcp.port == 443** Filtra tutto il traffico TCP diretto alla porta 443, tipicamente usata per HTTPS. Perfetto per analizzare connessioni sicure, verificare handshake TLS o identificare problemi con servizi web cifrati.
- **dns** Mostra solo i pacchetti relativi al protocollo DNS. Fondamentale per diagnosticare problemi di risoluzione dei nomi, identificare query sospette o analizzare il comportamento di applicazioni.

2. In quali altri modi Wireshark potrebbe essere utilizzato in una rete di produzione?

- **Diagnosi di problemi di rete:** Analizzo latenze, ritrasmissioni, pacchetti persi o fuori ordine per identificare colli di bottiglia, errori di configurazione o problemi hardware.
- **Verifica della sicurezza:** Monitoro traffico sospetto, tentativi di scansione, connessioni non autorizzate o protocolli non cifrati. Wireshark mi aiuta a individuare vulnerabilità e comportamenti anomali.
- **Ottimizzazione delle performance:** Studio il flusso dei pacchetti per capire come vengono utilizzate le risorse di rete, quali servizi generano più traffico e dove intervenire per migliorare l'efficienza.
- **Analisi dei protocolli applicativi:** Decodifico protocolli come HTTP, DNS, FTP, SMB, per capire come si comportano le applicazioni e verificare che rispettino gli standard.
- **Audit e conformità:** Registro e analizzo il traffico per dimostrare la conformità a normative come GDPR, ISO 27001 o PCI-DSS, documentando accessi e flussi sensibili.
- **Test e debug di nuove implementazioni:** Durante il rollout di nuovi servizi, uso Wireshark per verificare handshake, negoziazioni TLS, autenticazioni e corretto instradamento del traffico.