

Introduzione:

Il compito di oggi prevedeva di effettuare una **catturare il traffico**, in particolare il protocollo **TCP (3 way handshake)**, con i tool quali:

Wireshark: è uno strumento grafico per l'analisi del traffico di rete. Consente di catturare i dati che transitano su un'interfaccia di rete in tempo reale e di esaminarli in dettaglio. Offre una visualizzazione chiara e intuitiva dei pacchetti, permettendo di vedere informazioni come indirizzi IP, porte e flag utilizzati nei protocolli di comunicazione.

Tcpdump: invece, è uno strumento a riga di comando che serve allo stesso scopo, ma è più leggero e spesso usato in ambienti server o per analisi rapide. Anche Tcpdump cattura il traffico di rete, ma visualizza i dati **direttamente nel terminale** o li salva in un file per ulteriori analisi. Pur essendo meno intuitivo di Wireshark, è estremamente potente e supporta filtri complessi per concentrarsi su traffico specifico.

Come funziona il TCP?

Il **TCP (Transmission Control Protocol)** è un protocollo di comunicazione orientato alla connessione che garantisce un trasferimento affidabile dei dati tra due dispositivi. Prima di inviare dati, TCP stabilisce una connessione tramite un processo chiamato *stretta di mano a tre vie* (3-way handshake), in cui i dispositivi negoziano e sincronizzano i parametri di comunicazione.

UDP (User Datagram Protocol), al contrario, è un protocollo senza connessione e meno complesso. Non verifica se i pacchetti arrivano a destinazione né li ordina. Per questo motivo, è più veloce e leggero, ma meno affidabile rispetto a TCP.

Pratica:

Configurazione della Topologia di rete virtuale:

Ho eseguito un script in Python (cyberops_topo.py) con il comando:

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

Che ha configurato una **rete virtuale**, la topologia è composta da:

1 router centrale (R1);

1 interruttore (S1);

4 Host (H1, H2, H3, H4);

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

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
0.0.0.0          0.0.0.0         0.0.0.0         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> 
```

Successivamente ho aperto **2 terminali**, per **interagire** con i **nodì H1 e H4**, questo mi ha permesso di effettuare comandi su i dispositivi;

```
"Node: H1"
[root@sec0ps analyst]#

"Node: H4"
[root@sec0ps analyst]#

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

Successivamente nel **nodo H4**, avvio il **web server**, invece nel **nodo H1**, scalo i miei privilegi ad **analyst**, perché per motivi di sicurezza non potrei altrimenti avviare Firefox da **root**;

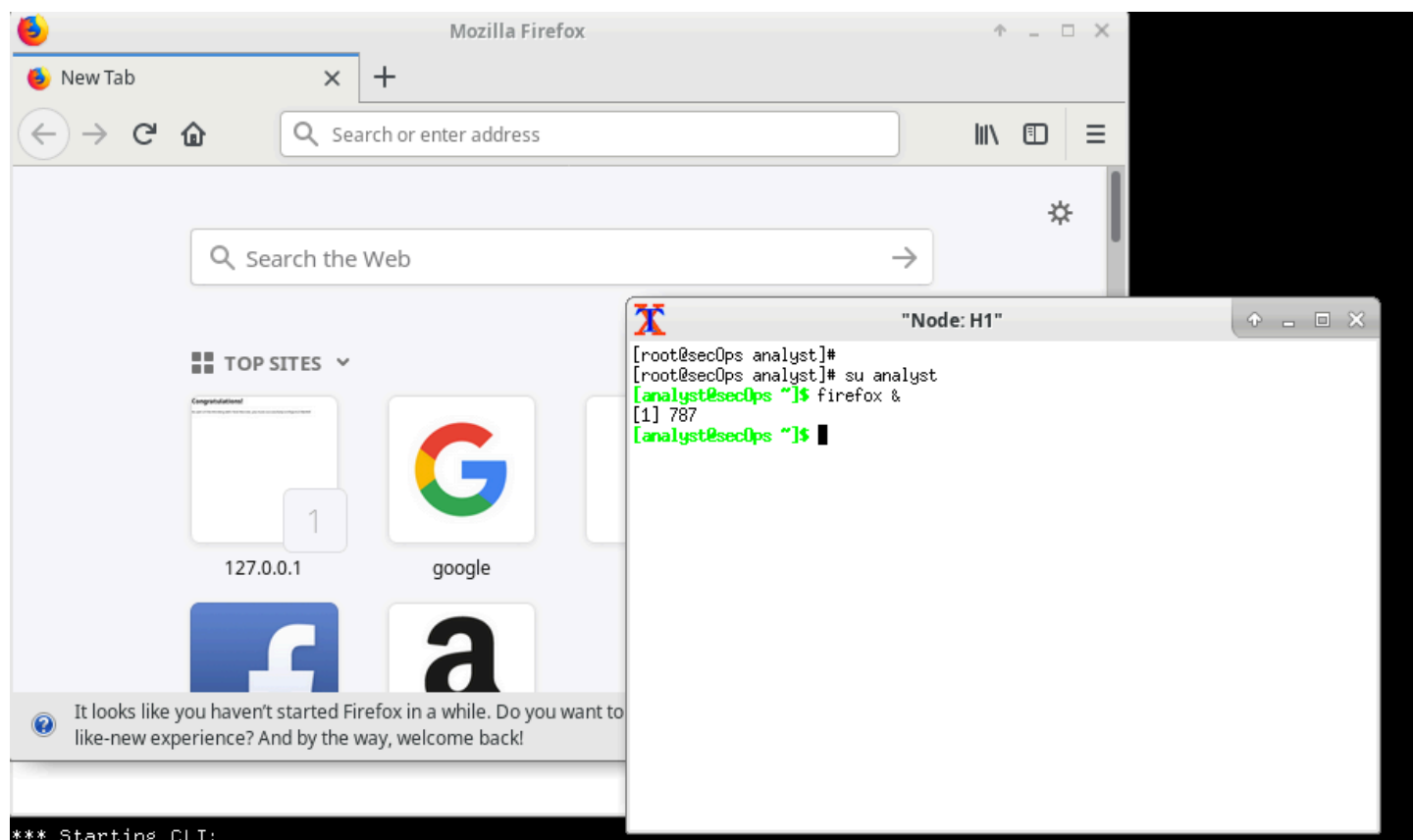
```
"Node: H4"
[root@sec0ps analyst]# /home/analyst/lab.support.files/scripts/reg_server_start
.sh
[root@sec0ps analyst]#

"Node: H1"
[root@sec0ps analyst]#
[root@sec0ps analyst]# su analyst
[analyst@sec0ps ~]$

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

Una volta scalato i privilegi, apro Firefox con il comando:

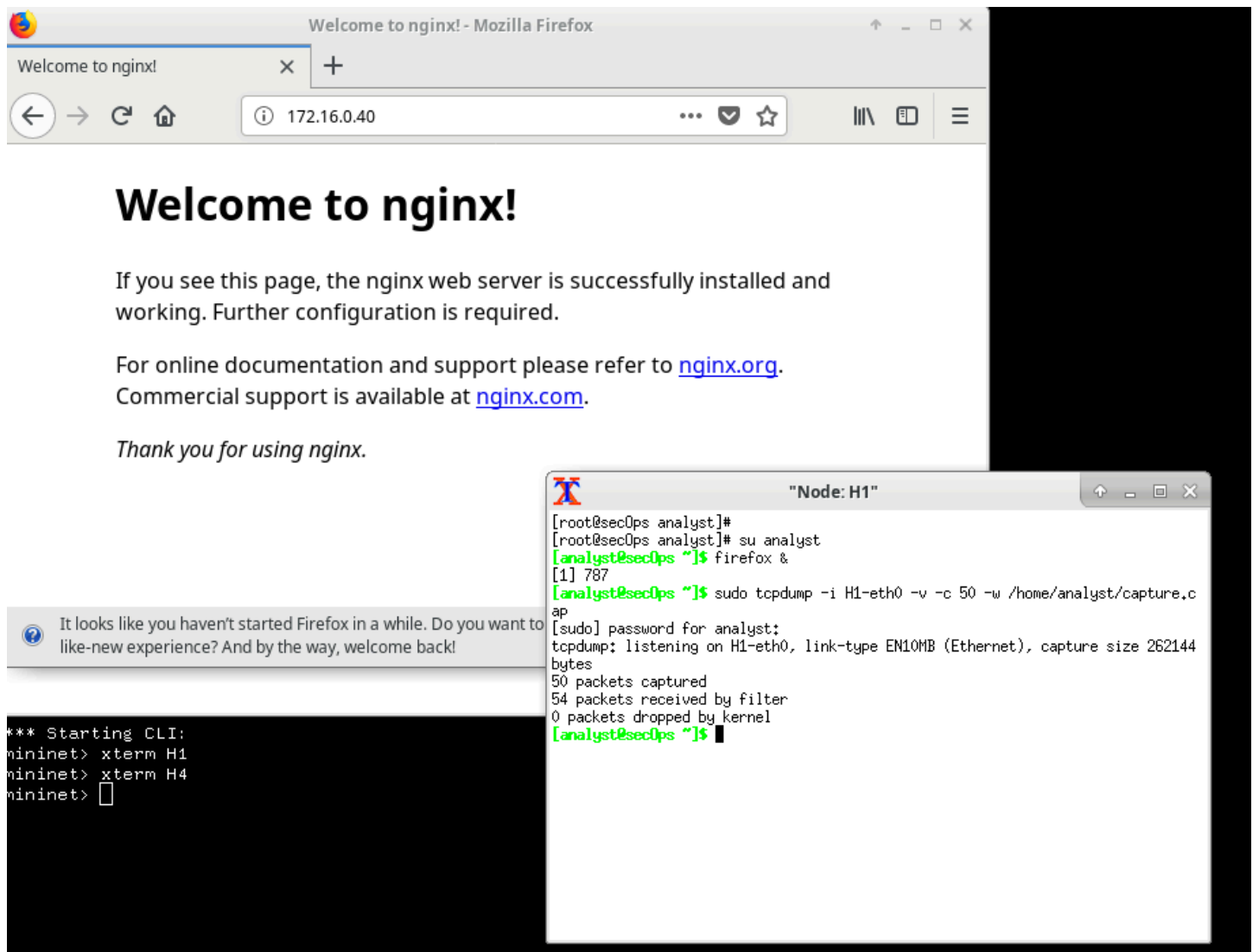
firefox &



Come passo finale, nel **nodo H1** mi metto in ascolto, per monitorare e catturare il traffico in ingresso con il comando:

tcpdump -i H1-eth0 -w /home/analyst/capture.cap

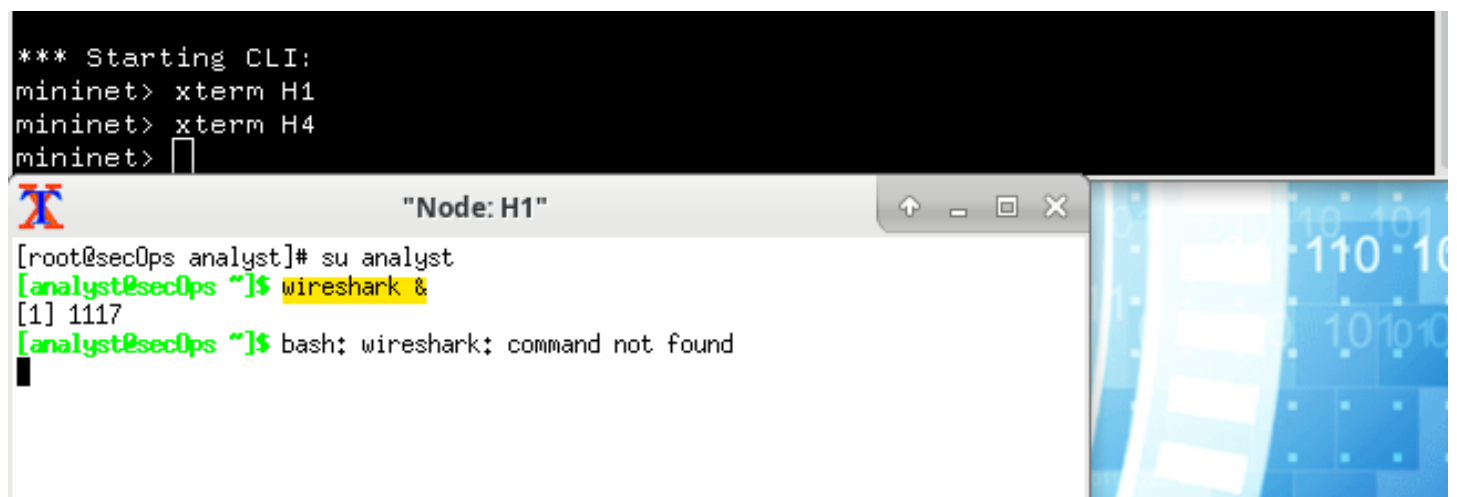
Infatti una volta fatto accesso al server, possiamo vedere come nel **nodo H1** verranno catturati 50 pacchetti di 54;



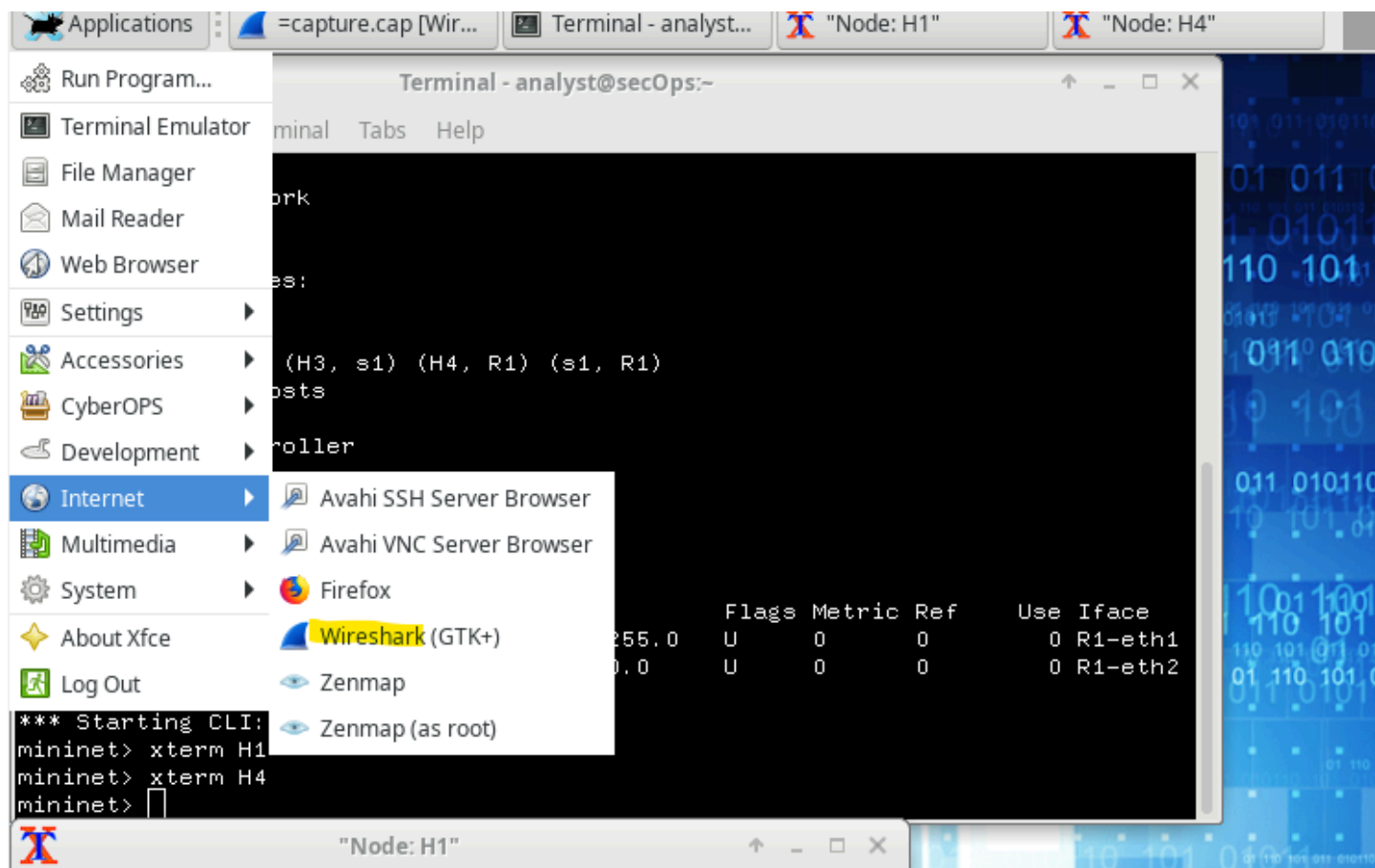
Cattura con WireShark:

Il procedimento iniziale è uguale a quello per tcpdump, ma ora nel nodo H1 dovremmo avviare invece che Firefox, **WireShark**;

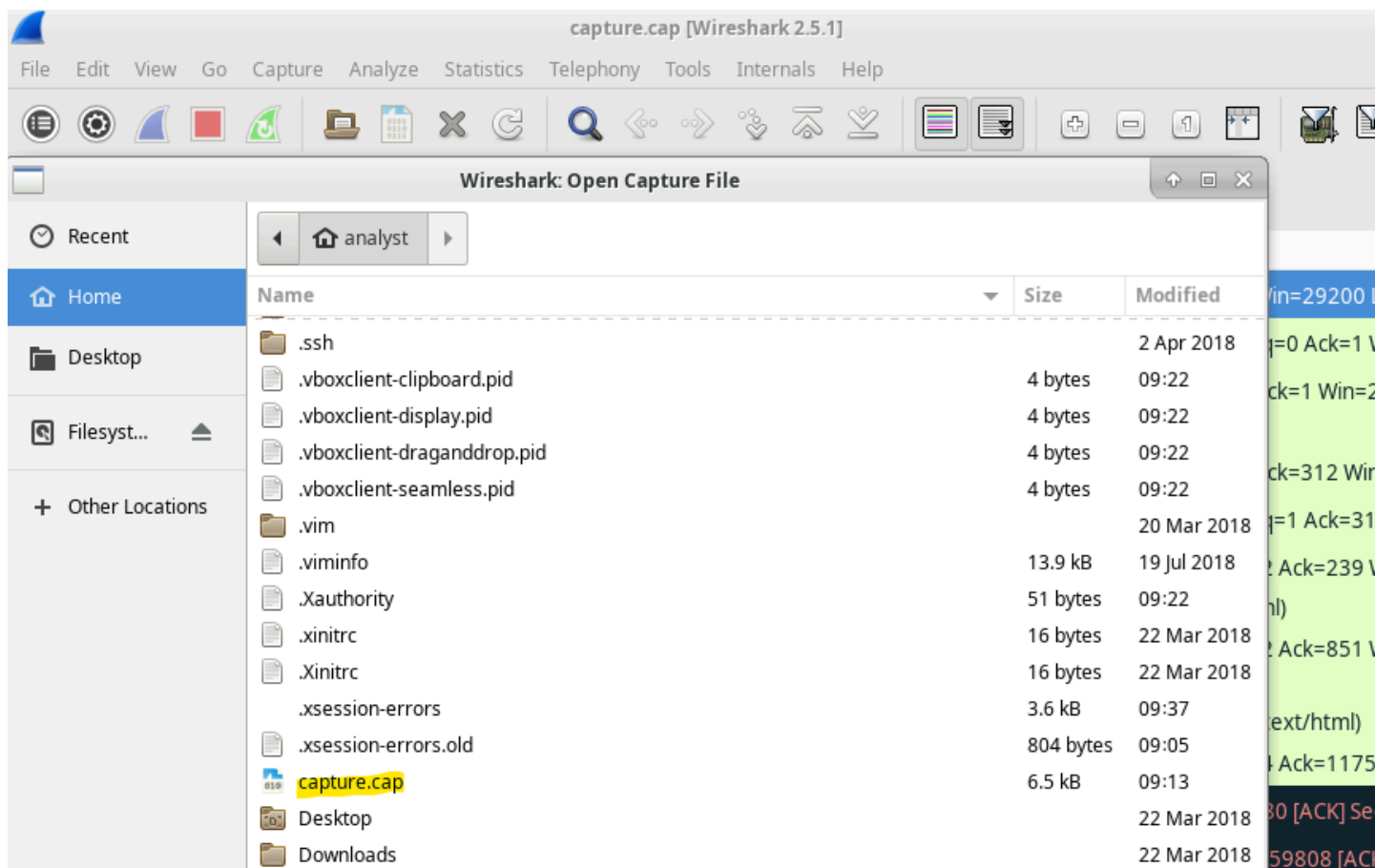
Se **non dovesse avviarsi** dal terminale del nodo H1;



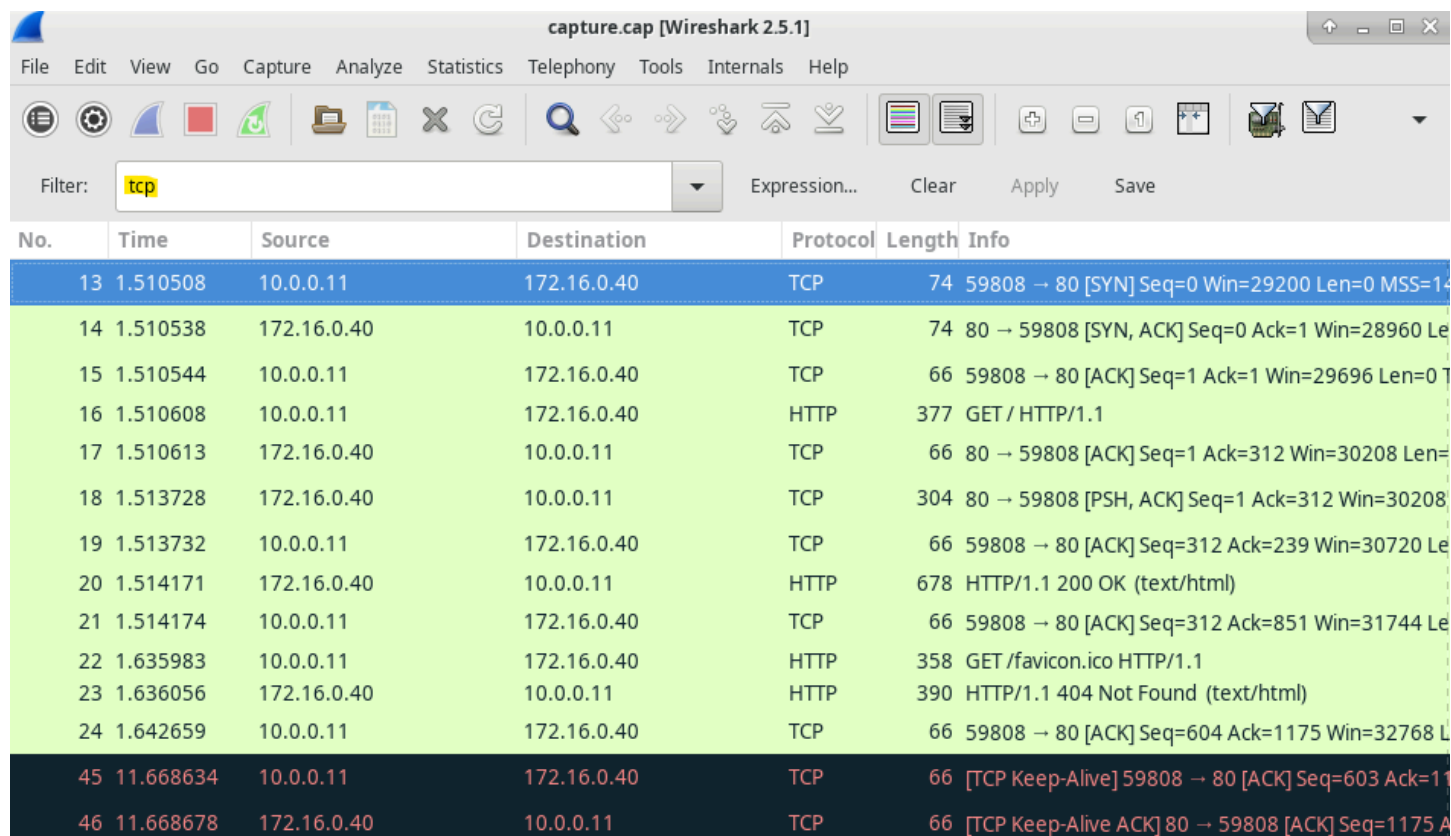
Possiamo avviarlo **manualmente**;



Una volta aperto andiamo su **file, open**, e nella directory **home** andiamo a prendere il file **capture.cap**;



Infine, dovremmo andare a filtrare i risultati, scrivendo nel campo **filter** “**tcp**”. E così potremmo **vedere** e **analizzare** solo il **TCP** da **Wireshark**;



The image shows the Wireshark 2.5.1 interface with a packet capture named 'capture.cap'. The filter field is set to 'tcp'. The packet list shows 46 packets, with the first 24 packets highlighted in green and the last two in dark blue. The packet details pane is empty.

No.	Time	Source	Destination	Protocol	Length	Info
13	1.510508	10.0.0.11	172.16.0.40	TCP	74	59808 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=14
14	1.510538	172.16.0.40	10.0.0.11	TCP	74	80 → 59808 [SYN, ACK] Seq=0 Ack=1 Win=28960 Le
15	1.510544	10.0.0.11	172.16.0.40	TCP	66	59808 → 80 [ACK] Seq=1 Ack=1 Win=29696 Len=0 T
16	1.510608	10.0.0.11	172.16.0.40	HTTP	377	GET / HTTP/1.1
17	1.510613	172.16.0.40	10.0.0.11	TCP	66	80 → 59808 [ACK] Seq=1 Ack=312 Win=30208 Len=
18	1.513728	172.16.0.40	10.0.0.11	TCP	304	80 → 59808 [PSH, ACK] Seq=1 Ack=312 Win=30208
19	1.513732	10.0.0.11	172.16.0.40	TCP	66	59808 → 80 [ACK] Seq=312 Ack=239 Win=30720 Le
20	1.514171	172.16.0.40	10.0.0.11	HTTP	678	HTTP/1.1 200 OK (text/html)
21	1.514174	10.0.0.11	172.16.0.40	TCP	66	59808 → 80 [ACK] Seq=312 Ack=851 Win=31744 Le
22	1.635983	10.0.0.11	172.16.0.40	HTTP	358	GET /favicon.ico HTTP/1.1
23	1.636056	172.16.0.40	10.0.0.11	HTTP	390	HTTP/1.1 404 Not Found (text/html)
24	1.642659	10.0.0.11	172.16.0.40	TCP	66	59808 → 80 [ACK] Seq=604 Ack=1175 Win=32768 L
45	11.668634	10.0.0.11	172.16.0.40	TCP	66	[TCP Keep-Alive] 59808 → 80 [ACK] Seq=603 Ack=11
46	11.668678	172.16.0.40	10.0.0.11	TCP	66	[TCP Keep-Alive ACK] 80 → 59808 [ACK] Seq=1175 A