https://itexamanswers.net/9-2-6-lab-using-wireshark-to-observe-the-tcp-3-way-handshake-answers.html

Come primo passaggio usiamo Mininet per emulare una rete virtuale sulla macchina virtuale CyberOps

Usando il comando:

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

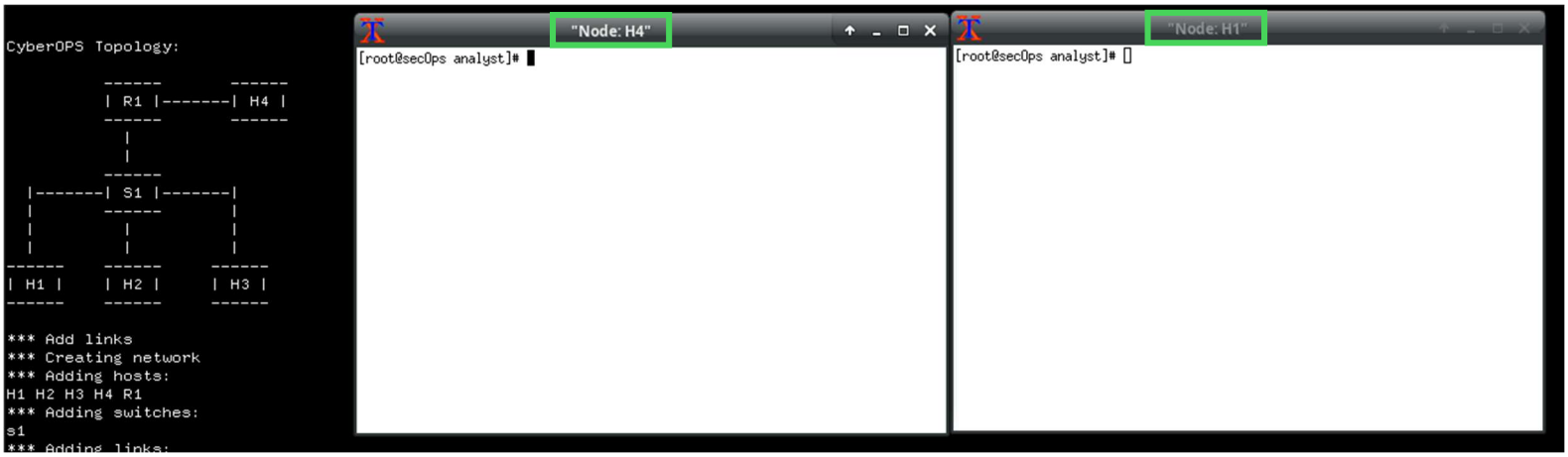
Che, come viene scritto nel codice, è un ambiente preimpostato da CyberOps che importa Mininet:

```
[analyst@secOps scripts]$ cat cyberops_topo.py
#!/usr/bin/python2

"""
cyberops_top.py: Sets up, configures and start Cisco CyberOps Course Topology.

The example topology creates a router and two IP subnets:

    1. 10.0.0.0/24   (R1-eth1, IP: 10.0.0.1)
    2. 172.16.0.0/24 (R1-eth2, IP: 172.16.0.1)

Subnet 1 is the internal network containing:
    - 1 Single switch
    - 3 Three internal hosts -  H1, H2 and H3

Subnet 2 represents the external network containing:
    - 1 External host, H4

The Topology implemented by this script looks like the diagram below:

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

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, NOX, OVSController
#from mininet.node import Controller
```

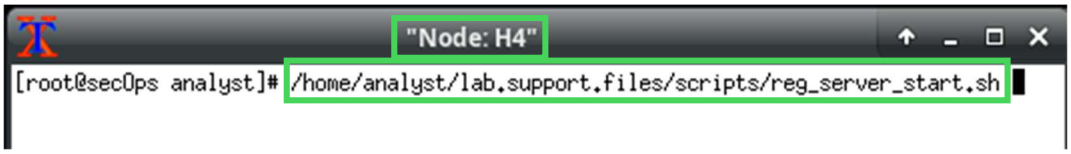
Viene aperta una rappresentazione grafica con tutti i dati che definiscono lo stato della rete virtuale:

```
CyberOPS Topology:

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

*** Add 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
*** 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>
```

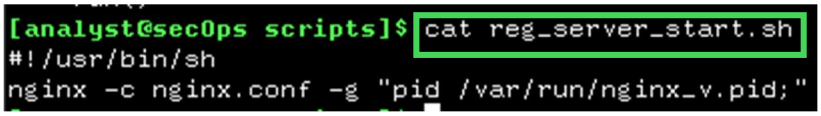
Per poter aprire un terminale su ogni dispositivo uso il comando:
xterm H1 (per aprire un terminale su host H1)
xterm H4 (per aprire un terminale su host H2)



Successivamente, avvio un web server su H4 tramite lo script preimpostato da CyberOps:

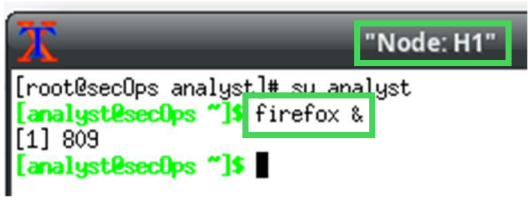



(codice scritto all'interno del file ↓)

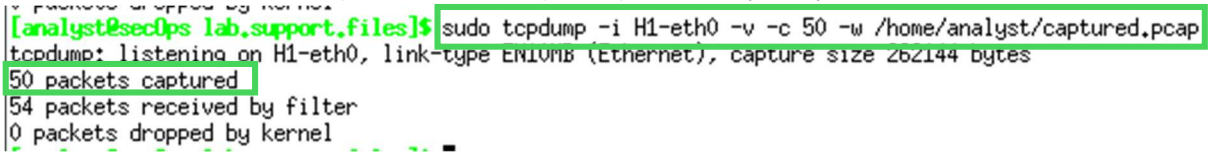



Ora cambio l'user di h1 da root ad Analyst dato che, come viene scritto nella guida, per motivi di sicurezza non è possibile eseguire Firefox dall'account utente root.


E successivamente avvio il browser su h1



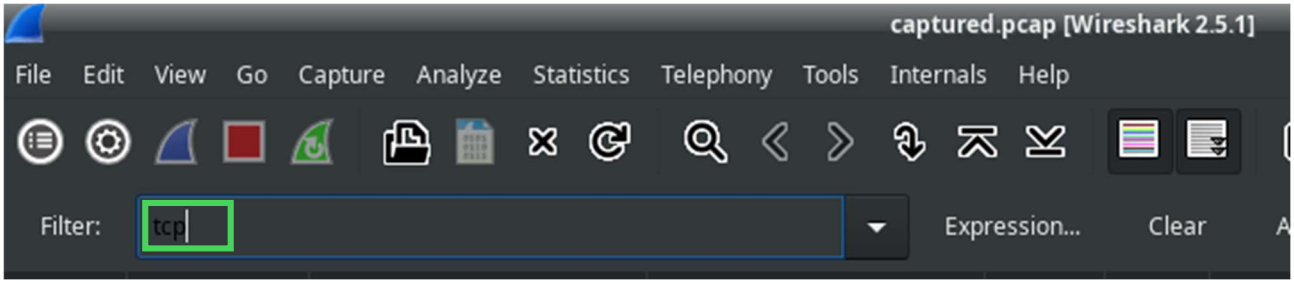

Adesso inizio la sessione di tcpdump dal terminale del host 1 mettendo il risultato all'interno di un file che chiamerò "capture.pcap" .
Usando lo switch -v posso vedere in tempo reale l'avanzamento del dump.
Questo comando interrompe la cattura dopo 50 pacchetti (-c 50).



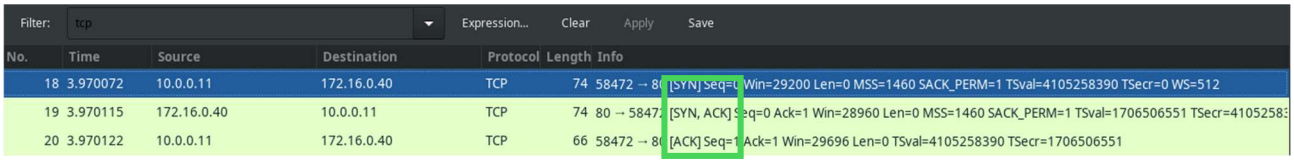

Visualizzo la corretta creazione del file .pcap:

La fase successiva è quella di analizzare il traffico catturato usando Wireshark che aprirò dell'host H1 con il comando:

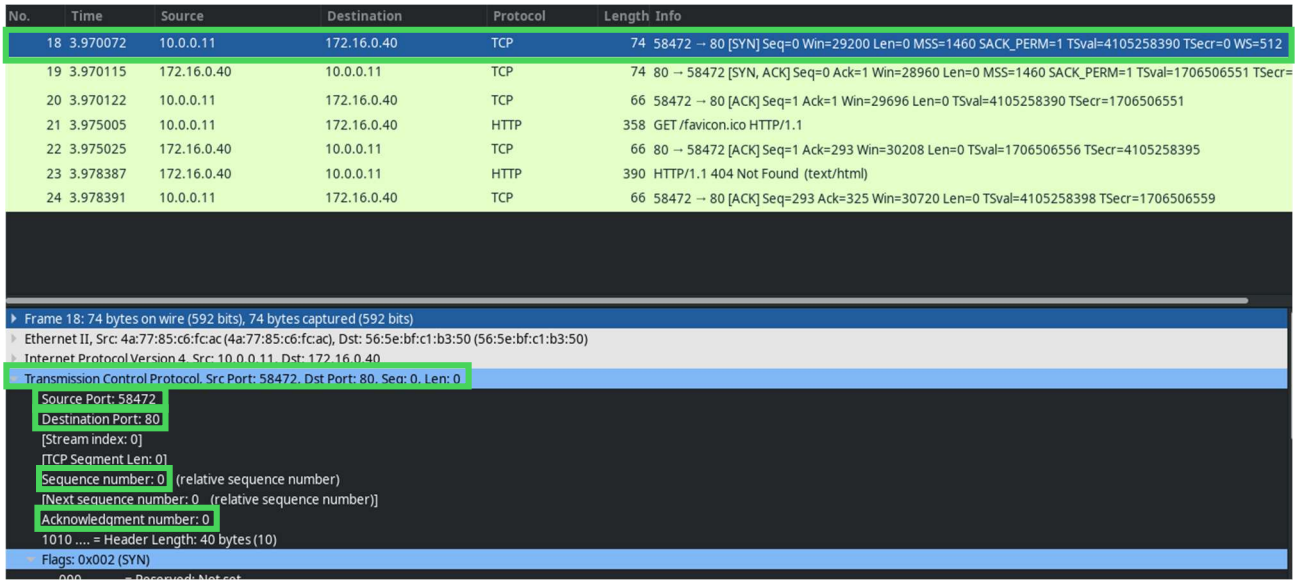

Applico il filtro per avere solo pacchetti di tipo tcp:




Nei primi tre pacchetti possiamo notare il three-way-handshake con i tipici pacchetti [SYN], [SYN,ACK] e [ACK]

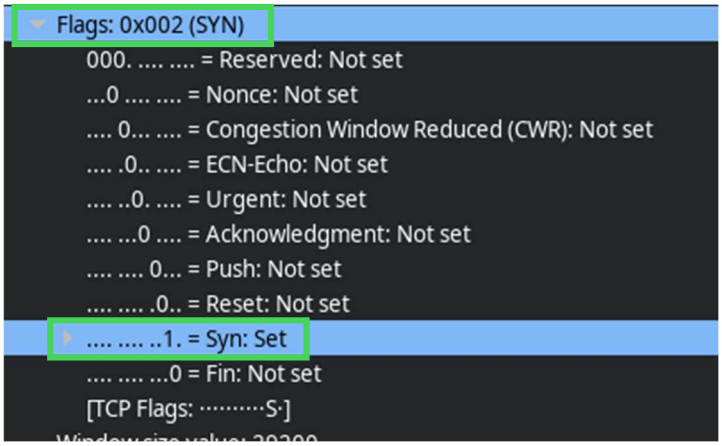



Analizziamo il primo pacchetto cliccando su Trasmission Control Protocol nella sezione inferiore:



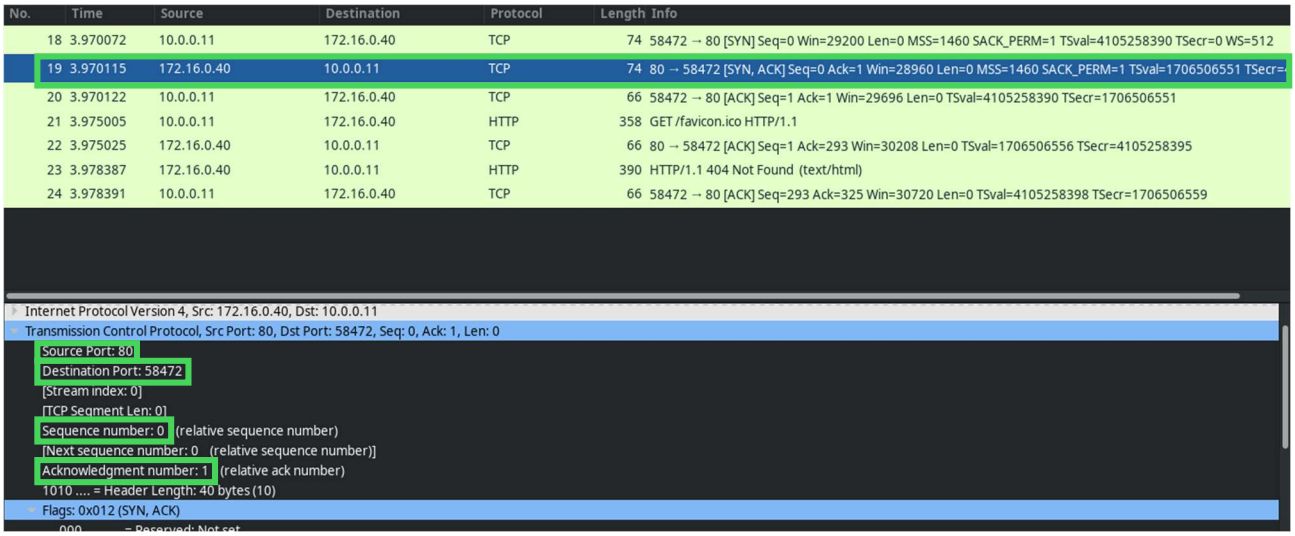

possiamo vedere altri dettagli, tra cui le porte di destinazione/sorgente e il sequence number che generalmente viene inizializzato casualmente per evitare spoofing ma Wireshark normalizza questo dato a 0 per comodità di lettura e il Acknowledgment number impostato anch'esso a 0 dato che non serve per il primo scambio.

se clicchiamo su Flag, possiamo vedere dei dettagli relativi ai bit dei flag impostati in quel pacchetto:

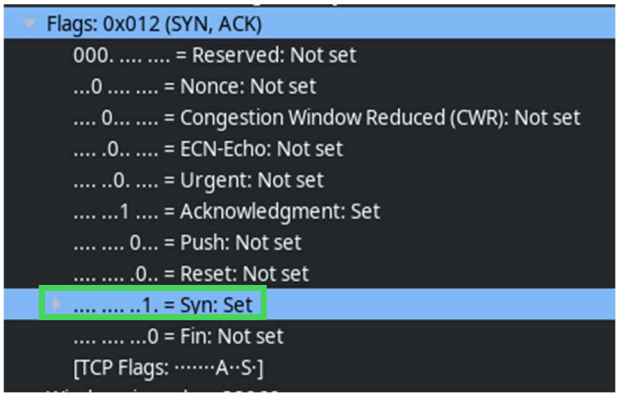



Possiamo infatti notare come l'unico bit settato a 1 è quello del SYN dato che stiamo analizzando il primo pacchetto del three-way-handshake.

Analizzando secondo pacchetto:

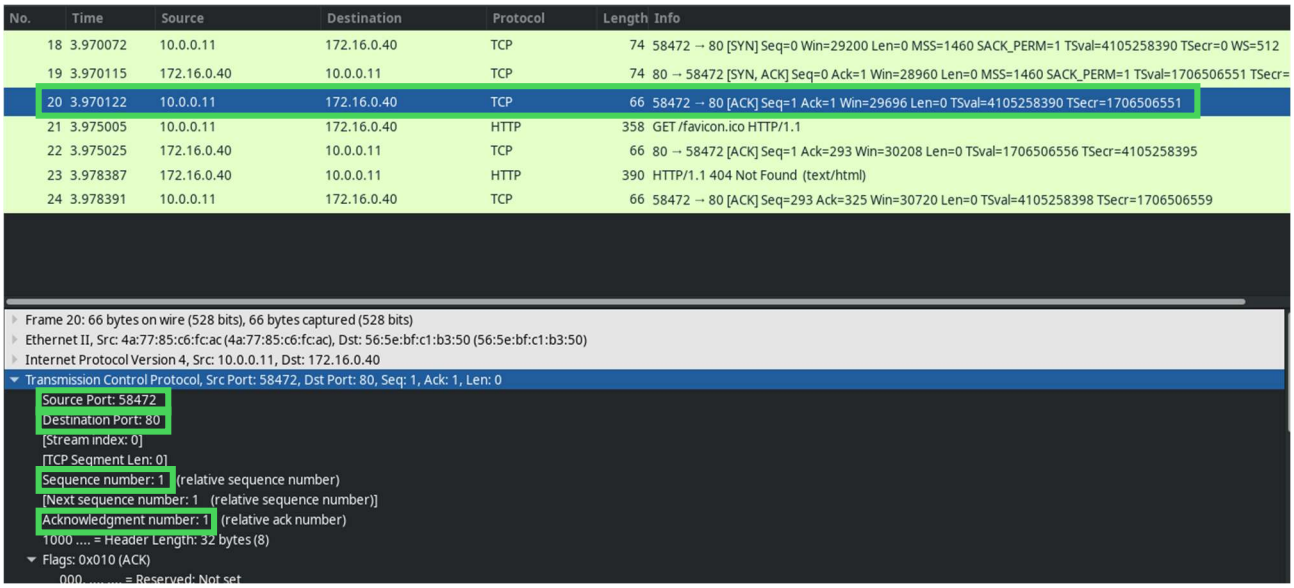



Possiamo vedere come le porte sorgente e destinazione si sono invertite rispetto al pacchetto precedente, dato che è una risposta dal server di destinazione al client sorgente.
il sequence è rimasto a 0 mentre il Acknowledgment number è diventato 1 (seq+1).




Il flag impostato è quello di Acknowledgment.

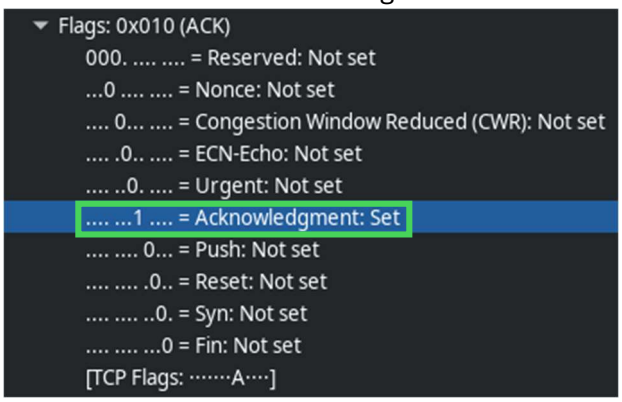
Analizzando il terzo pacchetto:




Possiamo vedere che le porte destinazione e sorgente si sono nuovamente invertite, tornando ad essere quelle del primo pacchetto dato che quest'ultimo pacchetto è una conferma per il destinatario dell'avvenuta connessione tra i due dispositivi dato che il mittente ha già avuto conferma con il precedente pacchetto [SYN/ACK].

Possiamo anche notare come il sequence number è diventato 1 e il Acknowledgment number 1 confermando la corretta connessione tra i due dispositivi.

Se andiamo a visualizzare i flag:




L'unico è quello di Acknowledgment