

S13 / L2

WIRESHARK AND NMAP

# CONTENUTI

00

Traccia

01

Using Wireshark to Observe the TCP 3-Way Handshake

02

Exploring Nmap

03

Using Wireshark to Examine TCP and UDP Captures

04

Using Wireshark to Examine HTTP and HTTPS Traffic

# 00 TRACCIA

## 1) Using Wireshark to Observe the TCP 3-Way Handshake

In this lab, you will complete the following objectives:

- Part 1: Prepare the Hosts to Capture the Traffic
- Part 2: Analyze the Packets using Wireshark
- Part 3: View the Packets using tcpdump

## 2) Exploring Nmap

Port scanning is usually part of a reconnaissance attack. There are a variety of port scanning methods that can be used. We will explore how to use the Nmap utility. Nmap is a powerful network utility that is used for network discovery and security auditing.

# 00 TRACCIA

## **3) Using Wireshark to Examine TCP and UDP Captures**

In this lab, you will complete the following objectives:

- Identify TCP Header Fields and Operation Using a Wireshark FTP Session Capture
- Identify UDP Header Fields and Operation Using a Wireshark TFTP Session Capture

## **4) Using Wireshark to Examine HTTP and HTTPS Traffic**

In this lab, you will complete the following objectives:

- Capture and view HTTP traffic
- Capture and view HTTPS traffic

# 01 - USING WIRESHARK TO OBSERVE THE TCP 3-WAY HANDSHAKE

In questa parte del laboratorio, ho avviato la macchina virtuale **CyberOps** e mi sono autenticato con l'utente "analyst". Ho avviato **Mininet** e acceso gli host **H1** e **H4**. Ho configurato H4 come web server e ho eseguito un browser su **H1** dopo aver cambiato l'utente da root a analyst. Ho poi avviato una sessione **tcpdump** su H1 per catturare i pacchetti e ho navigato all'indirizzo IP del server web, 172.16.0.40, con Firefox per generare traffico.

```
[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         0.0.0.0         U        0     0        0 R1-eth2

*** Starting CLI:
mininet>
```

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

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

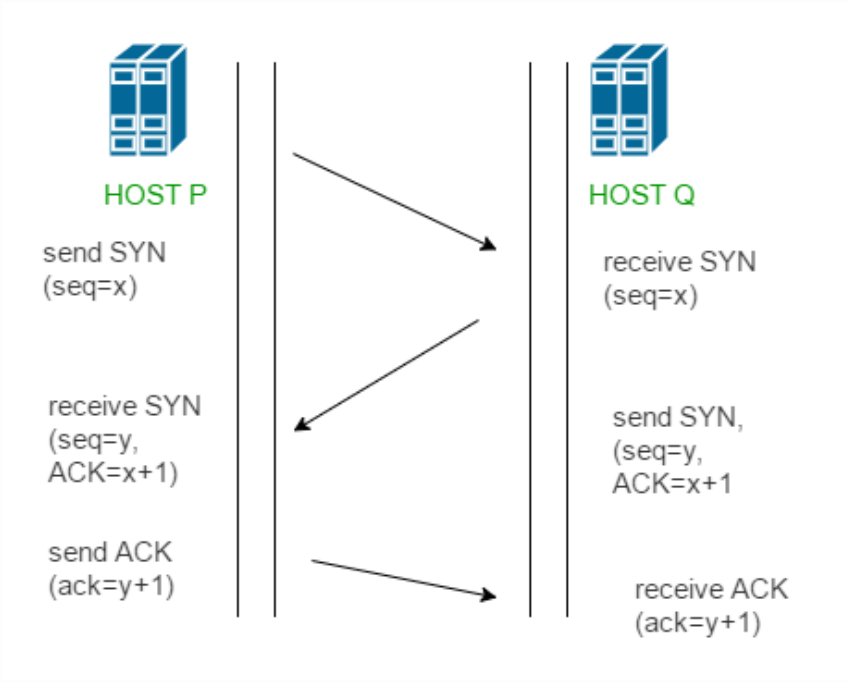
172.16.0.40

## Welcome to nginx!

```
"Node: H1"
[analyst@secOps ~]$ firefox &
[2] 993
[1] Done
[analyst@secOps ~]$ sudo tcpdump -i H1-eth0 -v -c 50 -w /home/analyst/capture.p
cap
tcpdump: listening on H1-eth0, link-type EN10MB (Ethernet), capture size 262144
bytes
50 packets captured
51 packets received by filter
0 packets dropped by kernel
[analyst@secOps ~]$
```

# 01 - USING WIRESHARK TO OBSERVE THE TCP 3-WAY HANDSHAKE

Dopo aver catturato i pacchetti, ho aperto il file pcap salvato utilizzando **Wireshark**. Ho applicato un **filtro TCP** per concentrarmi sul traffico di interesse, in particolare sui primi tre pacchetti che rappresentano la stretta di mano a tre vie TCP. Ho esaminato i dettagli del primo pacchetto, identificando il numero di porta sorgente e di destinazione, i flag impostati (come SYN) e il numero di sequenza relativo. Successivamente, ho analizzato i pacchetti successivi per osservare la risposta del server e la conferma finale della connessione TCP.



Filter: tcp							Expression...		Clear	Apply	Save
No.	Time	Source	Destination	Protocol	Length	Info					
6	2.831370	10.0.0.11	172.16.0.40	TCP	74	40966 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=28000					
7	2.831410	172.16.0.40	10.0.0.11	TCP	74	80 → 40966 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 T					
8	2.831423	10.0.0.11	172.16.0.40	TCP	66	40966 → 80 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=2800064207 TSecr=14934					

## 3-Way Handshake

Transmission Control Protocol, Src Port: 40966, Dst Port: 80, Seq: 0, Len: 0

Source Port: 40966

Destination Port: 80

[TCP Segment Len: 0]

Sequence number: 0 (relative sequence number)

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

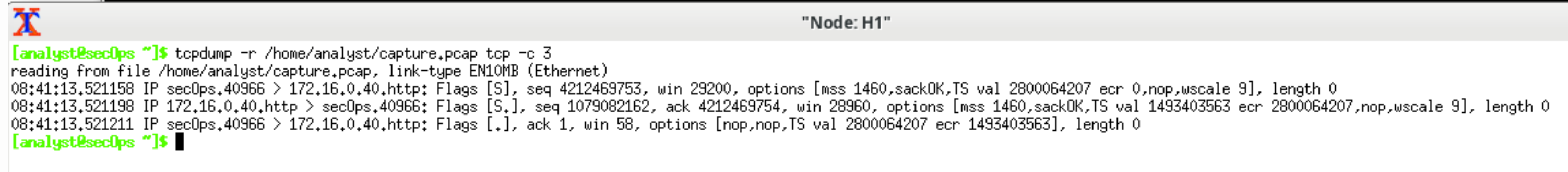
Acknowledgment number: 0

1010 .... = Header Length: 40 bytes (10)

Flags: 0x002 (SYN)

# 01 - USING WIRESHARK TO OBSERVE THE TCP 3-WAY HANDSHAKE

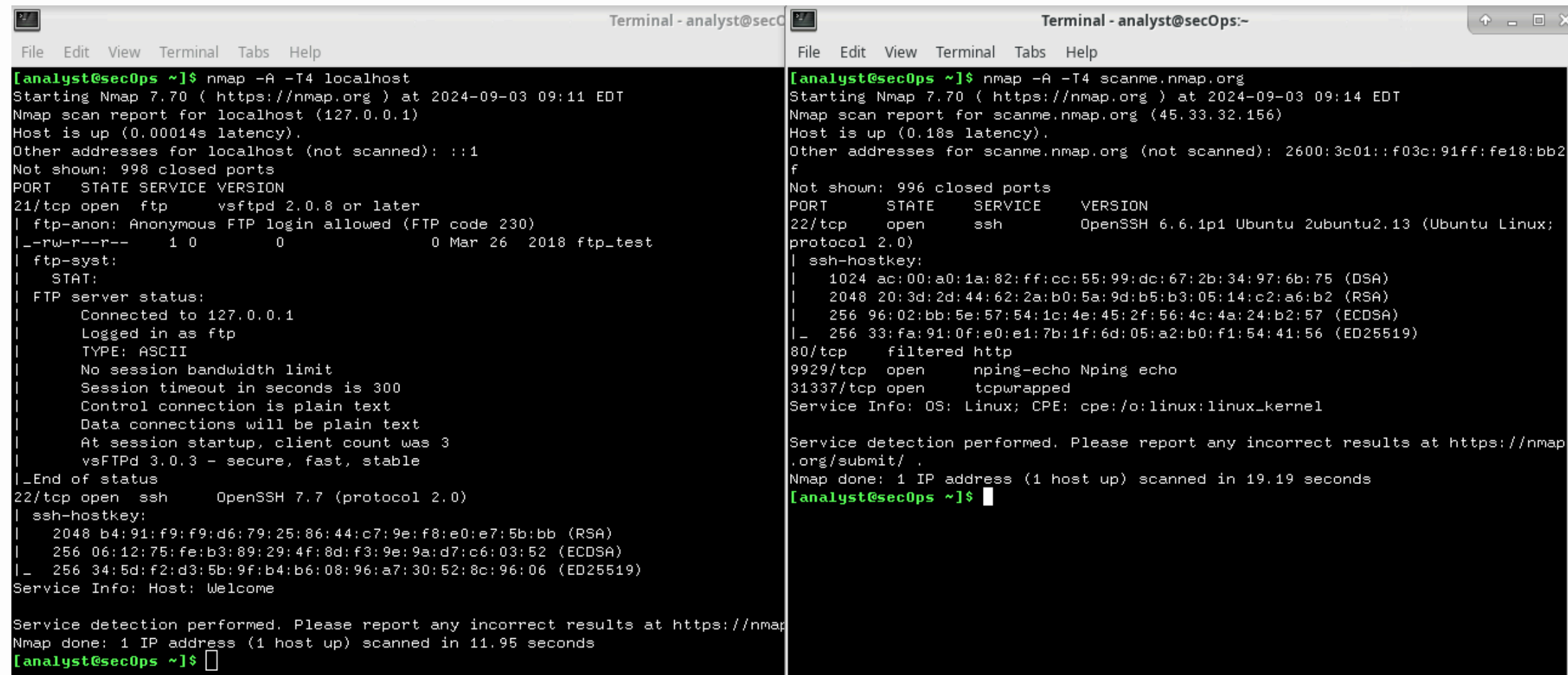
Ho utilizzato il comando **tcpdump** per visualizzare i pacchetti catturati nel file pcap direttamente dal terminale. Ho esplorato le opzioni disponibili per il comando tcpdump utilizzando il manuale (man tcpdump) e ho verificato l'opzione -r, che permette di leggere pacchetti da un file precedentemente salvato. Ho quindi visualizzato i primi tre pacchetti TCP catturati, confermando i dettagli della stretta di mano a tre vie.

A screenshot of a terminal window titled "Node: H1" with a red 'X' icon. The terminal shows the execution of the tcpdump command to read from a pcap file and display the first three TCP packets. The output shows a SYN packet, an ACK packet, and an ACK packet, confirming the three-way handshake.

```
[analyst@secOps ~]$ tcpdump -r /home/analyst/capture.pcap tcp -c 3
reading from file /home/analyst/capture.pcap, link-type EN10MB (Ethernet)
08:41:13.521158 IP secOps.40966 > 172.16.0.40.http: Flags [S], seq 4212469753, win 29200, options [mss 1460,sackOK,TS val 2800064207 ecr 0,nop,wscale 9], length 0
08:41:13.521198 IP 172.16.0.40.http > secOps.40966: Flags [S.], seq 1079082162, ack 4212469754, win 28960, options [mss 1460,sackOK,TS val 1493403563 ecr 2800064207,nop,wscale 9], length 0
08:41:13.521211 IP secOps.40966 > 172.16.0.40.http: Flags [.], ack 1, win 58, options [nop,nop,TS val 2800064207 ecr 1493403563], length 0
[analyst@secOps ~]$
```

## 02 - EXPLORING NMAP

ho scansionato la macchina locale (localhost) utilizzando il comando **nmap -A -T4 localhost**, identificando porte aperte come **21/tcp** per FTP e **22/tcp** per **SSH**, insieme ai rispettivi software di servizio. Ho poi eseguito una scansione su un server remoto ([scanme.nmap.org](https://scanme.nmap.org)) per scoprire porte aperte, come 22/tcp per SSH e 80/tcp per HTTP, e servizi filtrati, ottenendo informazioni dettagliate sul sistema operativo (Ubuntu Linux) e sugli indirizzi IP del server.



```
Terminal - analyst@secOps
[analyst@secOps ~]$ nmap -A -T4 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2024-09-03 09:11 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00014s 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 3
|   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: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 11.95 seconds
[analyst@secOps ~]$

Terminal - analyst@secOps:~
[analyst@secOps ~]$ nmap -A -T4 scanme.nmap.org
Starting Nmap 7.70 ( https://nmap.org ) at 2024-09-03 09:14 EDT
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.18s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 996 closed 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    filtered 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 19.19 seconds
[analyst@secOps ~]$
```



## 03 - USING WIRESHARK TO EXAMINE TCP AND UDP CAPTURES

In questa prima parte del laboratorio, ho utilizzato Wireshark per catturare una **sessione FTP** e analizzare i campi dell'intestazione TCP. Dopo aver avviato la macchina virtuale CyberOps e Wireshark, ho stabilito una connessione con un server FTP esterno (**ftp.gnu.org**) e ho scaricato il file README.

```
File Edit View Terminal Tabs Help
[analyst@secOps ~]$ sudo /bin/wireshark-gtk
Gtk-Message: 10:34:35.430: GtkDialog mapped
```

```
ftp> get README
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for README (2748 bytes).
226 Transfer complete.
2748 bytes received in 5.2e-05 seconds (50.4 Mbytes/s)
ftp> █
```

```
[analyst@secOps ~]$ ftp ftp.gnu.org
Connected to ftp.gnu.org.
220 GNU FTP server ready.
Name (ftp.gnu.org:analyst): anonymous
230-NOTICE (Updated October 15 2021):
230-
230-If you maintain scripts used to access ftp.gnu.org over FTP,
230-we strongly encourage you to change them to use HTTPS instead.
230-
230-Eventually we hope to shut down FTP protocol access, but plan
230-to give notice here and other places for several months ahead
230-of time.
230-
230-
230-
230-Due to U.S. Export Regulations, all cryptographic software on this
230-site is subject to the following legal notice:
230-
230-   This site includes publicly available encryption source code
230-   which, together with object code resulting from the compiling of
230-   publicly available source code, may be exported from the United
230-   States under License Exception "TSU" pursuant to 15 C.F.R. Section
230-   740.13(e).
230-
230-This legal notice applies to cryptographic software only. Please see
230-the Bureau of Industry and Security (www.bxa.doc.gov) for more
230-information about current U.S. regulations.
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
lrwxrwxrwx  1 0      0          8 Aug 20   2004 CRYPTO.README -> .message
-rw-r--r--  1 0      0       17864 Oct 23   2003 MISSING-FILES
-rw-r--r--  1 0      0       4178 Aug 13   2003 MISSING-FILES.README
-rw-r--r--  1 0      0       2748 May 23   2023 README
-rw-r--r--  1 0      0      405121 Oct 23   2003 before-2003-08-01.md5sums.asc
-rw-rw-r--  1 0      3003     251693 Sep 02 13:30 find.txt.gz
drwxrwxr-x  324 0      3003     12288 Aug 05 10:50 gnu
drwxrwxr-x  3 0      3003      4096 Mar 10   2011 gnu+linux-distros
-rw-rw-r--  1 0      3003     485105 Sep 02 13:30 ls-lRt.txt.gz
drwxr-xr-x  3 0      0        4096 Apr 20   2005 mirrors
lrwxrwxrwx  1 0      0          11 Apr 15   2004 non-gnu -> gnu/non-gnu
drwxr-xr-x  99 0      0        4096 May 08   2023 old-gnu
lrwxrwxrwx  1 0      0           1 Aug 05   2003 pub -> .
drwxr-xr-x  2 0      0        4096 Nov 08   2007 savannah
drwxr-xr-x  2 0      0        4096 Aug 02   2003 third-party
drwxr-xr-x  2 0      0        4096 Apr 07   2009 tmp
-rw-rw-r--  1 0      3003     572332 Sep 02 13:30 tree.json.gz
```

# 03 - USING WIRESHARK TO EXAMINE TCP AND UDP CAPTURES

Durante la sessione FTP, Wireshark ha **catturato numerosi pacchetti**, che ho poi filtrato utilizzando l'indirizzo IP del server. Ho analizzato i primi tre pacchetti della stretta di mano a tre vie TCP, esaminando campi come l'indirizzo IP sorgente e di destinazione, i numeri di porta, i numeri di sequenza e di riconoscimento, nonché i bit di controllo come SYN e ACK. Questo mi ha permesso di comprendere meglio come TCP gestisce la consegna dei dati e la chiusura della sessione.

Filter: tcp and ip.addr == 192.168.1.74

Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
32	16.296338492	192.168.1.74	209.51.188.20	TCP	74	52926 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2255879597 TSecr=0 WS=128
33	16.417192057	209.51.188.20	192.168.1.74	TCP	74	21 → 52926 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2137565030 TSecr=2255879597 WS=128
34	16.417229639	192.168.1.74	209.51.188.20	TCP	66	52926 → 21 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2255879718 TSecr=2137565030
35	16.603967478	209.51.188.20	192.168.1.74	FTP	93	Response: 220 GNU FTP server ready.
36	16.604001696	192.168.1.74	209.51.188.20	TCP	66	52926 → 21 [ACK] Seq=1 Ack=28 Win=29312 Len=0 TSval=2255879905 TSecr=2137565217
37	19.176109439	192.168.1.74	209.51.188.20	FTP	82	Request: USER anonymous
38	19.295757187	209.51.188.20	192.168.1.74	TCP	66	21 → 52926 [ACK] Seq=28 Ack=17 Win=65280 Len=0 TSval=2137567909 TSecr=2255882477
39	19.341252729	209.51.188.20	192.168.1.74	FTP	105	Response: 230-NOTICE (Updated October 15 2021):
40	19.341273070	192.168.1.74	209.51.188.20	TCP	66	52926 → 21 [ACK] Seq=17 Ack=67 Win=29312 Len=0 TSval=2255882642 TSecr=2137567955
41	19.341376480	209.51.188.20	192.168.1.74	FTP	72	Response: 230-
42	19.341379703	192.168.1.74	209.51.188.20	TCP	66	52926 → 21 [ACK] Seq=17 Ack=73 Win=29312 Len=0 TSval=2255882642 TSecr=2137567955

▶ Frame 37: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0

▶ Ethernet II, Src: PcsCompu\_b8:0e:a7 (08:00:27:b8:0e:a7), Dst: D-LinkIn\_5c:89:24 (28:3b:82:5c:89:24)

▶ Internet Protocol Version 4, Src: 192.168.1.74, Dst: 209.51.188.20

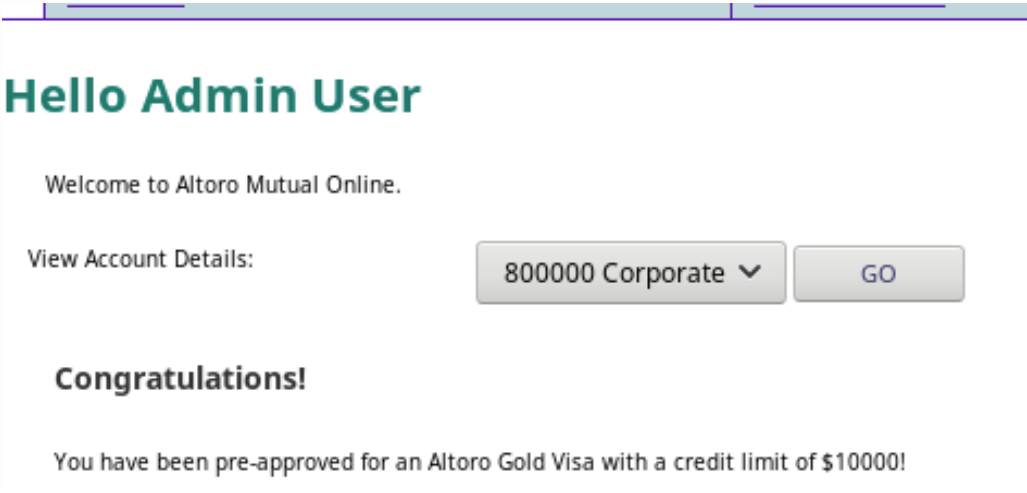
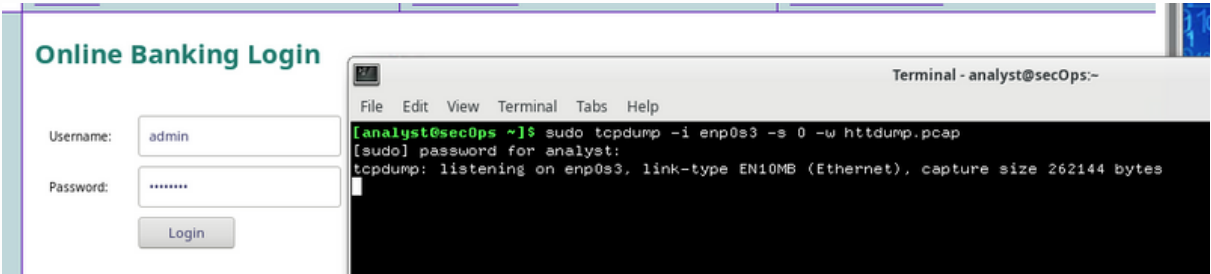
▶ Transmission Control Protocol, Src Port: 52926, Dst Port: 21, Seq: 1, Ack: 28, Len: 16

▼ File Transfer Protocol (FTP)

▼ USER anonymous\r\nRequest command: USERRequest arg: anonymous[Current working directory: ]

# 04 - USING WIRESHARK TO EXAMINE HTTP AND HTTPS TRAFFIC

Ho utilizzato **tcpdump** per catturare il traffico **HTTP** e analizzarlo successivamente con Wireshark. Dopo aver avviato la macchina virtuale CyberOps, ho aperto un terminale e avviato tcpdump sull'interfaccia di rete **enp0s3** per registrare il traffico in un file denominato **httdump.pcap**. Ho visitato il sito web **http://www.altoromutual.com/login.jsp** utilizzando HTTP, che non cifra il traffico, e ho effettuato un tentativo di accesso con credenziali di test. Successivamente, ho aperto il file di cattura in Wireshark, applicato un filtro per visualizzare solo il traffico HTTP, e ho esaminato i messaggi **POST**, osservando che le credenziali di accesso (**uid e passw**) erano visibili in chiaro.



562	49.173355	192.168.1.74	65.61.137.117	HTTP	601 POST /doLogin HTTP/1.1 (application/x-www-form-urlencoded)
▶ Frame 562: 601 bytes on wire (4808 bits), 601 bytes captured (4808 bits)					
▶ Ethernet II, Src: PcsCompu_b8:0e:a7 (08:00:27:b8:0e:a7), Dst: D-LinkIn_5c:89:24 (28:3b:82:5c:89:24)					
▶ Internet Protocol Version 4, Src: 192.168.1.74, Dst: 65.61.137.117					
▶ Transmission Control Protocol, Src Port: 37914, Dst Port: 80, Seq: 2030, Ack: 33574, Len: 535					
▶ Hypertext Transfer Protocol					
▼ HTML Form URL Encoded: application/x-www-form-urlencoded					
▶ Form item: "uid" = "admin"					
▶ Form item: "passw" = "admin"					
▶ Form item: "btnSubmit" = "Login"					

# 04 - USING WIRESHARK TO EXAMINE HTTP AND HTTPS TRAFFIC

Qui invece ho catturato il traffico **HTTPS** utilizzando nuovamente tcpdump e poi analizzato i dati con Wireshark. Dopo aver avviato tcpdump sull'interfaccia di rete, ho visitato il sito web **www.netacad.com**, che utilizza HTTPS, e ho effettuato un tentativo di accesso. Ho aperto il file di cattura in Wireshark e filtrato il traffico HTTPS, osservando che, a differenza del traffico HTTP, **i dati delle applicazioni erano cifrati e non leggibili in chiaro**. Questo dimostra come HTTPS utilizzi il protocollo TLS per garantire la riservatezza dei dati.

Filter: tcp.port==443

Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
3	0.001126	192.168.1.74	104.85.248.56	TLSv1.2	2003	Application Data
6	0.023741	104.85.248.56	192.168.1.74	TCP	66	443 → 55958 [ACK] Seq=1 Ack=1373 Win=501 Len=0 TSval=2769367994 TSecr=3088139423
7	0.024042	104.85.248.56	192.168.1.74	TCP	66	443 → 55958 [ACK] Seq=1 Ack=1938 Win=501 Len=0 TSval=2769367994 TSecr=3088139423
8	0.350869	104.85.248.56	192.168.1.74	TLSv1.2	2777	Application Data
9	0.350899	192.168.1.74	104.85.248.56	TCP	66	55958 → 443 [ACK] Seq=1938 Ack=2712 Win=7644 Len=0 TSval=3088139773 TSecr=2769368321

▶ Frame 8: 2777 bytes on wire (22216 bits), 2777 bytes captured (22216 bits)

▶ Ethernet II, Src: D-LinkIn\_5c:89:24 (28:3b:82:5c:89:24), Dst: PcsCompu\_b8:0e:a7 (08:00:27:b8:0e:a7)

▶ Internet Protocol Version 4, Src: 104.85.248.56, Dst: 192.168.1.74

▶ Transmission Control Protocol, Src Port: 443, Dst Port: 55958, Seq: 1, Ack: 1938, Len: 2711

Secure Sockets Layer

- ▶ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls

Secure Sockets Layer

TLSv1.2 Record Layer: Application Data Protocol: http-over-tls

- Content Type: Application Data (23)
- Version: TLS 1.2 (0x0303)
- Length: 2706
- Encrypted Application Data: 7a73cdd70bad2370740e925eca3d7c01dffb9e50e92ae865...



# GRAZIE

**Flavio Scognamiglio**