

1.-

Quan volem identificar l'establiment d'una connexió TCP a Wireshark, podem fer servir el filtre `tcp.flags.syn == 1 && tcp.flags.ack == 0`, que ens mostra els paquets inicials on s'envia el bit SYN sense l'ACK, indicant el començament de la connexió. Això correspon al primer pas del handshake de tres vies. SYN, SYN-ACK i ACK, podem aplicar el filtre `tcp.flags.syn == 1 || tcp.flags.ack == 1`, cosa que ens permet veure tots els paquets relacionats i comprendre com s'estableix la connexió.

```
▼ Transmission Control Protocol, Src Port: 23 (23), Dst Port: 52322 (52322), Seq: 21, Ack: 7, Len: 0
  Source Port: 23 (23)
  Destination Port: 52322 (52322)
  [Stream index: 3]
  [TCP Segment Len: 0]
  Sequence number: 21 (relative sequence number)
  Acknowledgment number: 7 (relative ack number)
  Header Length: 32 bytes
  ▼ ... 0000 0001 0001 = Flags: 0x011 (FIN, 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
    ▶ .... .... ...1 = Fin: Set
```

2.-

Una connexió TCP passa per diversos estats, començant amb l'establiment, continuant amb la transferència de dades i finalitzant el tancament. El procés s'inicia a l'estat CLOSED, on no hi ha connexió activa. Per establir la connexió, el client envia un paquet SYN i entra a SYN-SENT, mentre que el servidor respon amb un SYN-ACK, entrant a SYN-RECEIVED.

Quan el client confirma amb un ACK, tots dos costats passen a l'estat ESTABLISHED, que és on es fa la transferència de dades. Quan una de les parts decideix tancar la connexió, inicia el procés enviant un paquet FIN, entrant a FIN-WAIT-1. L'altra part respon amb un ACK, passant a CLOSE-WAIT, i eventualment envia el seu propi FIN. Això porta al primer costat a FIN-WAIT-2 i, en rebre l'ACK final, entra a TIME-WAIT per assegurar-se que no hi hagi paquets endarrerits. Finalment, ambdues parts arriben a l'estat CLOSED, completant el tancament de la connexió de manera segura.

3.-

Sí, el servidor pot acceptar connexions de múltiples clients simultàniament, i això s'aconsegueix gràcies a com el protocol TCP gestiona els sockets i les connexions a nivell de ports.

El servidor escolta en un port fix (el 23 per a Telnet) i roman en estat de "escolta" (LISTEN). Quan un client es connecta, el servidor accepta aquesta connexió i crea un socket únic per gestionar-la. Aquest socket únic està identificat per quatre valors: adreça IP i port del client, adreça IP i port del servidor.

Això permet que múltiples clients es connectin simultàniament al mateix port del servidor (23 per Telnet), ja que cada connexió té una combinació única d'adreça IP i port del client. El port del servidor continua constant, però els ports d'origen dels clients són diferents.

L'eina ss mostra aquesta relació en temps real. Si executem una ordre com ss -tna, podem observar:

```
Every 1,0s: ss -tan                                debian95-INTE-server: Mon Nov 18 11:24:25 2024
State     Recv-Q  Send-Q Local Address:Port      Peer Address:Port
LISTEN     0       10     192.168.88.237:53       *:*
LISTEN     0       10     127.0.0.1:53           *:*
LISTEN     0      128     *:22                   *:*
LISTEN     0        5     127.0.0.1:631          *:*
LISTEN     0      128     *:23                   *:*
LISTEN     0     100     *:25                   *:*
LISTEN     0      128     127.0.0.1:953         *:*
ESTAB      0        0     192.168.88.237:23      192.168.88.236:41638
ESTAB      0        0     192.168.88.237:23      192.168.88.236:52322
LISTEN     0      128     :::80                  :::*
LISTEN     0       10     :::53                  :::*
LISTEN     0       32     :::21                  :::*
LISTEN     0      128     :::22                  :::*
LISTEN     0        5     :::1:631               :::*
entel@debian95-INTE-server:~$ watch -n 1 ss -tan_      :::*
```

El port d'escolta del servidor per a Telnet (LISTEN 0 128 *:23).

```
LISTEN     0      128     *:23                   *:*
```

Les connexions actives on veurem múltiples entrades amb el port del servidor fix (23), però diferents ports d'origen per a cada client.

```
ESTAB      0        0     192.168.88.237:23      192.168.88.236:41638
ESTAB      0        0     192.168.88.237:23      192.168.88.236:52322
```

Aquí, el servidor (192.168.88.237:23) té dues connexions establertes amb clients a diferents ports d'origen (41638, 52322). Podem veure com TCP utilitza diferents combinacions de ports i adreces per gestionar múltiples connexions simultànies sobre un mateix port de servidor.

4.-

Les comandes s'envien en text clar des del client cap a la consola del servidor. Això significa que les dades, incloses les credencials d'inici de sessió i les ordres executades, es transmeten sense xifrar a través de la xarxa.

```
entel@debian95-INTE-server:~$ hola
-bash: hola: no s'ha trobat l'ordre
entel@debian95-INTE-server:~$
```

```
▶ Frame 107: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface 0
▶ Ethernet II, Src: CadmusCo_99:2c:e5 (08:00:27:99:2c:e5), Dst: 6c:4b:90:c1:68:68 (6c:4b:90:c1:68:68)
▶ Internet Protocol Version 4, Src: 192.168.88.237 (192.168.88.237), Dst: 192.168.88.236 (192.168.88.236)
▼ Transmission Control Protocol, Src Port: 23 (23), Dst Port: 52322 (52322), Seq: 177, Ack: 26, Len: 1
    Source Port: 23 (23)
    Destination Port: 52322 (52322)
    [Stream index: 1]
    [TCP Segment Len: 1]
    Sequence number: 177 (relative sequence number)
    [Next sequence number: 178 (relative sequence number)]
    Acknowledgment number: 26 (relative ack number)
    Header Length: 32 bytes
    ▶ ... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
    Window size value: 905
    [Calculated window size: 905]
    [Window size scaling factor: -1 (unknown)]
    ▶ Checksum: 0xd6e2 [validation disabled]
    Urgent pointer: 0
    ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▶ [SEQ/ACK analysis]
▼ Telnet
    Data: h
```

7.-

FTP utilitza dues connexions TCP: una per al control i una altra per a la transferència de dades. La connexió de control s'estableix al port 21 i es manté sempre activa durant tota la sessió, ja que s'encarrega de gestionar les ordres entre el client i el servidor. En canvi, la connexió de dades s'utilitza només quan cal transferir fitxers o directoris. Depenent del mode (actiu o passiu), el client o el servidor estableix aquesta connexió en un port diferent. sempre activa, ja que només s'obre quan cal transferir informació i es tanca després de completar la transferència.

```
Every 1,0s: ss -tan          debian95-INTE-server: Mon Nov 18 11:42:00 2024
```

State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
LISTEN	0	10	192.168.88.237:53	*:*
LISTEN	0	10	127.0.0.1:53	*:*
LISTEN	0	128	*:22	*:*
LISTEN	0	5	127.0.0.1:631	*:*
LISTEN	0	128	*:23	*:*
LISTEN	0	100	*:25	*:*
LISTEN	0	128	127.0.0.1:953	*:*
LISTEN	0	128	:::80	:::*
LISTEN	0	32	:::21	:::*
LISTEN	0	10	:::53	:::*
LISTEN	0	128	:::22	:::*
LISTEN	0	5	:::1:631	:::*
LISTEN	0	100	:::25	:::*
LISTEN	0	128	:::1:953	:::*
TIME-WAIT	0	0	::ffff:192.168.88.237:20	::ffff:192.168.88.236:60991
ESTAB	0	0	::ffff:192.168.88.237:21	::ffff:192.168.88.236:60991

8.-

A FTP, la responsabilitat d'establir la connexió de dades varia segons el mode d'operació. En mode actiu, el servidor és qui estableix la connexió de dades, ja que el client us indica el port on esteu escoltant i el servidor es connecta a aquest port des del seu port 20. En mode passiu, la situació canvia: el servidor obre un port aleatori i comunica al client a quin port s'ha de connectar i el client inicia la connexió de dades cap al port especificat pel servidor.

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
226 Directory send OK.
```

A la imatge anterior podem observar que el mètode d'operació és mode passiu.

9.-

Sí, es poden fer diverses peticions HTTP en una mateixa connexió TCP. Amb HTTP/1.1, gràcies a Keep-Alive, la connexió es manté oberta per fer múltiples sol·licituds consecutives. HTTP/2 permet fer múltiples peticions simultànies amb multiplexing, millorant la velocitat i reduint la latència. Això redueix la sobrecàrrega, millora l'eficiència i disminueix la latència, sempre que client i servidor ho suportin.

10.-

Es fa servir la versió HTTP/1.1

```
▶ Frame 36: 555 bytes on wire (4440 bits), 555 bytes captured (4440 bits) on interface 0
▶ Ethernet II, Src: 6c:4b:90:c1:68:68 (6c:4b:90:c1:68:68), Dst: CadmusCo_99:2c:e5 (08:00:27:99:2c:e5)
▶ Internet Protocol Version 4, Src: 192.168.88.236 (192.168.88.236), Dst: 192.168.88.237 (192.168.88.237)
▶ Transmission Control Protocol, Src Port: 49868 (49868), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 489
▼ Hypertext Transfer Protocol
  ▼ GET / HTTP/1.1\r\n
    ▶ [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /
      Request Version: HTTP/1.1
      Host: 192.168.88.237\r\n
      User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:130.0) Gecko/20100101 Firefox/130.0\r\n
      Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8\r\n
      Accept-Language: ca,en-US;q=0.7,en;q=0.3\r\n
      Accept-Encoding: gzip, deflate\r\n
      Connection: keep-alive\r\n
      Upgrade-Insecure-Requests: 1\r\n
      If-Modified-Since: Tue, 06 Nov 2018 12:08:59 GMT\r\n
      If-None-Match: "29cd-579fddda245b-gzip"\r\n
      Priority: u=0, i\r\n
      \r\n
      [Full request URI: http://192.168.88.237/]
      [HTTP request 1/1]
      [Response in frame: 40]
```

Els mètodes utilitzats són:

-GET: Per sol·licitar recursos.

-POST: Per enviar dades.

Existeixen altres mètodes menys utilitzats, com PUT, DELETE, HEAD o OPTIONS.

Els codis de resposta HTTP més comuns que pots observar són el 200 OK i el 204 No Content per a èxits, el 301 Moved Permanently, 302 Found i 304 Not Modified per a redireccions, el 400 Bad Request i 404 Not Found per a errors del client, i el 500 Internal Server Error i 503 Service Unavailable per a errors del servidor.

```
▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 200 OK\r\n
    ▶ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Request Version: HTTP/1.1
      Status Code: 200
      Response Phrase: OK
      Date: Mon, 18 Nov 2024 10:48:05 GMT\r\n
```

11.-

```
Line-based text data: text/html
\n
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">\n
<html xmlns="http://www.w3.org/1999/xhtml">\n
  <head>\n
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />\n
    <title>Apache2 Debian Default Page: It works</title>\n
    <style type="text/css" media="screen">\n
      * {\n
        margin: 0px 0px 0px 0px;\n
        padding: 0px 0px 0px 0px;\n
      }\n
\n
      body, html {\n
        padding: 3px 3px 3px 3px;\n
\n
        background-color: #D8DBE2;\n
\n
        font-family: Verdana, sans-serif;\n
        font-size: 11pt;\n
        text-align: center;\n
\n
      }\n
\n
0000 6c 4b 90 c1 68 68 08 00 27 99 2c e5 08 00 45 00 |LK..hh.. '.,...E.
0010 07 c0 99 eb 40 00 40 06 66 22 c0 a8 58 ed c0 a8 |...@.@. f"..X...
0020 58 ec 00 50 c2 cc d5 ea 26 1c 3d 45 9a 57 80 18 |X..P.... &.=E.W...

Frame (1998 bytes)   Reassembled TCP (3380 bytes)   Uncompressed entity body (10701 bytes)
```

La pàgina per defecte del servidor Apache, esta composta per alguns recursos bàsics com HTML, imatges, fulls d'estil (CSS), links...

12.-

Quan es demana un recurs que no existeix al servidor, aquest retorna un error 404 Not Found. Això vol dir que el servidor ha processat la sol·licitud, però no ha trobat el fitxer o la pàgina que s'ha demanat.

```
▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 404 Not Found\r\n
    ▼ [Expert Info (Chat/Sequence): HTTP/1.1 404 Not Found\r\n\r\n]
      [HTTP/1.1 404 Not Found\r\n\r\n]
      [Severity level: Chat]
      [Group: Sequence]
      Request Version: HTTP/1.1
      Status Code: 404
      Response Phrase: Not Found
```

13.-

Quan el servidor web d'Atenea ens passa d'una connexió HTTP a una HTTPS, utilitza el codi d'estat 302, conegut com "Moved Temporarily". Aquest codi vol dir que el recurs que hem demanat es troba temporalment en una altra adreça, que es detalla a l'encapçalament Location. Això passa quan volem accedir a una pàgina http, i ens redirigeix a la https.

```
▶ Frame 7133: 171 bytes on wire (1368 bits), 171 bytes captured (1368 bits) on interface 0
▶ Ethernet II, Src: d4:01:c3:5a:1d:08 (d4:01:c3:5a:1d:08), Dst: 6c:4b:90:c1:68:68 (6c:4b:90:c1:68:68)
▶ Internet Protocol Version 4, Src: 147.83.2.135 (147.83.2.135), Dst: 192.168.88.236 (192.168.88.236)
▶ Transmission Control Protocol, Src Port: 80 (80), Dst Port: 43780 (43780), Seq: 1, Ack: 135, Len: 105
▼ Hypertext Transfer Protocol
  ▶ HTTP/1.1 302 Moved Temporarily\r\n
    Location: https://upc.edu/\r\n
    Connection: Keep-Alive\r\n
  ▶ Content-Length: 0\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.001825000 seconds]
  [Request in frame: 7131]
```

14.-

Per veure que la connexió és segura a Wireshark, cal filtrar per tls o ssl. A continuació, es pot observar el procés de negociació TLS, on apareixen els missatges ClientHello i ServerHello, que indiquen l'establiment de la connexió segura. Un cop establerta la connexió, el trànsit entre el client i el servidor es xifra, i els paquets mostraran "Application Data". A més, podem veure el certificat SSL del servidor i la llista de ciphersuites que s'utilitzaran per al xifratge de la comunicació.

131	8.034712364	192.168.1.221	185.199.109.154	TLSv1.3	1863 Client Hello
135	8.044689529	192.168.1.221	140.82.121.4	TLSv1.3	1818 Client Hello
138	8.048324642	185.199.109.154	192.168.1.221	TLSv1.3	1446 Server Hello, Change Cipher Spec, Application Data
142	8.048324746	185.199.109.154	192.168.1.221	TLSv1.3	379 Application Data, Application Data, Application Data
145	8.048402427	185.199.109.154	192.168.1.221	TLSv1.3	1446 [TCP Fast Retransmission], Application Data
152	8.051312905	192.168.1.221	185.199.109.154	TLSv1.3	130 Change Cipher Spec, Application Data
160	8.058609340	192.168.1.221	142.251.168.91	TLSv1.3	2294 Client Hello
174	8.070850978	140.82.121.4	192.168.1.221	TLSv1.3	2914 Server Hello, Change Cipher Spec, Application Data
176	8.070851060	140.82.121.4	192.168.1.221	TLSv1.3	711 Application Data, Application Data, Application Data
178	8.073254600	192.168.1.221	140.82.121.4	TLSv1.3	130 Change Cipher Spec, Application Data
179	8.073344303	192.168.1.221	140.82.121.4	TLSv1.3	158 Application Data
180	8.073424267	192.168.1.221	140.82.121.4	TLSv1.3	623 Application Data
184	8.090558755	142.251.168.91	192.168.1.221	TLSv1.3	1538 Server Hello, Change Cipher Spec, Application Data
187	8.092119989	192.168.1.221	142.251.168.91	TLSv1.3	140 Change Cipher Spec, Application Data