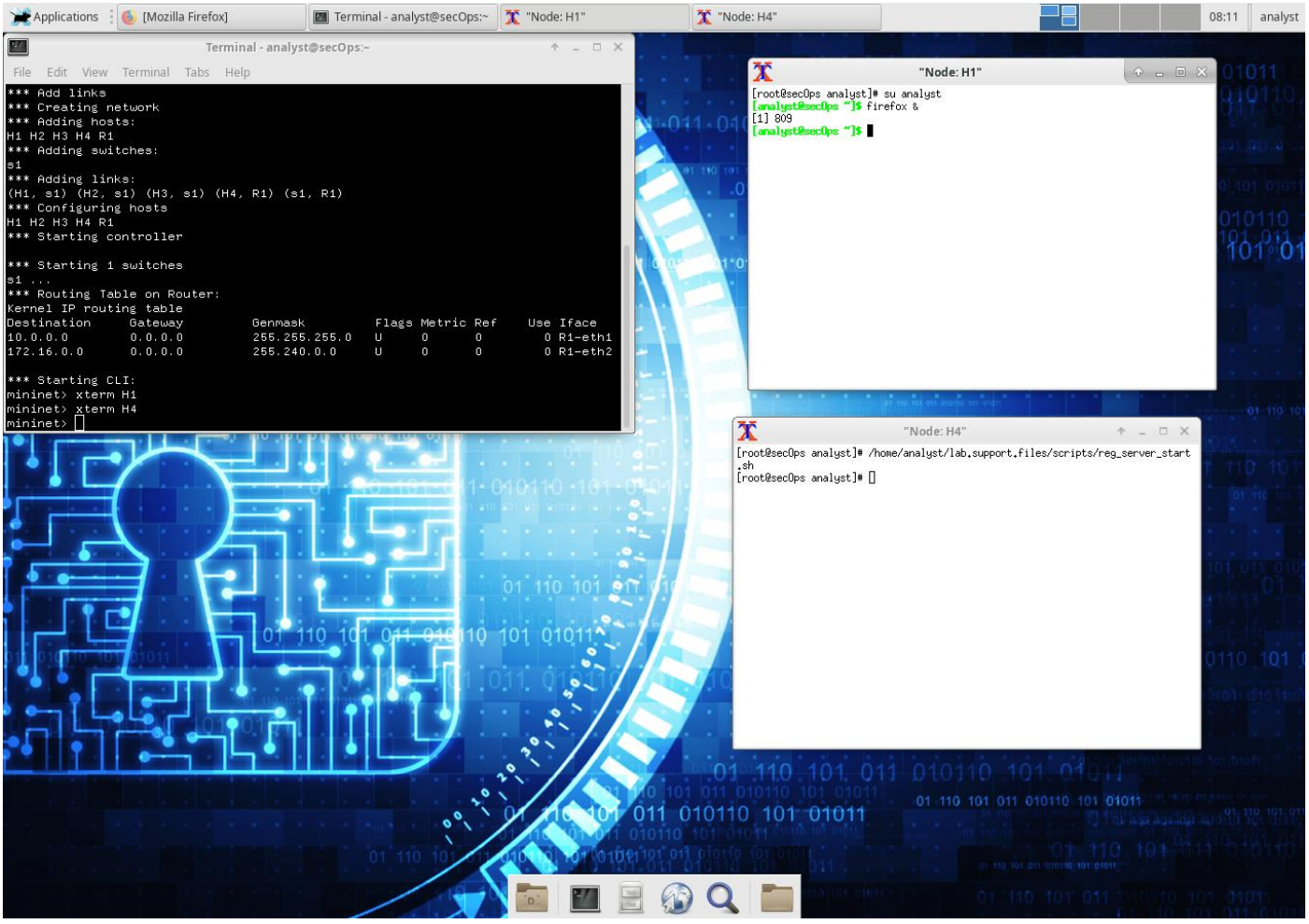
Utilizzo di Wireshark per Osservare la Stretta di Mano TCP a 3 Vie

Introduzione

Il protocollo TCP è uno dei pilastri su cui si fonda la comunicazione affidabile su Internet. A differenza di altri protocolli che inviano dati sperando che arrivino, TCP si assicura che mittente e destinatario siano pronti e sincronizzati prima di iniziare lo scambio vero e proprio. Questo processo iniziale di "messa d'accordo" è noto come "stretta di mano a tre vie" (3-Way Handshake). Lo scopo di questo laboratorio è stato quello di "spiare" questo dialogo iniziale tra due macchine, catturando il traffico di rete nel momento esatto in cui avviene una connessione e analizzandolo poi con strumenti specifici per vederne i dettagli.

Preparare gli host per catturare il traffico

Per poter osservare questo processo fondamentale in un ambiente controllato, senza le interferenze del traffico di rete esterno, è stato utilizzato il simulatore di rete Mininet all'interno della macchina virtuale CyberOps. Questo strumento ci ha permesso di creare una piccola rete virtuale composta da host (computer simulati) e switch. In particolare, abbiamo attivato due host: H1, destinato a fungere da client (il computer che inizia la connessione), e H4, configurato come server web grazie a uno script apposito (reg_server_start.sh) che lo ha messo in ascolto per richieste in arrivo sull'indirizzo IP 172.16.0.40.



Il passo successivo è stato generare l'evento che volevamo osservare: l'inizio di una connessione TCP. Per farlo, dall'host H1, abbiamo avviato il browser web Firefox. È stato necessario prima passare dall'utente root all'utente analyst (su analyst), poiché per motivi di sicurezza non è consigliato eseguire applicazioni come un browser con privilegi elevati.

Prima, però, di effettuare la connessione vera e propria, abbiamo avviato sull'host H1 lo strumento tcpdump. Questo programma agisce come una sorta di "registratore" del traffico di rete, ascoltando tutto ciò che passa sull'interfaccia di rete specificata (H1-eth0, l'interfaccia virtuale di H1) e salvandolo in un file (capture.pcap). Abbiamo impostato tcpdump per catturare solo i primi 50 pacchetti (-c 50) per mantenere la cattura focalizzata sull'inizio della comunicazione.

Con tcpdump attivo e in ascolto, abbiamo rapidamente inserito l'indirizzo del server web (172.16.0.40) nella barra degli indirizzi di Firefox e premuto Invio. Questa azione ha scatenato la richiesta di connessione da parte di Firefox (il client) verso il server web su H4. tcpdump ha fedelmente registrato i pacchetti scambiati in quei primi istanti.

Analizzare i pacchetti utilizzando Wireshark

Terminata la breve cattura, abbiamo utilizzato un altro strumento fondamentale, Wireshark, per analizzare il contenuto del file capture.pcap. Wireshark è come un microscopio per il traffico di rete, capace di decodificare i pacchetti e mostrarne il contenuto in modo organizzato e leggibile.

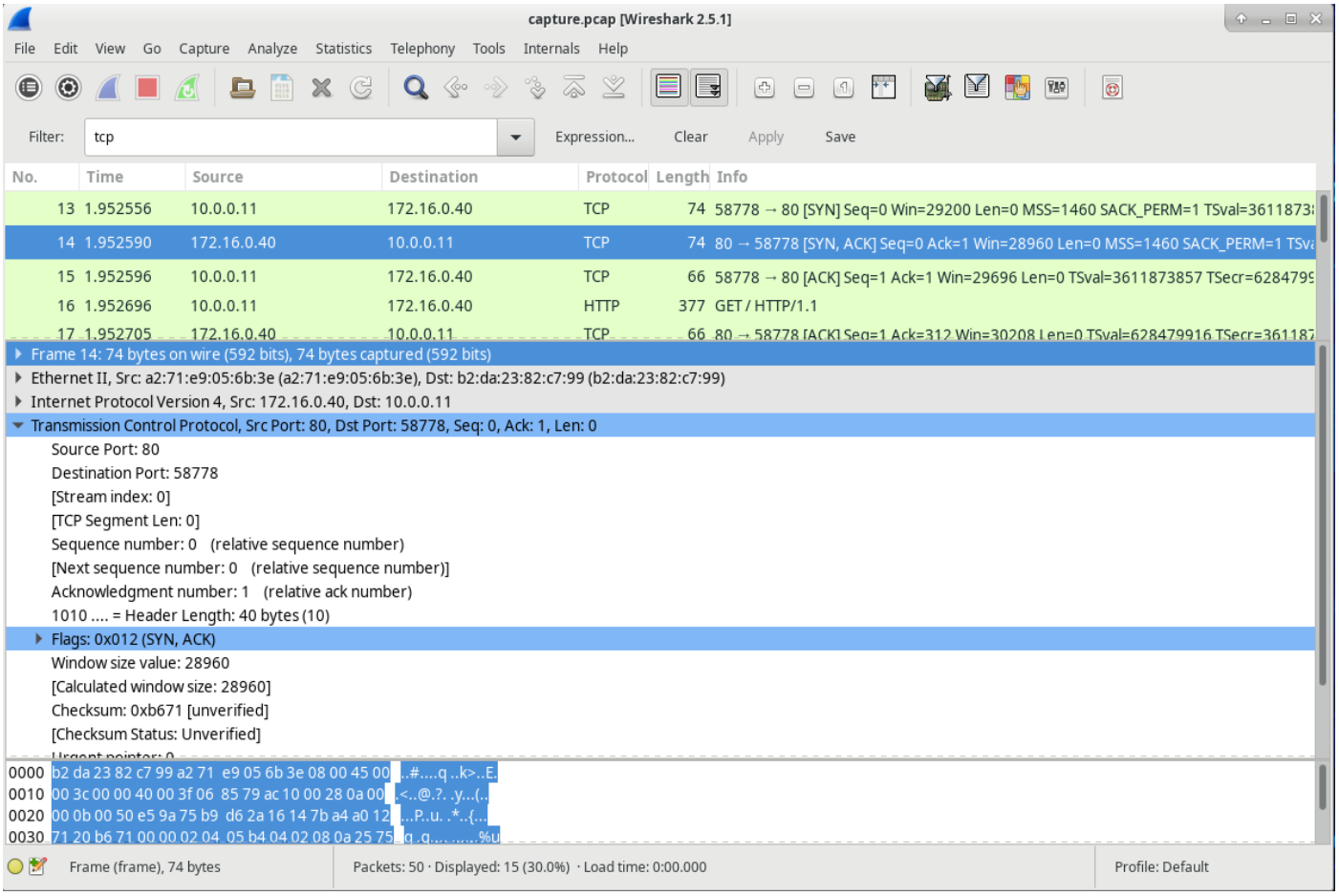
Aprendo il file e applicando un filtro per visualizzare solo il traffico TCP (tcp), l'attenzione si è concentrata sui primissimi pacchetti scambiati tra H1 e H4. Qui è stato possibile osservare chiaramente la sequenza della stretta di mano a tre vie:

Il Primo Contatto (SYN): Il primo pacchetto visibile da H1 verso H4 era un pacchetto speciale, identificato dal flag SYN (Synchronize). È come se il client H1 avesse bussato alla porta del server H4, dicendo "Ciao, vorrei iniziare una conversazione, il mio numero di partenza è [un numero casuale X]".

La Risposta e Conferma (SYN-ACK): Il server H4, ricevuta la richiesta, ha risposto quasi immediatamente. Il suo pacchetto conteneva due informazioni cruciali, segnalate dai flag SYN e ACK (Acknowledgment). Era come se dicesse: "Ho sentito la tua richiesta (confermo il tuo numero X+1) e accetto di parlare. Il mio numero di partenza è [un numero casuale Y]".

L'Accordo Finale (ACK): Infine, il client H1 ha inviato un ultimo pacchetto, contrassegnato solo dal flag ACK. Questo pacchetto confermava la ricezione della risposta del server, dicendo essenzialmente: "Ricevuto il tuo numero di partenza (confermo Y+1). Siamo d'accordo, possiamo iniziare a scambiare dati!".

Questo scambio di tre messaggi ha stabilito formalmente la connessione, sincronizzando i numeri di sequenza che entrambi gli host useranno per tenere traccia dei dati inviati e ricevuti, garantendo così l'affidabilità della comunicazione successiva (la richiesta della pagina web e la relativa risposta).



What is the TCP source port number?

58778

How would you classify the source port?

Dynamic

What is the TCP destination port number?

80

How would you classify the destination port?

HTTP

Which flag (or flags) is set?

SYN flag

What is the relative sequence number set to?

0

What are the values of the source and destination ports?

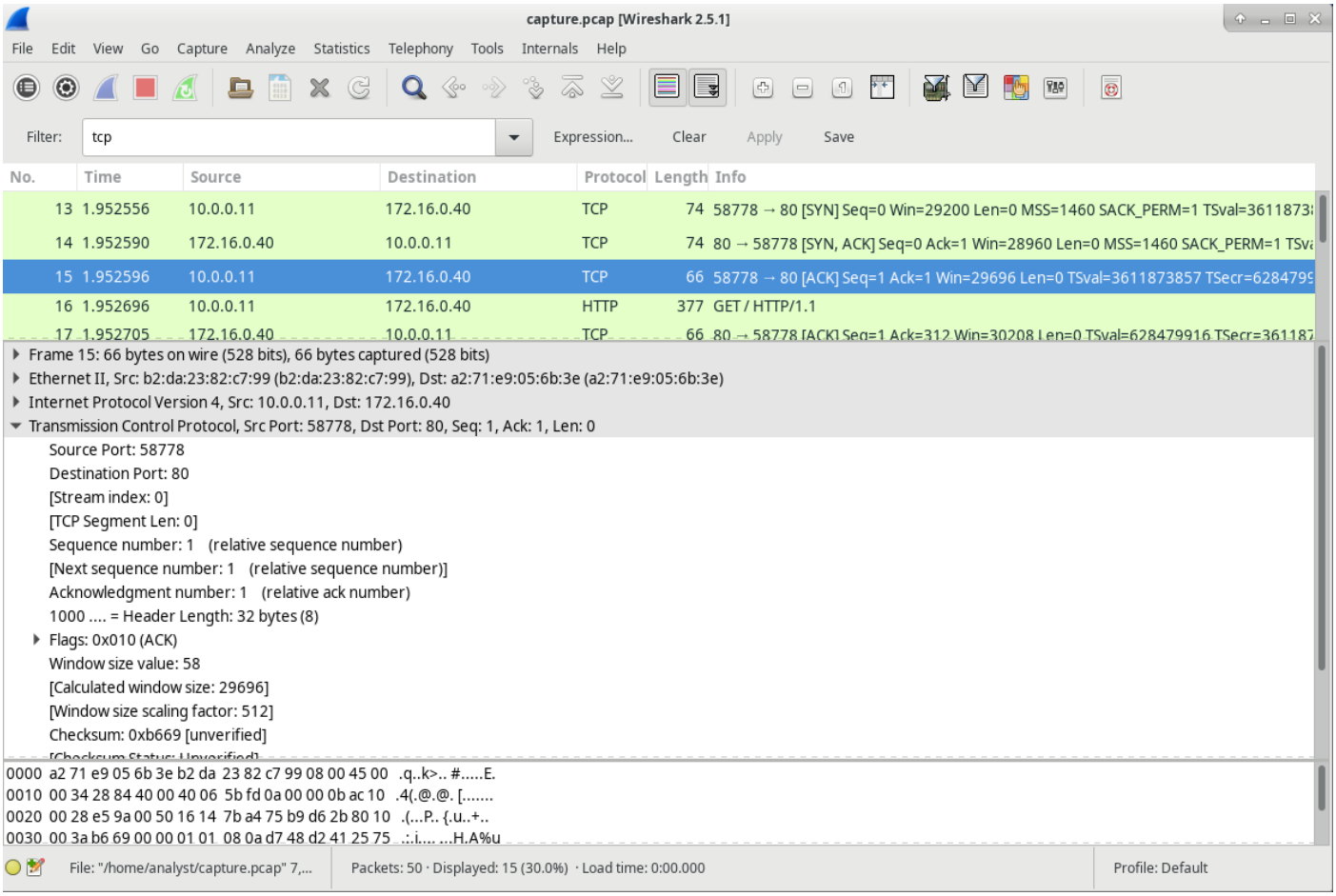
Source Port 80, Destination Port 58716

Which flags are set?

SYN, ACK

What are the relative sequence and acknowledgment numbers set to?

The relative sequence number is 0, and the relative acknowledgment number is 1.



Which flag (or flags) is set?

Acknowledgment flag (ACK)

Visualizzare i pacchetti con tcpdump

```
[analyst@secOps ~]$ tcpdump -r /home/analyst/capture.pcap tcp -c 3
reading from file /home/analyst/capture.pcap, link-type EN10MB (Ethernet)
08:13:08.122395 IP 10.0.0.11.58778 > 172.16.0.40.http: Flags [S], seq 370441123, win 29200, options
 [mss 1460,sackOK,TS val 3611873857 ecr 0,nop,wscale 9], length 0
08:13:08.122429 IP 172.16.0.40.http > 10.0.0.11.58778: Flags [S.], seq 1975113258, ack 370441124, w
in 28960, options [mss 1460,sackOK,TS val 628479916 ecr 3611873857,nop,wscale 9], length 0
08:13:08.122435 IP 10.0.0.11.58778 > 172.16.0.40.http: Flags [.], ack 1, win 58, options [nop,nop,T
S val 3611873857 ecr 628479916], length 0
```

1. There are hundreds of filters available in Wireshark. A large network could have numerous filters and many different types of traffic. List three filters that might be useful to a network administrator.

Answers will vary but could include TCP, specific IP Addresses (source and/or destination), and protocols such as HTTP.

2. What other ways could Wireshark be used in a production network?

Wireshark is often used for security purposes for after-the-fact analysis of normal traffic or after a network attack. New protocols or services may need to be captured to determine what port or ports are used.