

## **REPORT TECNICO**

### **Analisi del Three-Way-Handshake con Wireshark**

**Redatto da:** Nicolò Calì Cybersecurity Student

**Data:** 17/02/2026

**Oggetto:** Osservazione del traffico di rete e analisi del protocollo TCP

## 1. Introduzione

Il presente report documenta l'attività di analisi del traffico di rete volta all'osservazione del processo di connessione TCP, noto come "**Three-Way Handshake**". L'attività è stata svolta utilizzando una workstation **CyberOps** virtualizzata e l'ambiente di emulazione di rete **Mininet**.

L'obiettivo principale dell'esercitazione è catturare ed esaminare i pacchetti generati tra un **host** client e un **server** web durante l'instaurazione di una sessione **HTTP**. Per l'analisi dei pacchetti e dei frame Ethernet, sono stati impiegati strumenti di sniffing quali **tcpdump** e **Wireshark**.

Nello specifico, l'analisi si concentra sui flag TCP (**SYN**, **SYN-ACK**, **ACK**) e sui numeri di sequenza utilizzati per garantire l'affidabilità della comunicazione.

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

CyberOPS Topology:

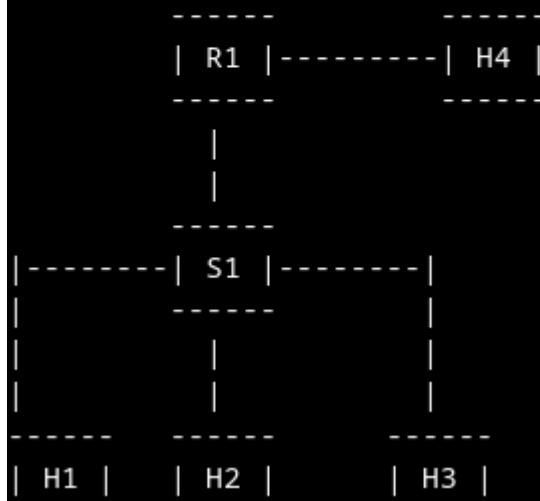


Fig. 1 Topologia di rete

## 2. Configurazione dell'ambiente di test e cattura del traffico

Per simulare uno scenario reale di comunicazione client-server, è stato utilizzato l'emulatore di rete **Mininet**.

La topologia configurata comprende diversi host, switch e router, replicando una piccola rete aziendale connessa a internet.

Dalla console di Mininet, sono state avviate due sessioni terminale per i nodi di interesse:

- **Host H1** (Client): indirizzo IP 10.0.0.11
- **Host H4** (Server Web): indirizzo IP 172.16.0.40

Sul nodo H4 è stato eseguito uno script bash ('**reg\_server\_start.sh**') per attivare un servizio web in ascolto sulla porta 80, simulando la funzione di un server **HTTP** pubblico. Questa configurazione è preliminare alla generazione e cattura del traffico TCP.

The screenshot shows a terminal window with three panes. The left pane displays the command-line interface for setting up a network topology. The middle pane, labeled 'CLIENT', shows the network diagram and the configuration process. The right pane, labeled 'SERVER', shows the command-line interface for initializing a web service on node H4.

**Left Pane (Terminal - analyst@secOps:~)**

```
[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> xterm H1
mininet> xterm H4
mininet> 
```

**Middle Pane (Node: H1)**

**Right Pane (Node: H4)**

```
[root@secOps analyst]# ./home/analyst/lab.support.files/scripts/reg_server_start.sh
[root@secOps analyst]# netstat -tulpn | grep 80
tcp        0      0 0.0.0.0:80          0.0.0.0:*
              LISTEN      896/n
ginx: master P
[root@secOps analyst]# 
```

Fig. 2 Avvio della topologia Mininet e inizializzazione del servizio web sul nodo Server H4.

Successivamente, sul nodo **client H1**, è stato avviato il browser web Mozilla Firefox con i privilegi dell'utente 'analyst'. Per intercettare il traffico di rete generato dal browser, è stato utilizzato lo strumento a riga di comando '**tcpdump**'.

Il comando eseguito:

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

Questo comando è servito a:

- Ascoltare sull'interfaccia di rete virtuale **H1-eth0** che è il Client.
- Limitare la cattura a 50 pacchetti (**-c 50**), sufficienti per registrare la fase di connessione.
- Salvare l'output in formato **.pcap** per una successiva analisi approfondita.

Durante l'esecuzione dello sniffer, è stata effettuata una richiesta HTTP all'indirizzo del server **172.16.0.40** tramite il browser, innescando così il processo di **handshake TCP**.

The screenshot shows a terminal window titled "Node: H1". The terminal content is as follows:

```
[root@secOps analyst]# su analyst
[analyst@secOps ~]$ firefox &
[1] 1057
[analyst@secOps ~]$ [Parent 1057, Main Thread] WARNING: Failed to create DBus proxy for org.a11y.Bus: Error spawning command line "dbus-launch --autolaunch=dc95ff0065de4db88cc6c0b568ea557f --binary-syntax --close-stderr": Child process exited with code 1 : 'glib warning', file /usr/src/debug/firefox/firefox-139.0.4/toolkit/xre/nsSigHandle.rs.cpp:201

** (firefox:1057): WARNING **: 08:47:56.781: Failed to create DBus proxy for org.a11y.Bus: Error spawning command line "dbus-launch --autolaunch=dc95ff0065de4db88cc6c0b568ea557f --binary-syntax --close-stderr": Child process exited with code 1

sudo tcpdump -i H1-eth0 -v -c 50 -w /home/analyst/capture.pcap
[sudo] password for analyst:
tcpdump: listening on H1-eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
50 packets captured
50 packets received by filter
0 packets dropped by kernel
[analyst@secOps ~]$
```

Fig. 3 Esecuzione di **tcpdump** su H1 e generazione di traffico HTTP verso il server.

### 3. Analisi del Traffico TCP

#### 3.1 Importazione e Filtraggio dei Dati

Il file di cattura (estensione **.pcap**), generato precedentemente tramite **tcpdump**, è stato importato nell'interfaccia grafica di **Wireshark** per un'analisi dettagliata.

Considerata la presenza di traffico di background generato dal sistema operativo e dal browser (*richieste DNS, protocolli di discovery etc..*), è stato necessario applicare un **filtro** di visualizzazione specifico.

Utilizzando il filtro '**ip.addr == 172.16.0.40**', è stato possibile isolare esclusivamente i pacchetti scambiati tra l'host client (10.0.0.11) e il server web target, escludendo tutto il "rumore" di rete non pertinente.

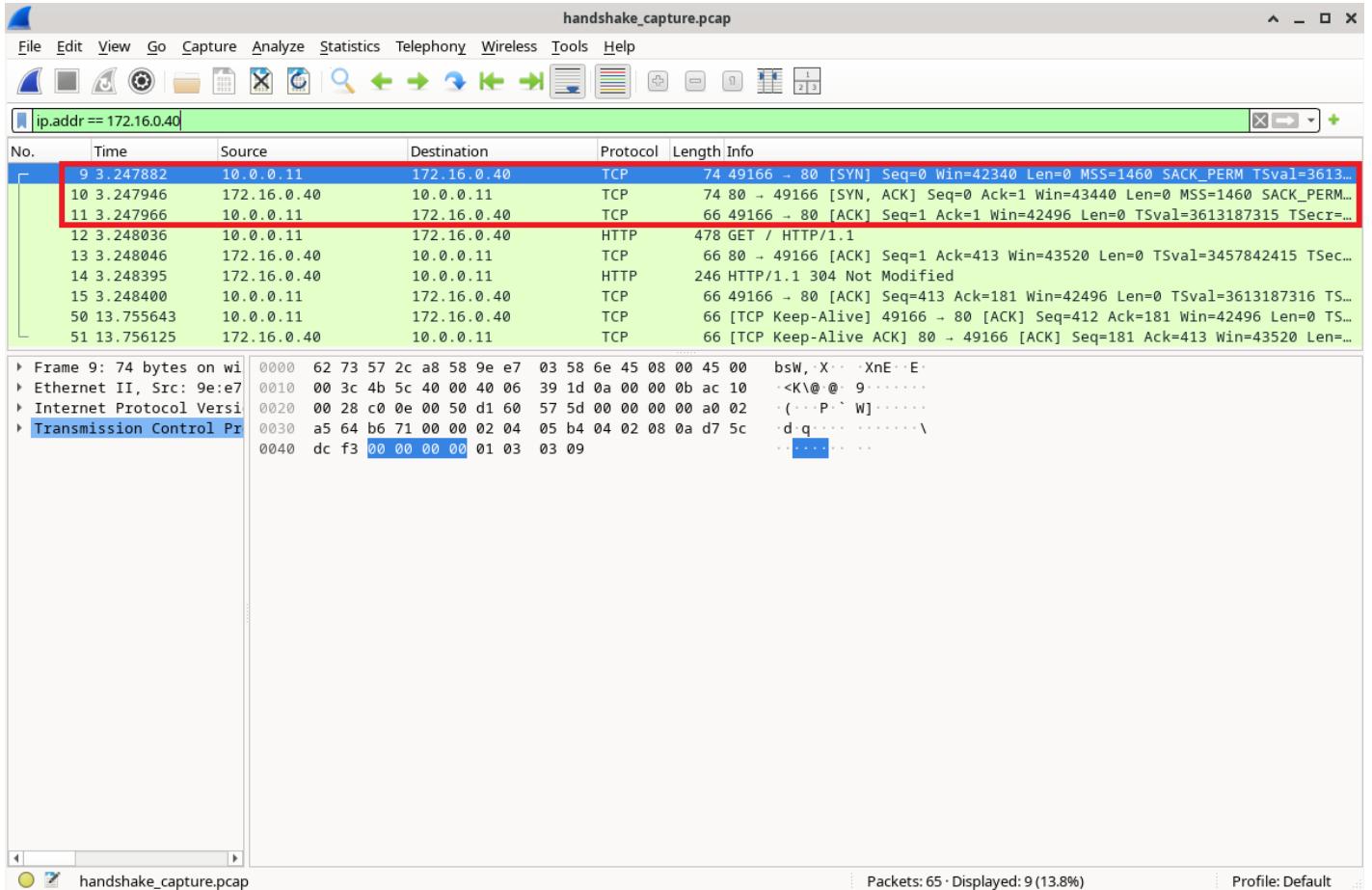


Fig. 4 Visualizzazione della sequenza di Three-Way Handshake TCP in Wireshark.

### 3.2 Osservazione del Three-Way Handshake

L'applicazione del filtro ha evidenziato con chiarezza la sequenza di instaurazione della connessione TCP. Come documentato in *Figura 4*, sono stati identificati i tre segmenti fondamentali del protocollo:

- **Pacchetto SYN (Client -> Server):** Il client 10.0.0.11 avvia la connessione inviando un segmento con il flag SYN impostato verso la porta 80 del server.
- **Pacchetto SYN-ACK (Server -> Client):** Il server 172.16.0.40 risponde confermando la richiesta (ACK) e inviando il proprio numero di sequenza (SYN).
- **Pacchetto ACK (Client -> Server):** Il client risponde con un ultimo ACK, completando il processo e stabilendo la sessione (Stato **ESTABLISHED**).

A seguito di questa procedura, è stato possibile osservare l'inizio dello scambio dati HTTP.

### 3.3 Dettagli dei Parametri TCP

Dall'analisi approfondita dei singoli frame Ethernet e dei segmenti TCP, sono stati estratti i seguenti parametri:

#### A) Primo Segmento (SYN):

- **Porta di Origine** → 49166. Si tratta di una porta effimera allocata temporaneamente dal sistema operativo del client.
- **Porta di Destinazione** → 80. Il protocollo HTTP.
- **Flags** → È stato rilevato il flag SYN attivo, indicando la richiesta di sincronizzazione dei numeri di sequenza.

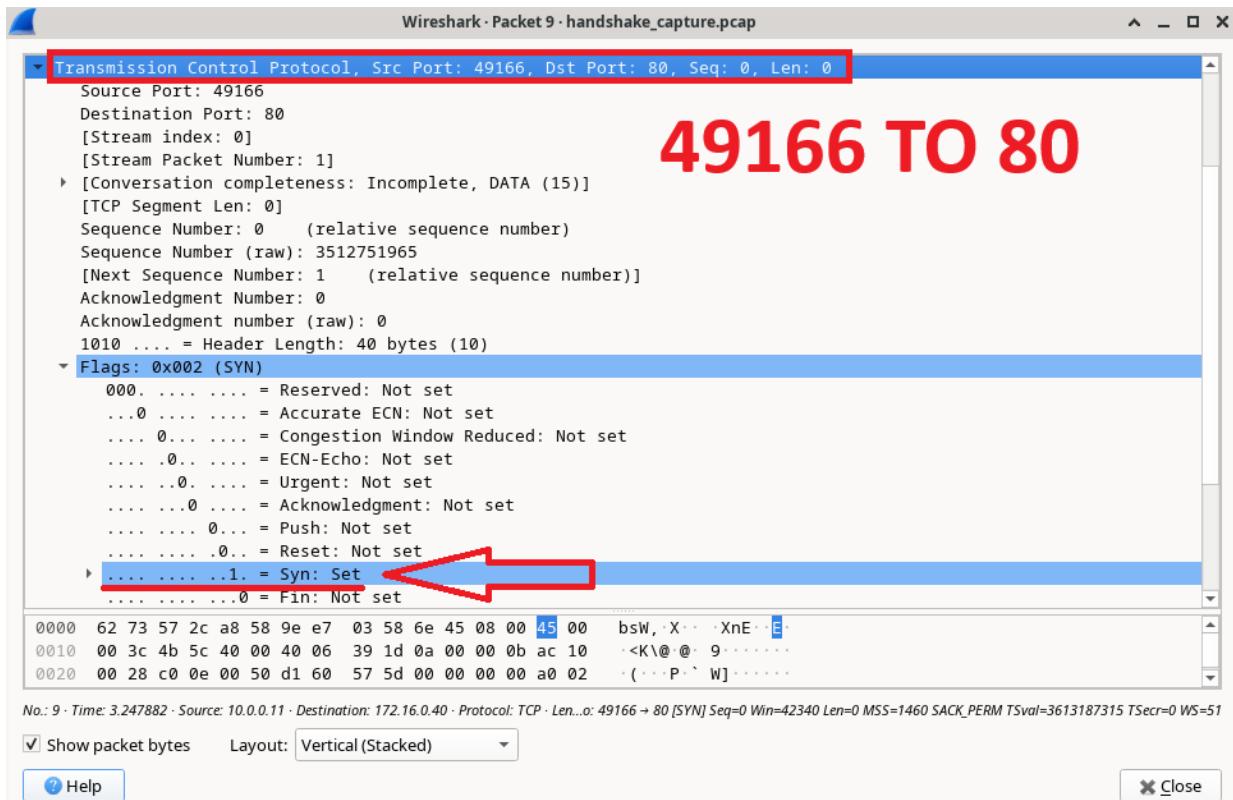


Fig. 5 Analisi del pacchetto SYN.

#### B) Secondo Segmento (SYN-ACK):

- **Porte** → I ruoli si invertono; la porta di origine è la 80 (**Server**) e quella di destinazione è la 49166 (**Client**).
- **Flags** → Sono attivi sia il flag **SYN** che il flag **ACK**, a conferma che il server ha ricevuto la richiesta ed è pronto a instaurare la sessione.

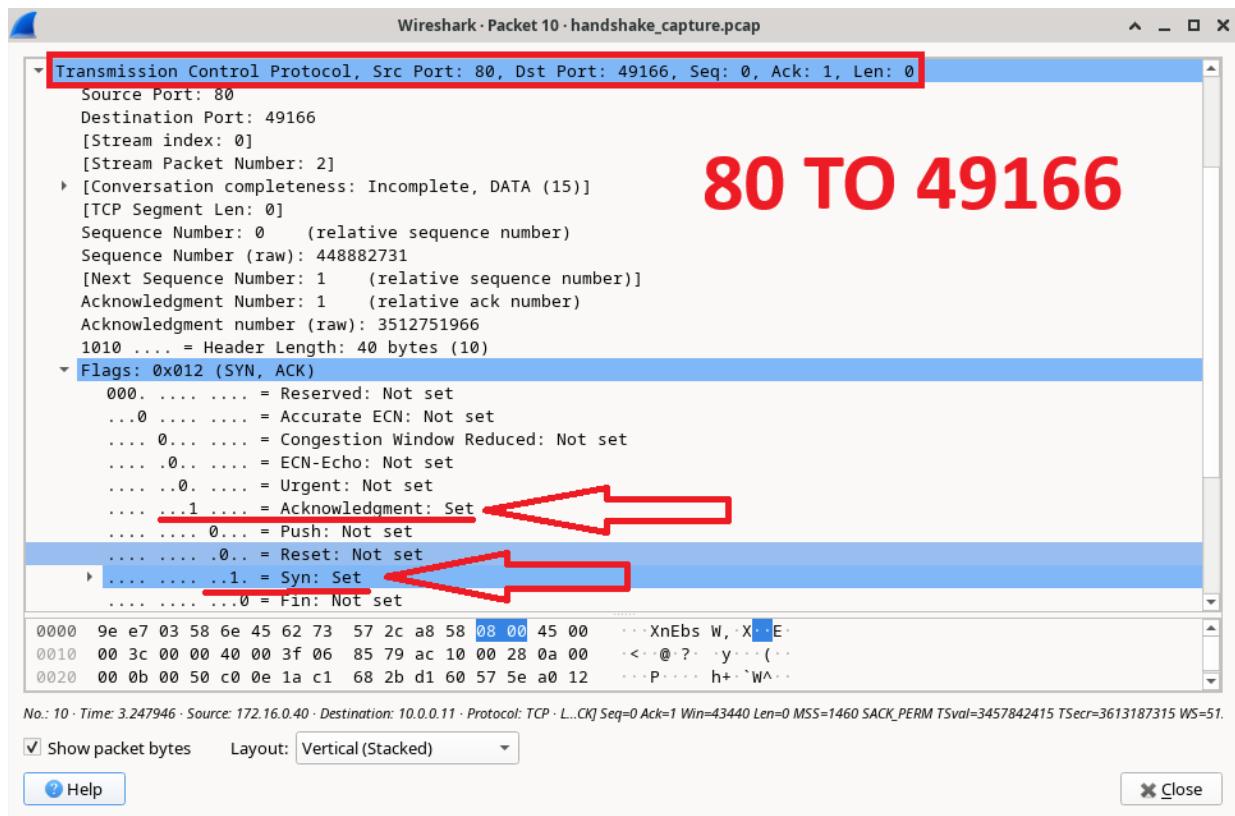


Fig. 6 Analisi del pacchetto SYN-ACK.

### C) Terzo Segmento (ACK):

- **Direzione** → Dal Client al Server.
- **Flags** → È attivo esclusivamente il flag ACK.
- **Numeri di Sequenza** → I valori relativi di **Sequence Number** e **Acknowledgment Number** sono entrambi impostati a **1**.

Questo pacchetto conclude l'handshake, la connessione TCP passa allo stato **ESTABLISHED**, permettendo l'inizio del trasferimento dati al livello HTTP.

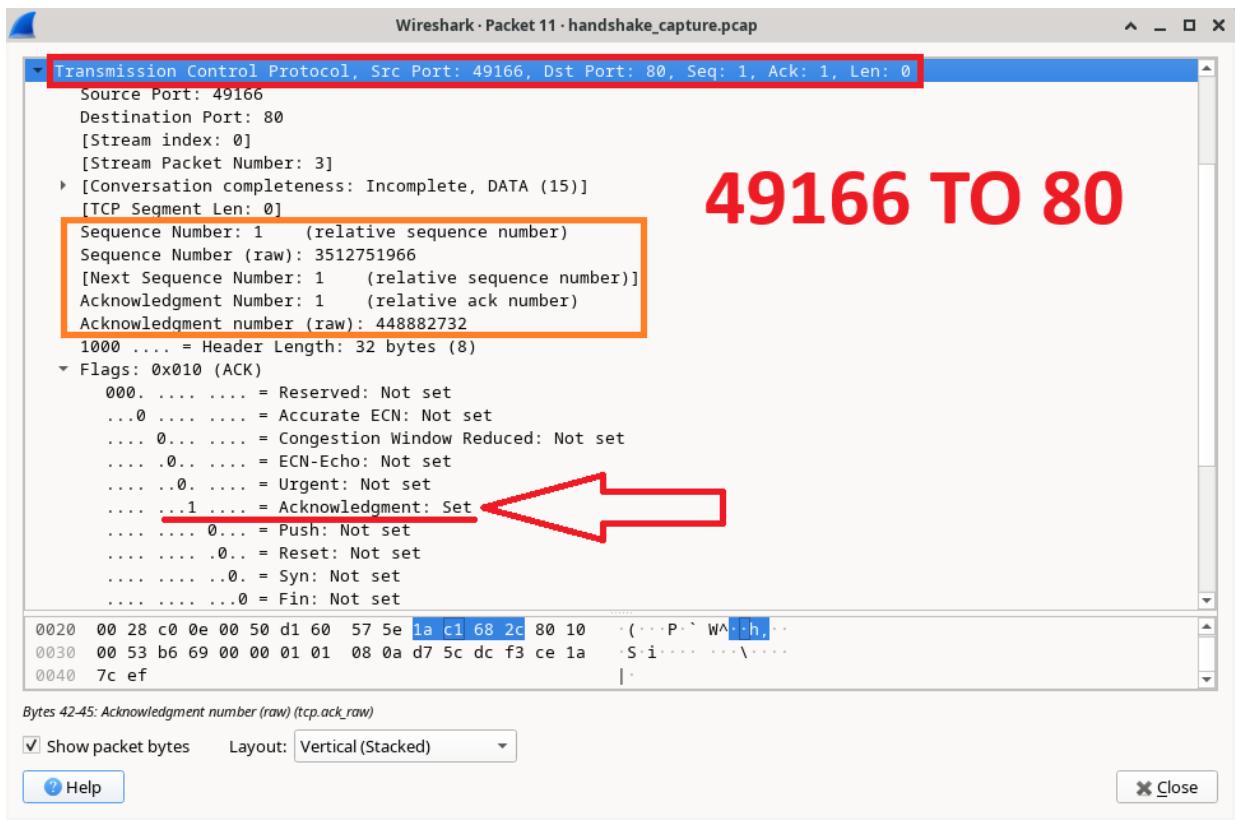


Fig. 7 Analisi del pacchetto ACK.

## 4. Conclusioni

L'attività di laboratorio ha permesso di osservare e analizzare con successo il meccanismo di instaurazione di una connessione TCP (**Three-Way Handshake**) in un ambiente di rete controllato.

L'utilizzo combinato di **Mininet** per la simulazione della topologia, **tcpdump** per la cattura del traffico e **Wireshark** per l'analisi approfondita, ha fornito una visione chiara di come i protocolli di trasporto garantiscano l'affidabilità della comunicazione.

Nello specifico, è stato verificato come i flag SYN e ACK, insieme ai numeri di sequenza, coordinino l'apertura della sessione tra client e server prima dello scambio effettivo dei dati HTTP.