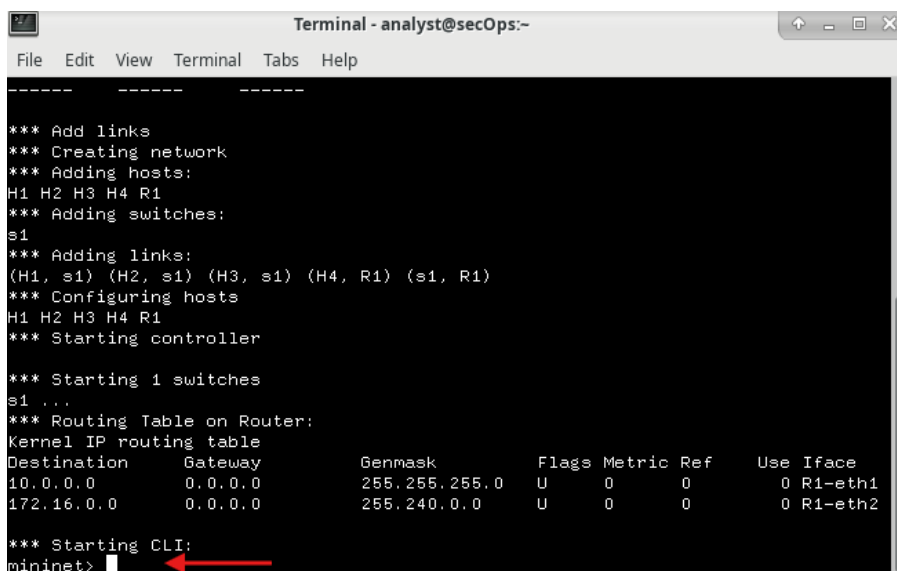


CYBEROPS PRACTICE 1

- **Part 1: Prepare the Hosts to Capture the Traffic**

Per prima cosa avvio **Mininet** sulla macchina virtuale, con il comando “**sudo lab.support.files/scripts/cyberops_topo.py**”.

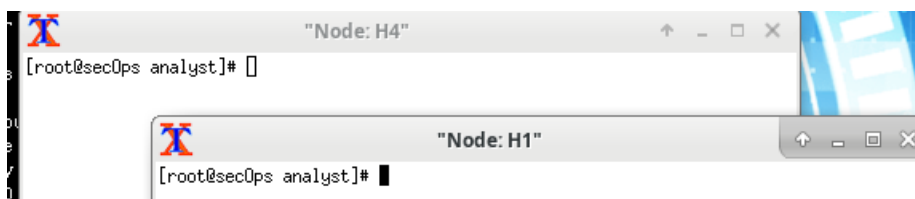


```
Terminal - analyst@secOps:~
File Edit View Terminal Tabs Help

*** 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> 
```

Avvio le due shell **xterm** coi comandi “**xterm H1**” e “**xterm H4**”.

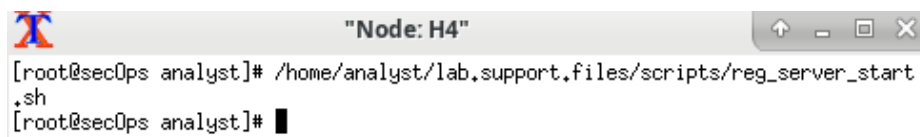


```
"Node: H4"
[root@secOps analyst]#

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

Avvio il web server sulla shell **H4** col comando

“**/home/analyst/lab.support.files/scripts/reg_server_start.sh**”.



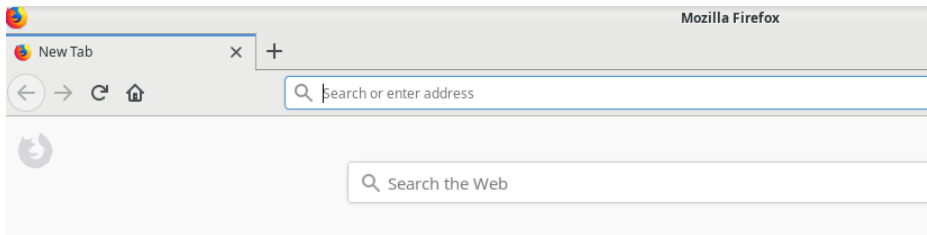
```
"Node: H4"
[root@secOps analyst]# /home/analyst/lab.support.files/scripts/reg_server_start
.sh
[root@secOps analyst]# 
```

Eseguo il comando “**su analyst**” sulla shell **H1** per cambiare utente.



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

Avvio firefox dalla shell **H1** col comando **“firefox &”**.



Col comando **“sudo tcpdump -i H1-eth0 -v -c 50 -w /home/analyst/capture.pcap”** avvio una sessione di **tcpdump** su **H1** che catturerà 50 pacchetti e scriverà il risultato della cattura sul file **capture.pcap**.

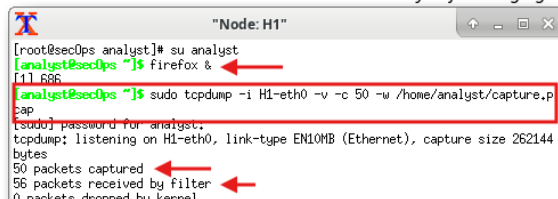


Welcome to nginx! ←

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

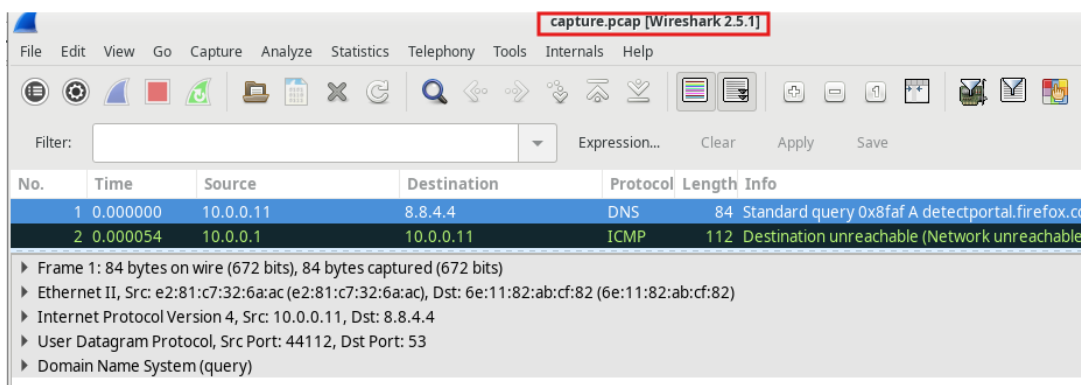
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.



- **Part 2: Analyze the Packets using Wireshark**
Step 1: Apply a filter to the saved capture.

Va aperto il file appena creato con **Wireshark**.



Applico un filtro per mostrare tutti i pacchetti **TCP**. Serve analizzare i pacchetti che hanno come destinazione l'IP **172.16.0.40**.

No.	Time	Source	Destination	Protocol	Length	Info
25	9.942782	10.0.0.11	172.16.0.40	TCP	74	52804 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=
26	9.943124	172.16.0.40	10.0.0.11	TCP	74	80 → 52804 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=
27	9.943220	10.0.0.11	172.16.0.40	TCP	66	52804 → 80 [ACK] Seq=1 Ack=1 Win=29696 Len=0
28	9.943567	10.0.0.11	172.16.0.40	HTTP	377	GET / HTTP/1.1
29	9.943577	172.16.0.40	10.0.0.11	TCP	66	80 → 52804 [ACK] Seq=1 Ack=312 Win=30208 Len=0
30	9.948577	172.16.0.40	10.0.0.11	TCP	304	80 → 52804 [PSH, ACK] Seq=1 Ack=312 Win=30208 Len=0
31	9.948580	10.0.0.11	172.16.0.40	TCP	66	52804 → 80 [ACK] Seq=312 Ack=239 Win=30720 Len=0
32	9.949379	172.16.0.40	10.0.0.11	HTTP	678	HTTP/1.1 200 OK (text/html)
33	9.949382	10.0.0.11	172.16.0.40	TCP	66	52804 → 80 [ACK] Seq=312 Ack=851 Win=31744 Len=0
36	10.041458	10.0.0.11	172.16.0.40	HTTP	358	GET /favicon.ico HTTP/1.1
37	10.041899	172.16.0.40	10.0.0.11	HTTP	390	HTTP/1.1 404 Not Found (text/html)
38	10.042119	10.0.0.11	172.16.0.40	TCP	66	52804 → 80 [ACK] Seq=604 Ack=1175 Win=32768

Step 2: Examine the information within packets including IP addresses, TCP port numbers, and TCP control flags.

Qui appaiono i pacchetti relativi al **Three Way Handshake**.

No.	Time	Source	Destination	Protocol	Length	Info
25	9.942782	10.0.0.11	172.16.0.40	TCP	74	52804 → 80 [SYN] seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
26	9.943124	172.16.0.40	10.0.0.11	TCP	74	80 → 52804 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SA
27	9.943220	10.0.0.11	172.16.0.40	TCP	66	52804 → 80 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=2610018483

Porta sorgente di tipo **dinamico**: **52804**. Porta di destinazione di tipo **well-known**: **80(http)**.

▼ Transmission Control Protocol, Src Port: 52804, Dst Port: 80, Seq: 0, Len: 0
Source Port: 52804
Destination Port: 80

Il **Sequence Number** è **0**.

Sequence number: 0 (relative sequence number)

[Next sequence number: 0 (relative sequence number)]

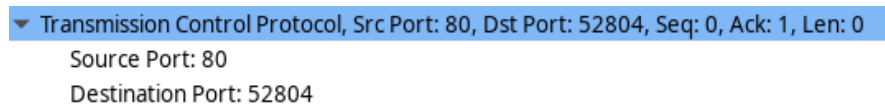
Acknowledgment number: 0

1010 = Header Length: 40 bytes (10)

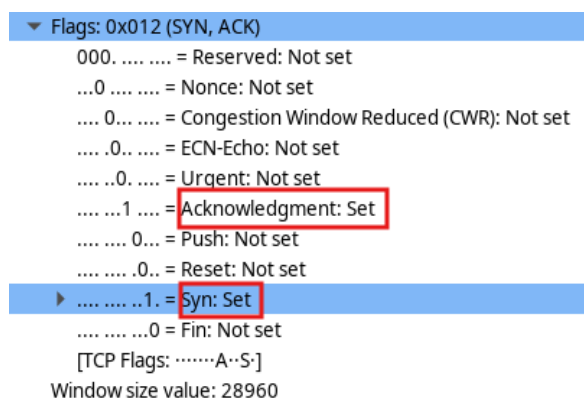
L'unico **flag** settato a **1** è quello relativo al **SYN**.



Nel secondo pacchetto della trasmissione appena avvenuta le porte sono invertite, perché la comunicazione adesso avviene dall'IP remoto verso la macchina, quindi sorgente destinazione sono invertiti.



I **flag** in settati sono **SYN** e **ACK**, dato che si tratta della 2° fase del **Three Way Handshake**.



SEQ = 0 ACK = 1

Sequence number: 0 (relative sequence number)
[Next sequence number: 0 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)

Nel terzo pacchetto, ovvero la terza fase del TWH, il flag settato sarà ACK.

▼ Flags: 0x010 (ACK)

- 000. = Reserved: Not set
- ...0 = Nonce: Not set
- 0... = Congestion Window Reduced (CWR): Not set
-0.. = ECN-Echo: Not set
-0. = Urgent: Not set
-1 = Acknowledgment: Set
- 0... = Push: Not set
-0.. = Reset: Not set
-0. = Syn: Not set
-0 = Fin: Not set

• Part 3: View the packets using tcpdump

All'interno del **man** relativo a **tcpdump**, se si digita **"/** si può effettuare una ricerca all'interno del testo (ad esempio lo switch **"-r"**).

```
[analyst@sec0ps ~]$ man tcpdump
```

```
[ -f file ] [ -U file ] [ -s snaplen ] [ -T type ] [ -w file ]  
[ -U filecount ]  
[ -E spi@ipaddr also:secret,... ]  
[ -y datalinktype ] [ -z postrotate-command ] [ -Z user ]  
[ --time-stamp-precision=timestamp-precision ]  
[ --immediate-mode ] [ --version ]  
[ expression ]  
  
DESCRIPTION  
tcpdump prints out a description of the contents of packets on a net-  
work interface that match the boolean expression; the description is  
preceded by a time stamp, printed, by default, as hours, minutes, sec-  
onds, 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 -f 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 -U 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 sig-
```

Con “?” si può fare lo stesso. In questo caso lo switch “-r” serve per leggere i pacchetti da un file anziché dalla rete.

```
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 -U 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 sig-
nal (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:

    packets ``captured`` (this is the number of packets that tcpdump
    has received and processed);

    packets ``received by filter`` (the meaning of this depends on
    the OS on which you're running tcpdump, and possibly on the way
    the OS was configured - if a filter was specified on the command
    line, on some OSes it counts packets regardless of whether they
    were matched by the filter expression and, even if they were

?-r
```

Con “**tcpdump -r /home/analyst/capture.pcap tcp -c 3**” si possono visionare i primi 3 pacchetti catturati nel terminale.

```
[analyst@sec0ps ~]$ tcpdump -r /home/analyst/capture.pcap tcp -c 3
reading from file /home/analyst/capture.pcap, link-type EN10MB (Ethernet)
07:36:34.551572 IP 10.0.0.11.52804 > 172.16.0.40.http: Flags [S], seq 942473942, win 29200, opt
ions [mss 1460,sackOK,TS val 2610018482 ecr 0,nop,wscale 9], length 0
07:36:34.551914 IP 172.16.0.40.http > 10.0.0.11.52804: Flags [S.], seq 798089173, ack 942473943
, win 28960, options [mss 1460,sackOK,TS val 4215513 ecr 2610018482,nop,wscale 9], length 0
07:36:34.552010 IP 10.0.0.11.52804 > 172.16.0.40.http: Flags [.], ack 1, win 58, options [nop,n
op,TS val 2610018483 ecr 4215513], length 0
```

Con “quit” possiamo terminare mininet.

```
mininet> quit
*** Stopping 0 controllers

*** Stopping 2 terms
*** Stopping 5 links
....
*** Stopping 1 switches
s1
*** Stopping 5 hosts
H1 H2 H3 H4 R1
*** Done
```

Con sudo “mn -c” si può fare la pulizia dei processi che sono stati avviati da terminale con mininet.

```
[analyst@sec0ps ~]$ sudo mn -c
[sudo] password for analyst:
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall1 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller
udpbwtest mnexec ivs 2> /dev/null
killall1 -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-control
ler udpbwtest mnexec ivs 2> /dev/null
pkill1 -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
*** Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill1 -9 -f mininet:
*** Shutting down stale tunnels
pkill1 -9 -f Tunnel=Ethernet
pkill1 -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
```

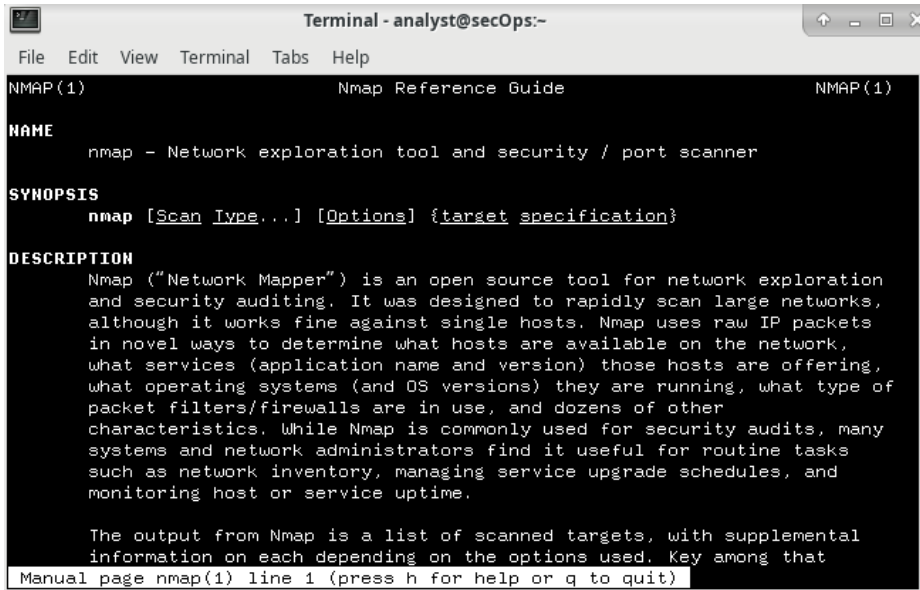
• Reflection Questions

1. Filtri utili per l'analisi del traffico su Wireshark: tcp, udp, http, ip=="..."
2. ...

CYBEROPS PRACTICE 2

- **Part 1: Exploring Nmap**

Nmap è un tool per eseguire il port scanning e l'analisi della rete. Viene usato per scovare le caratteristiche delle macchine connesse alla rete, conoscerne le vulnerabilità attraverso lo scanning delle porte di rete e dei servizi attivi su di esse. Si effettua la ricerca in avanti con "/" e indietro con "?" post ponendo a questi simboli in contenuto che si vuole trovare all'interno del testo. Con "n" si salta al successivo termine corrispondente alla nostra ricerca.

A screenshot of a terminal window titled "Terminal - analyst@secOps:-". The window displays the "Nmap Reference Guide" for NMAP(1). The guide includes sections for NAME, SYNOPSIS, and DESCRIPTION. The NAME section identifies nmap as a network exploration tool and port scanner. The SYNOPSIS section shows the command syntax: nmap [Scan Type...] [Options] {target specification}. The DESCRIPTION section explains that Nmap is an open-source tool for network exploration and security auditing, designed to scan large networks and determine host availability, services, and operating systems. It also mentions that the output is a list of scanned targets with supplemental information. At the bottom, a prompt indicates the user can press 'h' for help or 'q' to quit.

```
Terminal - analyst@secOps:-
File Edit View Terminal Tabs Help
NMAP(1) Nmap Reference Guide NMAP(1)

NAME
  nmap - Network exploration tool and security / port scanner

SYNOPSIS
  nmap [Scan Type...] [Options] {target specification}

DESCRIPTION
  Nmap ("Network Mapper") is an open source tool for network exploration
  and security auditing. It was designed to rapidly scan large networks,
  although it works fine against single hosts. Nmap uses raw IP packets
  in novel ways to determine what hosts are available on the network,
  what services (application name and version) those hosts are offering,
  what operating systems (and OS versions) they are running, what type of
  packet filters/firewalls are in use, and dozens of other characteristics.
  While Nmap is commonly used for security audits, many systems and network
  administrators find it useful for routine tasks such as network inventory,
  managing service upgrade schedules, and monitoring host or service uptime.

  The output from Nmap is a list of scanned targets, with supplemental
  information on each depending on the options used. Key among that

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

Ad esempio il comando "**nmap -A -T4 scanme.nmap.org**" effettua una scansione completa dell'indirizzo passato come argomento. Lo switch "**-A**" effettua sia la scansione delle porte e dei servizi attivi che l'**OS fingerprinting**, nonché il **traceroute**. Lo switch "**-T4**" indica la velocità alla quale svolgere la scansione. Più è alto il numero (max 5) e più rapida sarà, col rischio di essere rilevati da un eventuale sistema **IDS** o **IPS**. Più basso è il valore dello switch e maggiore sarà il ritardo temporale tra l'invio dei pacchetti che fa **nmap**.

- **Part 2: Scanning for Open Ports**

Col comando “**nmap -A -T4 localhost**” si effettua la scansione della macchina in locale.

Si nota che ci sono 2 porte aperte: **21/tcp (ftp)** e **22/tcp (ssh)**.

Il servizio **FTP** è dato da **vsFTPD** versione **3.0.3**, mentre quello **SSH** è dato da **OpenSSH** versione **2.0**.

```
[analyst@sec0ps ~]$ nmap -A -T4 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2024-09-03 09:02 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000037s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.0.8 or later
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--  1 0          0          0 Mar 26  2018 ftp_test
| ftp-syst:
|   STAT:
|   FTP server status:
|     Connected to 127.0.0.1
|     Logged in as ftp
|     TYPE: ASCII
|     No session bandwidth limit
|     Session timeout in seconds is 300
|     Control connection is plain text
|     Data connections will be plain text
|     At session startup, client count was 6
|     vsFTPD 3.0.3 - secure, fast, stable
|_-End of status
22/tcp    open  ssh      OpenSSH 7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 b4:91:f9:d6:79:25:86:44:c7:9e:f8:e0:e7:5b:bb (RSA)
|   256 06:12:75:fe:b3:89:29:4f:8d:f3:9e:9a:d7:c6:03:52 (ECDSA)
|_  256 34:5d:f2:d3:5b:9f:b4:b6:08:96:a7:30:52:8c:96:06 (ED25519)
Service Info: Host: Welcome

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.82 seconds
```

Step 2: Scan your network

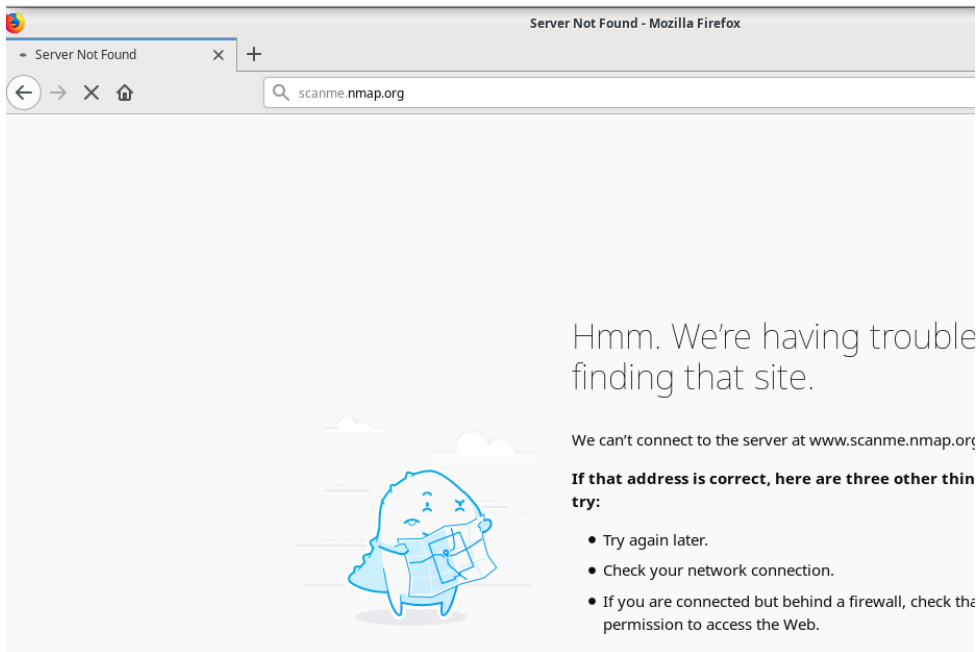
Col comando “**ip address**” si può verificare l’IP locale della macchina virtuale, ovvero **10.0.2.15/24**.

```
[analyst@sec0ps ~]$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
    link/ether 08:00:27:49:2c:f3 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 79497sec preferred_lft 79497sec
    inet6 fe80::a00:27ff:fe49:2cf3/64 scope link
        valid_lft forever preferred_lft forever
```

Col comando “**nmap -A -T4 10.0.2.0/24**” cerchiamo di ottenere informazioni sugli **host** connessi alla stessa rete **LAN**. In questo caso risulterà solo l’IP della macchina virtuale.

Step 3: Scan a remote server

Purtroppo dal browser non riesco ad accedere alla pagina...



Dopo la scansione dell'indirizzo web col comando **"nmap -A -T4 scanme.nmap.org"** troviamo dei servizi attivi su diverse porte aperte.

1. 22/tcp OpenSSH 6.6.1p1
2. 80/tcp http
3. 9929/tcp Nping echo
4. 31337/tcp tcpwrapped

```
[analyst@secOps ~]$ nmap -A -T4 scanme.nmap.org
Starting Nmap 7.70 ( https://nmap.org ) at 2024-09-03 09:24 EDT
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.22s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 996 filtered ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 1024 ac:00:a0:1a:82:ff:cc:55:99:dc:67:2b:34:97:6b:75 (DSA)
| 2048 20:3d:2d:44:62:2a:b0:5a:9d:b5:b3:05:14:c2:a6:b2 (RSA)
| 256 96:02:bb:5e:57:54:1c:4e:45:2f:56:4c:4a:24:b2:57 (ECDSA)
| 256 33:fa:91:0f:e0:e1:7b:1f:6d:05:a2:b0:f1:54:41:56 (ED25519)
80/tcp    open  http?
9929/tcp  open  nping-echo   Nping echo
31337/tcp open  tcpwrapped
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 153.26 seconds
```

Ci sono ben **996** porte filtrate che non vengono mostrate.

```
Not shown: 996 filtered ports
```

L'IP del server di **scanme.nmap.org** è **45.33.32.156**. L'OS usato è **Ubuntu Linux**.

```
scanme.nmap.org (45.33.32.156) ←  
ncnmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:  
red ports  
ICE    VERSION  
      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux;
```

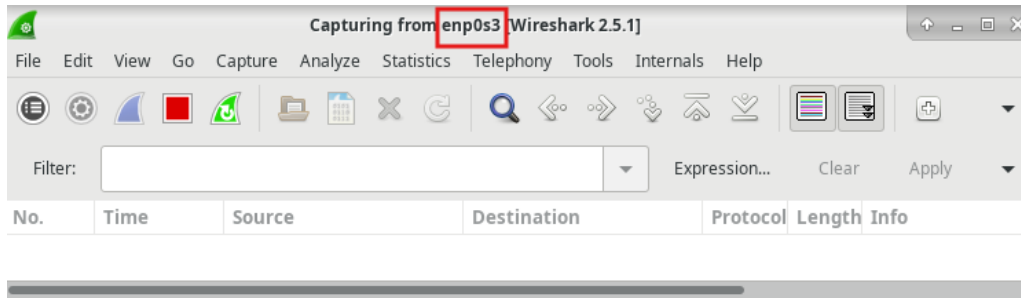
Reflection Questions

In conclusione si comprende bene il fatto che **nmap** potrebbe benissimo essere usato da un malintenzionato per scovare le vulnerabilità di un host. Infatti, venendo a conoscenza delle porte aperte e dei servizi attivi su di esse, potrebbe cercare online la documentazione su quella specifica versione del servizio per apprendere eventuali falle software e provare ad utilizzare un **exploit** per fare breccia nel sistema ed iniettare codice malevolo per prendere il controllo del sistema oppure per danneggiarlo.

CYBEROPS PRACTICE 3

- **Part 1: Identify TCP Header Fields and Operation Using a Wireshark FTP Session Capture**

Avvio **wireshark** sull'interfaccia di rete **enp0s3**.



Eseguo l'accesso al server ftp.cdc.gov da terminale.

```
[analyst@secOps ~]$ ftp ftp.cdc.gov
Connected to ftp.cdc.gov.
220 Microsoft FTP Service
Name (ftp.cdc.gov:analyst):
```

Purtroppo non si riesce ad effettuare l'accesso al server sopracitato...

```
[analyst@secOps ~]$ ftp ftp.cdc.gov
Connected to ftp.cdc.gov.
220 Microsoft FTP Service
Name (ftp.cdc.gov:analyst): anonymous
331 Valid hostname is expected.
Password:
503 Login with USER first.
ftp: Login failed.
Remote system type is Windows_NT.
ftp> quit
[analyst@secOps ~]$ ftp ftp.cdc.gov
Connected to ftp.cdc.gov.
220 Microsoft FTP Service
Name (ftp.cdc.gov:analyst): anonymous
331 Valid hostname is expected.
Password:
503 Login with USER first.
ftp: Login failed.
Remote system type is Windows_NT.
ftp>
```

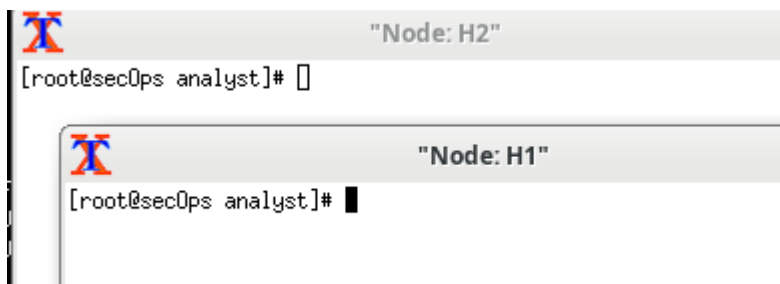
- **Part 2: Identify UDP Header Fields and Operation Using a Wireshark TFTP Session Capture**

Avvio Mininet col comando “`sudo lab.support.files/scripts/cyberops_topo.py`”.

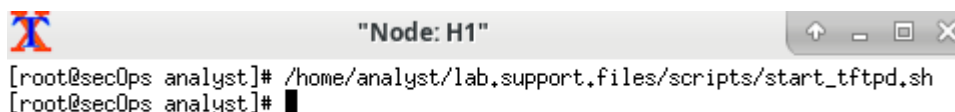
```
*** Starting 1 switches
s1 ...
*** Routing Table on Router:
Kernel IP routing table
Destination      Gateway
10.0.0.0          0.0.0.0
172.16.0.0        0.0.0.0

*** Starting CLI:
mininet> █
```

Avvio due shell **xterm** H1 e H2.

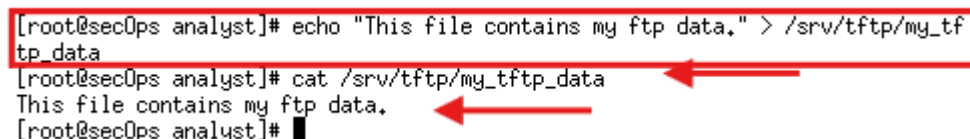


Col comando “`/home/analyst/lab.support.files/scripts/start_tftpd.sh`” avvio **tftpd** server dalla shell H1.



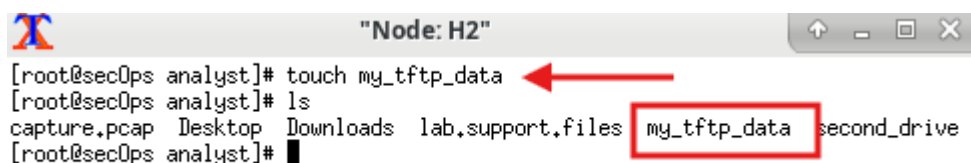
```
"Node: H1"
[root@secOps analyst]# /home/analyst/lab.support.files/scripts/start_tftpd.sh
[root@secOps analyst]# █
```

Creo un file all'interno della cartella `/srv/tftp/` col comando “`echo "This file contains my tftp data." > /srv/tftp/my_tftp_data`”. Verifico col comando `cat` “`nomefile`” il suo contenuto.



```
[root@secOps analyst]# echo "This file contains my ftp data." > /srv/tftp/my_tftp_data
[root@secOps analyst]# cat /srv/tftp/my_tftp_data
This file contains my ftp data.
[root@secOps analyst]# █
```

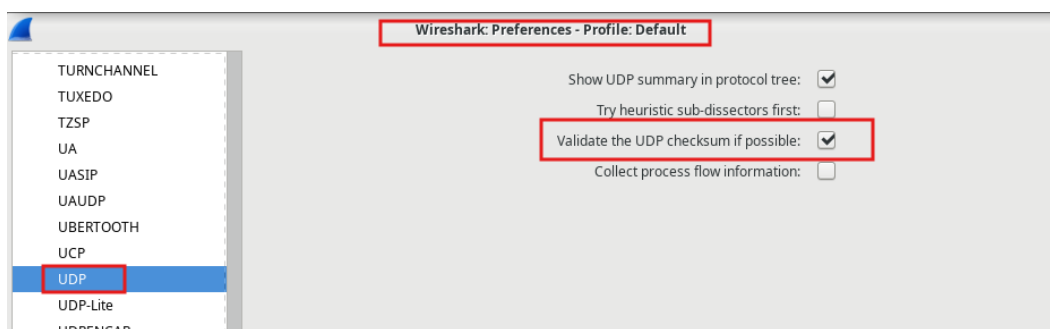
Sulla **shell H2** creo un secondo file nominato sempre **my_tftp_data** col comando **touch**.



```
[root@secOps analyst]# touch my_tftp_data
[root@secOps analyst]# ls
capture.pcap Desktop Downloads lab.support.files my_tftp_data second_drive
[root@secOps analyst]#
```

Step 3: Capture a TFTP session in Wireshark

Avvio **wireshark** e inserisco il flag come richiesto da traccia.



La traccia richiede di connettersi al server **tftp** aperto sulla **shell xterm H1** per scaricare il file **my_tftp_data**, col comando "**tftp 10.0.0.11 -c get my_tftp_data**", però anche modificando l'IP su quello della macchina virtuale, non riesce a connettersi in nessun modo.

CYBEROPS PRACTICE 4

- **Part 1: Capture and View HTTP Traffic**

Listo le interfacce di rete **lo** e **enp0s3** per conoscere l'IP.

Enp0s3 → 10.0.2.15/24

Lo → 127.0.0.1/8

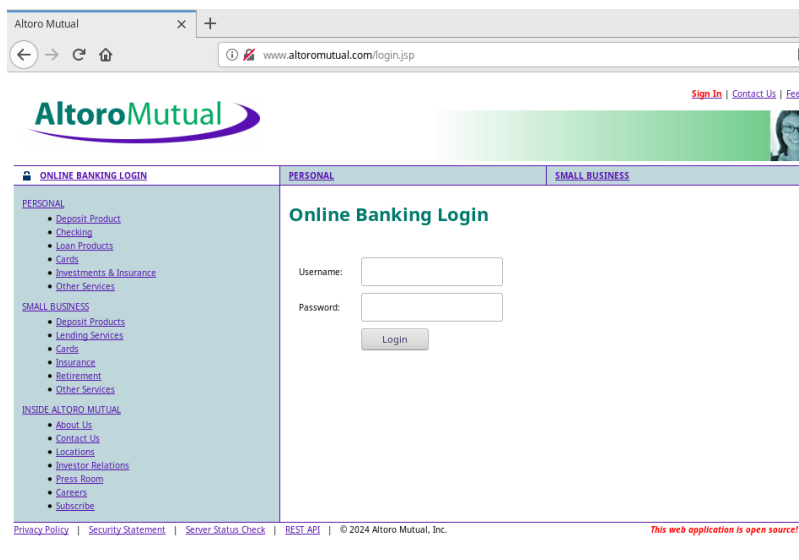
```
[analyst@secOps ~]$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:49:2c:f3 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 72296sec preferred_lft 72296sec
    inet6 fe80::a00:27ff:fe49:2cf3/64 scope link
        valid_lft forever preferred_lft forever

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
```

Col comando “**sudo tcpdump -i enp0s3 -s 0 -w httpdump.pcap**” mettiamo in ascolto **tcpdump** sull'interfaccia **enp0s3**, e salviamo il traffico di rete sul file **httpdump.pcap**.

```
[analyst@secOps ~]$ sudo tcpdump -i enp0s3 -s 0 -w httpdump.pcap
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
```


Poi avviamo Firefox per navigare verso l'indirizzo **http://www.altoromutual.com/login.jsp**.



Il sito usa http, quindi ci avvisa che se effettuiamo il login la nostra password potrebbe essere compromessa.

Username:

Password:

 This connection is not secure.
Logins entered here could be compromised. [Learn More](#)

Effettuiamo login con credenziali **Admin Admin**.

[MY ACCOUNT](#) | [PERSONAL](#) | [SMALL BUSINESS](#)

I WANT TO ...

- [View Account Summary](#)
- [View Recent Transactions](#)
- [Transfer Funds](#)
- [Search News Articles](#)
- [Customize Site Language](#)

ADMINISTRATION

- [Edit Users](#)

Hello Admin User

Welcome to Altoro Mutual Online.

View Account Details: 800000 Corporate GO

Congratulations!

You have been pre-approved for an Altoro Gold Visa with a credit limit of \$10000!

Click [Here](#) to apply.

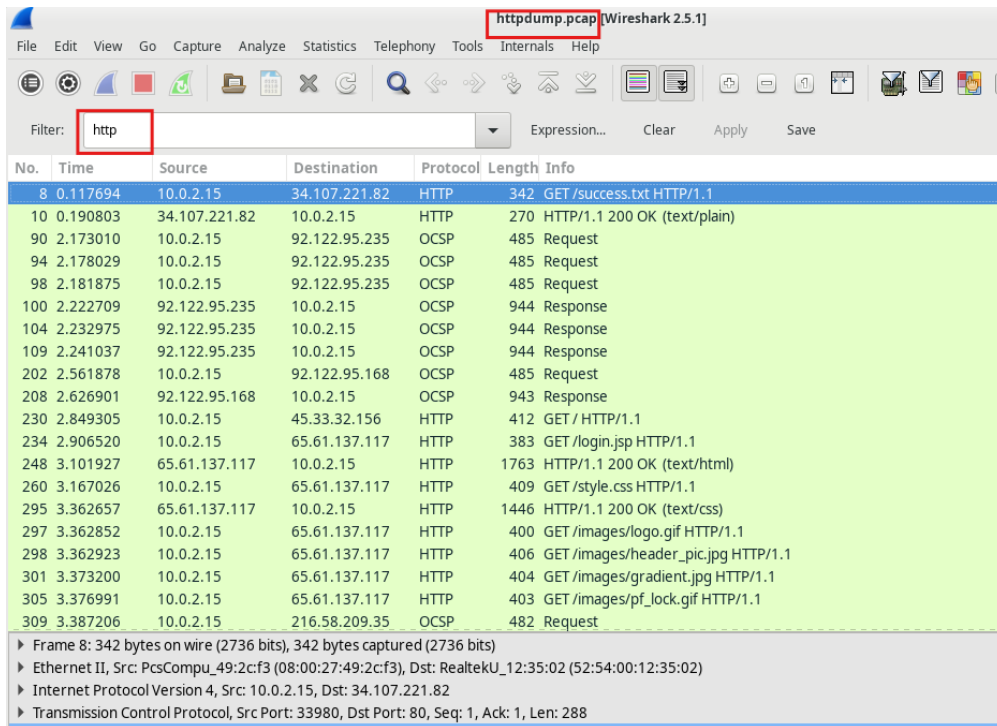
[Privacy Policy](#) | [Security Statement](#) | [Server Status Check](#) | [REST API](#) | © 2024 Altoro Mutual, Inc. This web application is open sou.

Chiudo il browser e con **Ctrl+C** fermo **tcpdump** dal terminale.

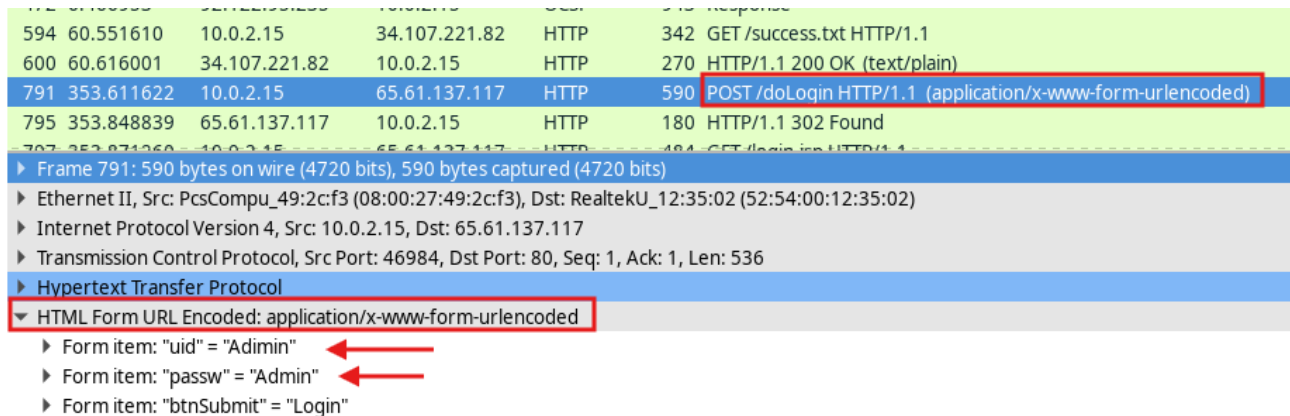
```
^C873 packets captured
873 packets received by filter
0 packets dropped by kernel
[analyst@secOps ~]$
```


Step 3: View the HTTP capture.

Apro il file **httpdump.pcap** su **Wireshark** e imposto il filtro su **http**.



Seleziono il pacchetto con la richiesta **POST** e dentro la voce **HTML Form**, se la espando cliccando sulla freccetta, vedo le credenziali utente in chiaro.

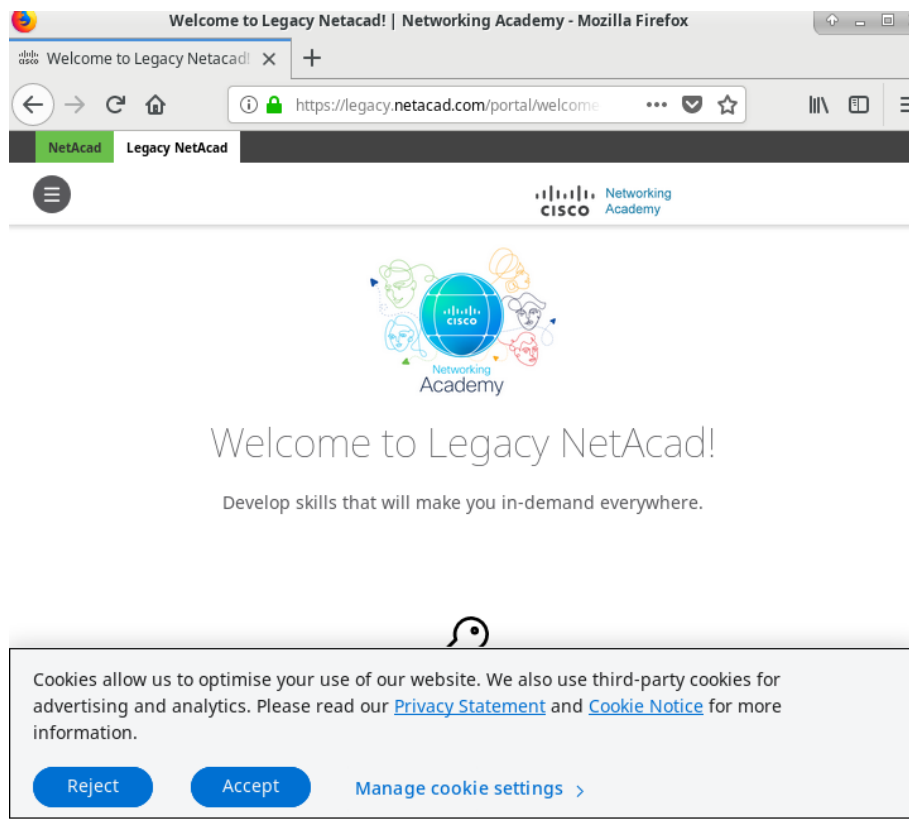


• Part 2: Capture and View HTTPS Traffic

Avvio **tcpdump** in ascolto sull'interfaccia di rete **enp0s3**.

```
[analyst@secOps ~]$ sudo tcpdump -i enp0s3 -s 0 -w httpsdump.pcap
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
```

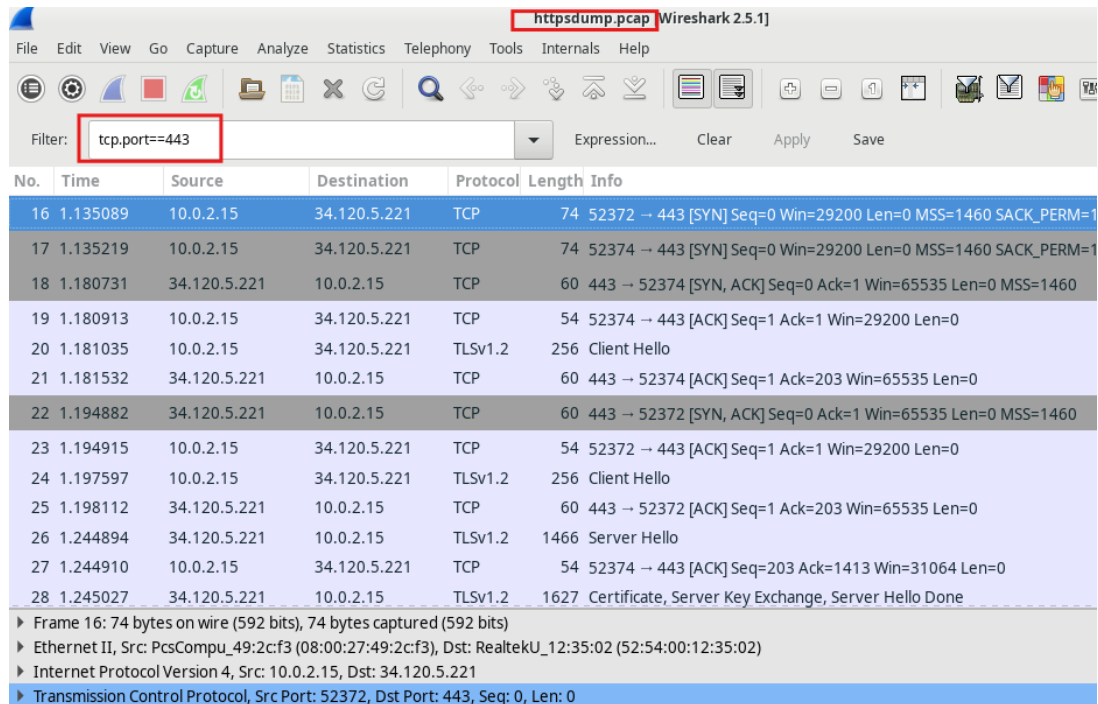
Poi ci fa collegare alla pagina **www.netacad.com/**.



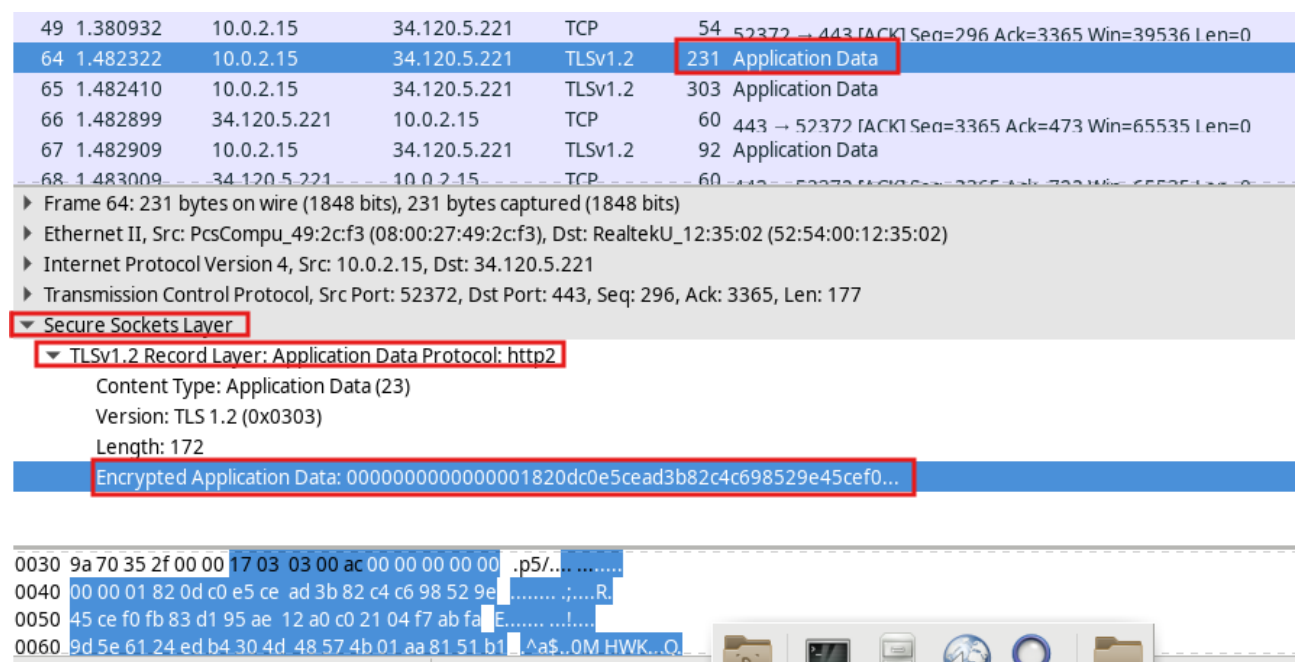
Proviamo a fare Login con le nostre credenziali. Poi fermo la cattura con **tcpdump** e chiudo il browser.

Step 2: View the HTTPS capture

Apro il file **httpsdump.pcap** su **wireshark** ed imposto il filtro **tcp.port==443**.



Seleziono, come da traccia, un pacchetto non la denominazione **“Application Data”**, e noto che al posto del sottomenu **HTTP Form** adesso c'è **Secure Socket Layer**. Adesso le credenziali che sono state inviate al form del sito per effettuare il login sono state criptate tramite **TLSv1.2** e sono indecifrabili.



In conclusione, si può affermare che il vantaggio enorme che offre HTTPS è la sicurezza delle proprie informazioni personali.