

Instalar Wireshark y observar cómo fluye el tráfico de red en el balanceador de la máquina m3 mientras se le hacen peticiones HTTP y HTTPS. Ejecuta al menos 3 peticiones al balanceador.

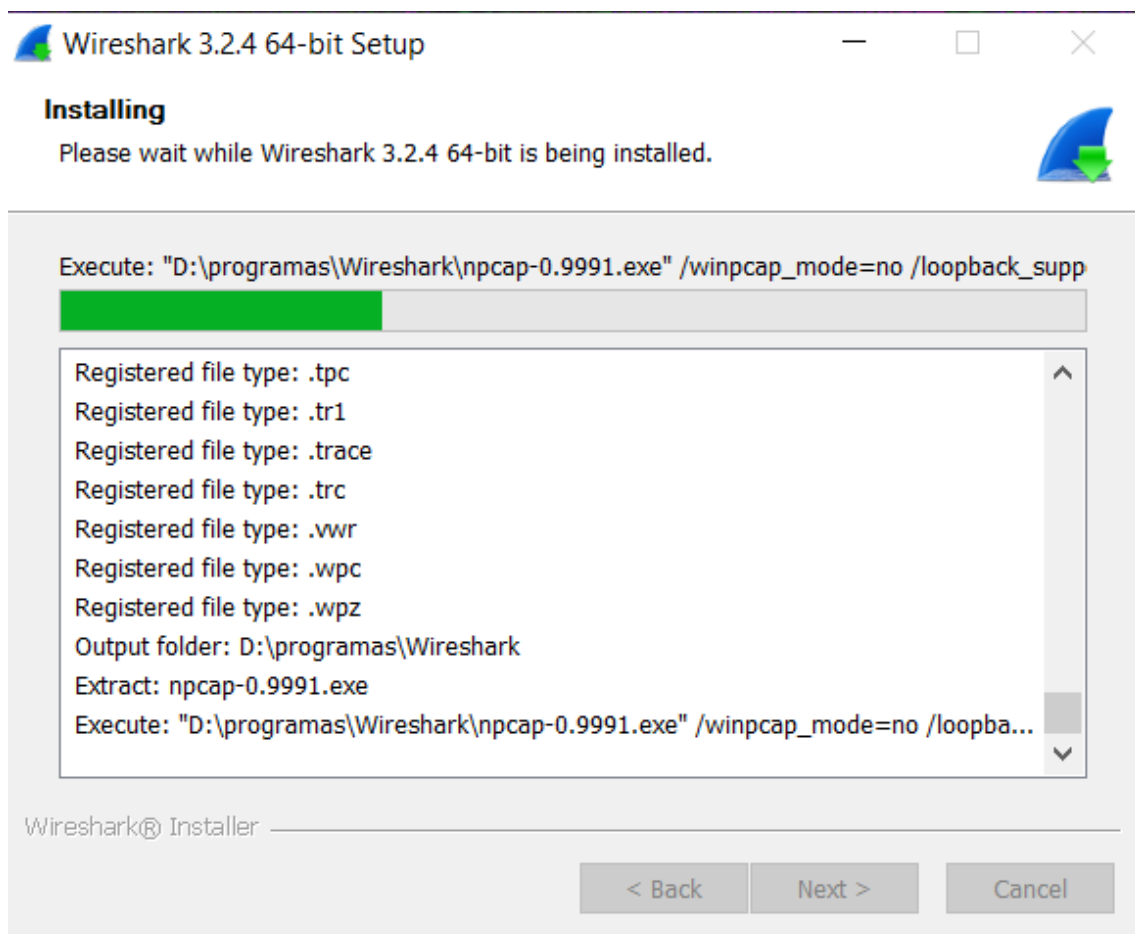
Realiza un análisis de una sesión TCP (establecer conexión y cierre) de peticiones HTTP y HTTPS y escribe tus propias conclusiones. Puedes ilustrarlo con capturas de pantalla.

Wireshark es uno de los analizadores de protocolos de red más importantes a nivel mundial. Es el estándar en muchas empresas comerciales, gubernamentales y organizaciones. Su software libre y su disponibilidad en multitud de sistemas operativos, enriquece a su característica más atractiva, su interfaz gráfica.

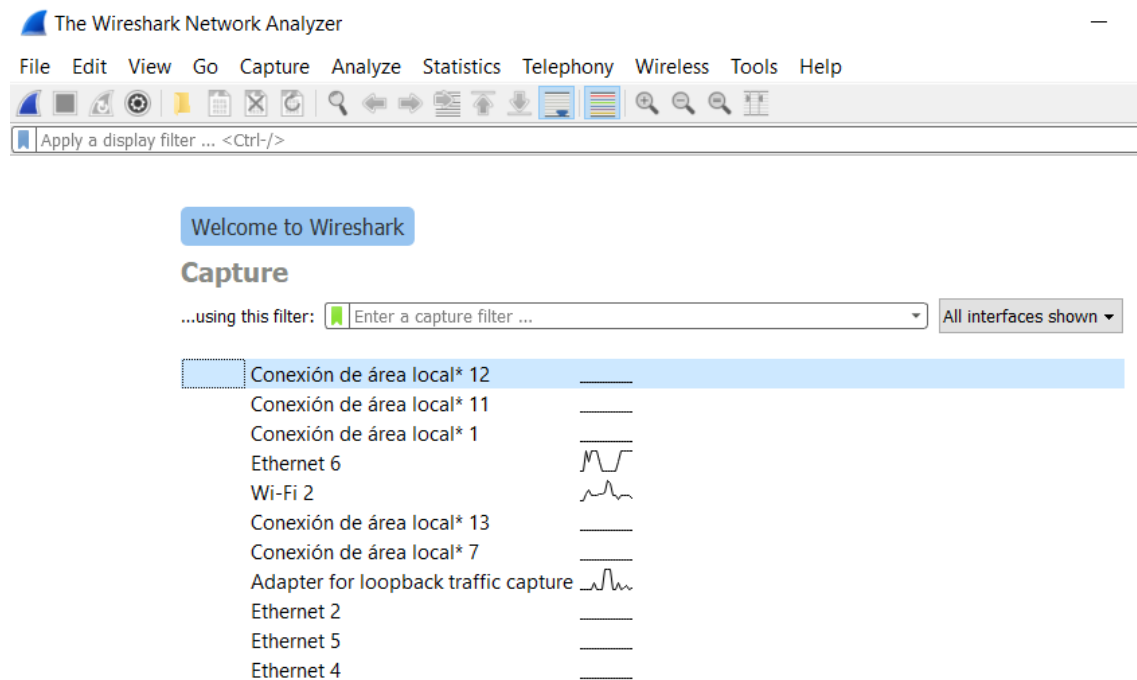
Permite examinar datos de una red en uso o de un archivo de captura guardado en disco, pudiendo analizar la información capturada, a través de los detalles de cada paquete.

Incluye un lenguaje completo para filtrar lo que necesitemos ver y la capacidad de mostrar el flujo reconstruido de una sesión TCP.

Primero procedemos a la instalación de Wireshark en la máquina anfitrión (con Windows en este caso). Descargamos la versión de 64 bits desde el siguiente enlace y comenzamos la instalación: <https://www.wireshark.org/download.html>



Debemos reiniciar el sistema al acabar la instalación. Una vez instalado, debemos ejecutar Wireshark como administrador.



Podemos ver las conexiones de la máquina en la que se ejecuta y un pequeño gráfico informativo acerca de su uso.

Seleccionamos la que queramos analizar (en este caso Ethernet 6) y realizamos desde el *cmd* de Windows los *curl* siguientes, tres veces cada uno:

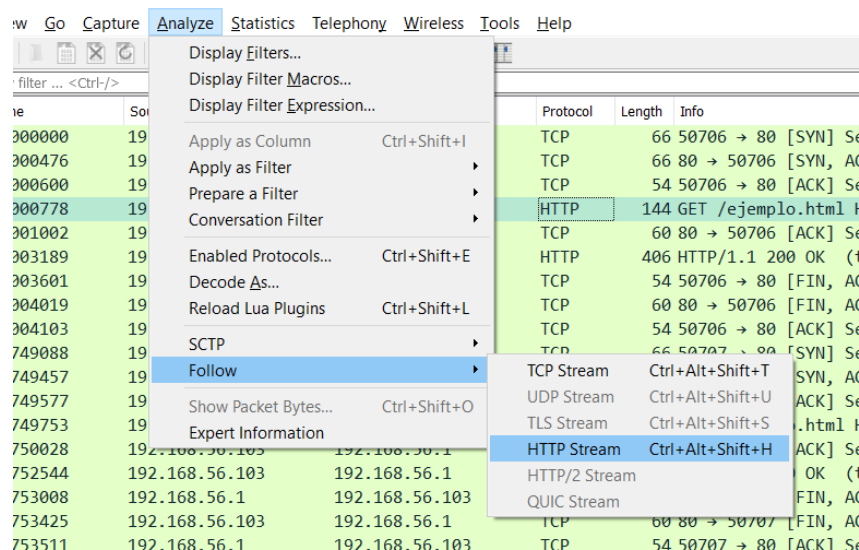
curl 192.168.56.103/ejemplo

Peticiones HTTP al balanceador

Time	Source	Destination	Protocol	Length	Info
1 0.000000	192.168.56.1	192.168.56.103	TCP	66	50706 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2 0.000476	192.168.56.103	192.168.56.1	TCP	66	80 → 50706 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
3 0.000600	192.168.56.1	192.168.56.103	TCP	54	50706 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
4 0.000778	192.168.56.1	192.168.56.103	HTTP	144	GET /ejemplo.html HTTP/1.1
5 0.001002	192.168.56.103	192.168.56.1	TCP	60	80 → 50706 [ACK] Seq=1 Ack=91 Win=64256 Len=0
6 0.003189	192.168.56.103	192.168.56.1	HTTP	406	HTTP/1.1 200 OK (text/html)
7 0.003601	192.168.56.1	192.168.56.103	TCP	54	50706 → 80 [FIN, ACK] Seq=91 Ack=353 Win=2102016 Len=0
8 0.004019	192.168.56.103	192.168.56.1	TCP	60	80 → 50706 [FIN, ACK] Seq=353 Ack=92 Win=64256 Len=0
9 0.004103	192.168.56.1	192.168.56.103	TCP	54	50706 → 80 [ACK] Seq=92 Ack=354 Win=2102016 Len=0
10 0.749088	192.168.56.1	192.168.56.103	TCP	66	50707 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
11 0.749457	192.168.56.103	192.168.56.1	TCP	66	80 → 50707 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
12 0.749577	192.168.56.1	192.168.56.103	TCP	54	50707 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
13 0.749753	192.168.56.1	192.168.56.103	HTTP	144	GET /ejemplo.html HTTP/1.1
14 0.750028	192.168.56.103	192.168.56.1	TCP	60	80 → 50707 [ACK] Seq=1 Ack=91 Win=64256 Len=0
15 0.752544	192.168.56.103	192.168.56.1	HTTP	406	HTTP/1.1 200 OK (text/html)
16 0.753008	192.168.56.1	192.168.56.103	TCP	54	50707 → 80 [FIN, ACK] Seq=91 Ack=353 Win=2102016 Len=0
17 0.753425	192.168.56.103	192.168.56.1	TCP	60	80 → 50707 [FIN, ACK] Seq=353 Ack=92 Win=64256 Len=0
18 0.753511	192.168.56.1	192.168.56.103	TCP	54	50707 → 80 [ACK] Seq=92 Ack=354 Win=2102016 Len=0
19 1.494525	192.168.56.1	192.168.56.103	TCP	66	50708 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
20 1.494878	192.168.56.103	192.168.56.1	TCP	66	80 → 50708 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
21 1.495006	192.168.56.1	192.168.56.103	TCP	54	50708 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
22 1.495268	192.168.56.1	192.168.56.103	HTTP	144	GET /ejemplo.html HTTP/1.1
23 1.495616	192.168.56.103	192.168.56.1	TCP	60	80 → 50708 [ACK] Seq=1 Ack=91 Win=64256 Len=0
24 1.498166	192.168.56.103	192.168.56.1	HTTP	406	HTTP/1.1 200 OK (text/html)
25 1.498643	192.168.56.1	192.168.56.103	TCP	54	50708 → 80 [FIN, ACK] Seq=91 Ack=353 Win=2102016 Len=0
26 1.498984	192.168.56.103	192.168.56.1	TCP	60	80 → 50708 [FIN, ACK] Seq=353 Ack=92 Win=64256 Len=0
27 1.499087	192.168.56.1	192.168.56.103	TCP	54	50708 → 80 [ACK] Seq=92 Ack=354 Win=2102016 Len=0

Time: 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{5F69FFAC-08-00-00-00-00} Ethernet II, Src: 0a:00:27:00:00:0f (0a:00:27:00:00:0f), Dst: PcsCompu_4d:8e:58 (08:00:27:4d:8e:58) Ethernet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.103 Transmission Control Protocol, Src Port: 50706, Dst Port: 80, Seq: 0, Len: 0

Una herramienta interesante está en la barra */Analyze/Follow/HTTP Stream* que nos facilitará ver dónde empieza y termina una sesión (en este caso la primera). Pulsamos en la primera petición HTTP previamente y seleccionamos la siguiente opción para analizar algunos de sus datos:



Como podemos comprobar podemos acceder a toda la información del GET fácilmente a través del protocolo HTTP.

SourceDestinationProtocolLengthInfo

192.168.56.1192.168.56.103TCP6650706 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK

192.168.56.103192.168.56.1TCP6680 → 50706 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 S

192.168.56.1192.168.56.103TCP5450706 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0

192.168.56.1192.168.56.103HTTP144GET /ejemplo.html HTTP/1.1

192.168.56.103192.168.56.1TCP6080 → 50706 [ACK] Seq=1 Ack=91 Win=64256 Len=0

192.168.56.103192.168.56.1HTTP406HTTP/1.1 200 OK (text/html)

192.168.56.1192.168.56.103TCP5450706 → 80 [FIN, ACK] Seq=91 Ack=353 Win=2102016 Len=0

192.168.56.103192.168.56.1TCP6080 → 50706 [FIN, ACK] Seq=353 Ack=92 Win=64256 Len=0

192.168.56.1192.168.56.103TCP5450706 → 80 [ACK] Seq=92 Ack=354 Win=2102016 Len=0

GET /ejemplo.html HTTP/1.1

Host: 192.168.56.103

User-Agent: curl/7.55.1

Accept: */*

HTTP/1.1 200 OK

Server: nginx/1.14.0 (Ubuntu)

Date: Thu, 28 May 2020 20:12:00 GMT

Content-Type: text/html

Content-Length: 79

Connection: keep-alive

Last-Modified: Thu, 12 Mar 2020 15:46:35 GMT

ETag: "4f-5a0aa41f61d8d"

Accept-Ranges: bytes

Vary: Accept-Encoding

<html>

<body>

Web de ejemplo de joselepdraza para SWAP(m1)

</body>

</html>

1 client pkt(s), 1 server pkt(s), 1 turn(s).

Entire conversation (442 bytes)

Show and save data as A

Find:

Filter Out This Stream

Print

Save as...

Back

Close

curl -k <https://192.168.56.103/ejemplo.html>

Peticiones HTTPS al balanceador

No.TimeSourceDestinationProtocolLengthInfo

10.000000192.168.56.1192.168.56.103TCP6650735 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

20.000364192.168.56.103192.168.56.1TCP66443 → 50735 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128

30.000519192.168.56.1192.168.56.103TCP5450735 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0

40.005111192.168.56.1192.168.56.103TLSv1.2201Client Hello

50.005442192.168.56.103192.168.56.1TCP60443 → 50735 [ACK] Seq=1 Ack=148 Win=64128 Len=0

60.007707192.168.56.103192.168.56.1TLSv1.21482Server Hello, Certificate, Server Key Exchange, Server Hello Done

70.009932192.168.56.1192.168.56.103TLSv1.2147Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

80.010760192.168.56.103192.168.56.1TLSv1.2296New Session Ticket, Change Cipher Spec, Encrypted Handshake Message

90.015362192.168.56.1192.168.56.103TLSv1.2173Application Data

100.018583192.168.56.103192.168.56.1TLSv1.2435Application Data

110.019548192.168.56.1192.168.56.103TLSv1.285Encrypted Alert

120.019746192.168.56.1192.168.56.103TCP5450735 → 443 [FIN, ACK] Seq=391 Ack=2052 Win=2102016 Len=0

130.020026192.168.56.103192.168.56.1TCP60443 → 50735 [FIN, ACK] Seq=2052 Ack=392 Win=64128 Len=0

140.020104192.168.56.1192.168.56.103TCP5450735 → 443 [ACK] Seq=392 Ack=2053 Win=2102016 Len=0

152.075676192.168.56.1192.168.56.103TCP6650736 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

162.076081192.168.56.103192.168.56.1TCP66443 → 50736 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128

172.076204192.168.56.1192.168.56.103TCP5450736 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0

182.080822192.168.56.1192.168.56.103TLSv1.2201Client Hello

192.081257192.168.56.103192.168.56.1TCP60443 → 50735 [ACK] Seq=1 Ack=148 Win=64128 Len=0

202.083355192.168.56.103192.168.56.1TLSv1.21482Server Hello, Certificate, Server Key Exchange, Server Hello Done

212.085297192.168.56.1192.168.56.103TLSv1.2147Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

222.086178192.168.56.103192.168.56.1TLSv1.2296New Session Ticket, Change Cipher Spec, Encrypted Handshake Message

232.090386192.168.56.1192.168.56.103TLSv1.2173Application Data

242.093014192.168.56.103192.168.56.1TLSv1.2435Application Data

252.093964192.168.56.1192.168.56.103TLSv1.285Encrypted Alert

262.094160192.168.56.1192.168.56.103TCP5450735 → 443 [FIN, ACK] Seq=391 Ack=2052 Win=2102016 Len=0

272.094437192.168.56.103192.168.56.1TCP60443 → 50735 [FIN, ACK] Seq=2052 Ack=392 Win=64128 Len=0

282.094547192.168.56.1192.168.56.103TCP5450735 → 443 [ACK] Seq=392 Ack=2053 Win=2102016 Len=0

293.671391192.168.56.1192.168.56.103TCP6650736 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

303.671766192.168.56.103192.168.56.1TCP66443 → 50736 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128

313.671888192.168.56.1192.168.56.103TCP5450736 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0

323.676444192.168.56.1192.168.56.103TLSv1.2201Client Hello

333.676825192.168.56.103192.168.56.1TCP60443 → 50735 [ACK] Seq=1 Ack=148 Win=64128 Len=0

343.679062192.168.56.103192.168.56.1TLSv1.21482Server Hello, Certificate, Server Key Exchange, Server Hello Done

353.681166192.168.56.1192.168.56.103TLSv1.2147Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

363.682147192.168.56.103192.168.56.1TLSv1.2296New Session Ticket, Change Cipher Spec, Encrypted Handshake Message

373.686407192.168.56.1192.168.56.103TLSv1.2173Application Data

383.689094192.168.56.103192.168.56.1TLSv1.2435Application Data

393.690000192.168.56.1192.168.56.103TLSv1.285Encrypted Alert

403.690199192.168.56.1192.168.56.103TCP5450735 → 443 [FIN, ACK] Seq=391 Ack=2052 Win=2102016 Len=0

413.690529192.168.56.103192.168.56.1TCP60443 → 50735 [FIN, ACK] Seq=2052 Ack=392 Win=64128 Len=0

423.690626192.168.56.1192.168.56.103TCP5450735 → 443 [ACK] Seq=392 Ack=2053 Win=2102016 Len=0

437.716626192.168.56.1192.168.56.255NBNS

448.466423192.168.56.1192.168.56.255NBNS

<

> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{5F6...}

Símbolo del sistema

C:\Users\Josele>curl -k https://192.168.56.103/ejemplo.html

<html>

<body>

Web de ejemplo de joselepdraza para SWAP(m2)

</body>

</html>

C:\Users\Josele>curl -k https://192.168.56.103/ejemplo.html

<html>

<body>

Web de ejemplo de joselepdraza para SWAP(m1)

</body>

</html>

C:\Users\Josele>curl -k https://192.168.56.103/ejemplo.html

<html>

<body>

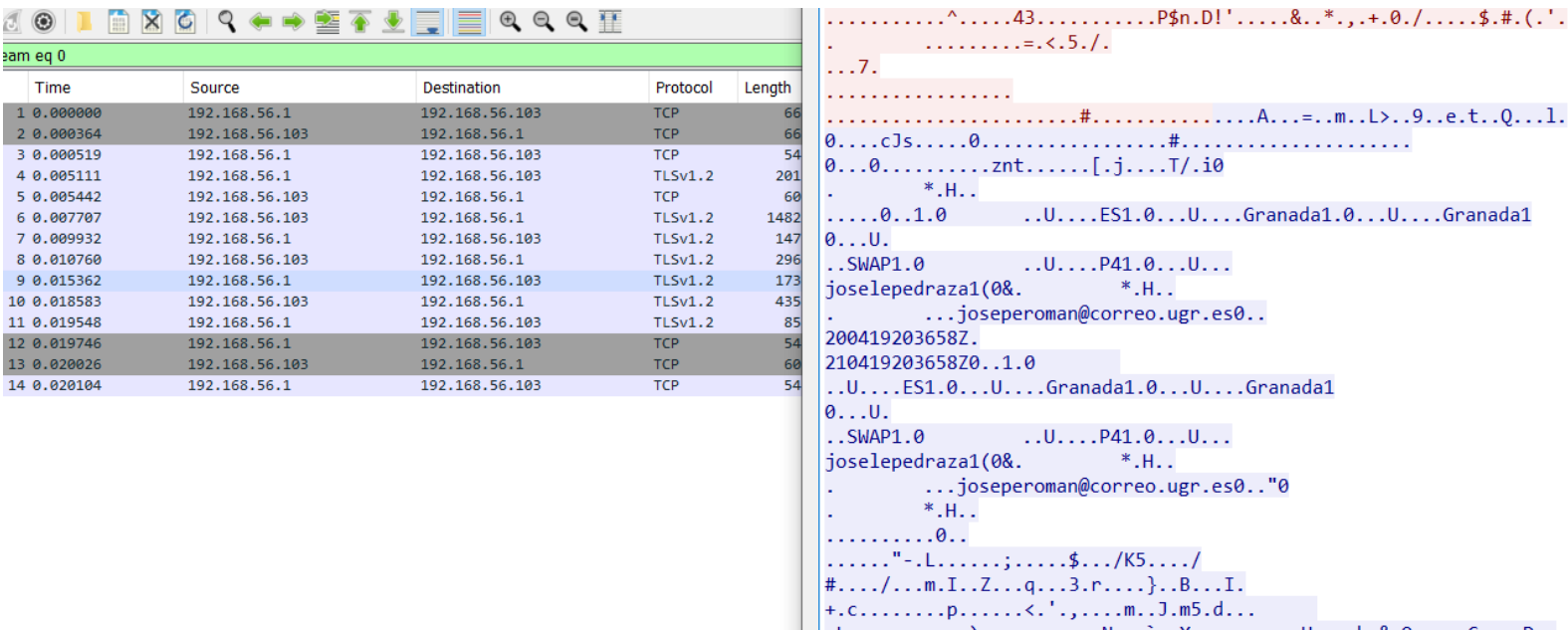
Web de ejemplo de joselepdraza para SWAP(m2)

</body>

</html>

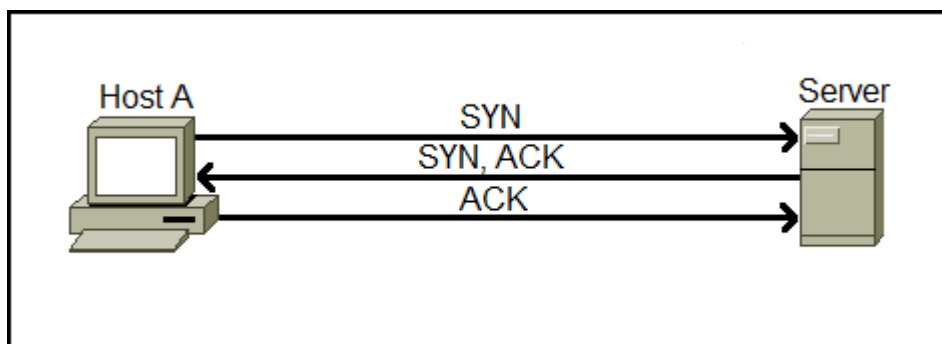
C:\Users\Josele>

Si volvemos a la herramienta de la barra */Analyze/Follow/HTTP Stream* pero pulsamos en *TCP stream* en el primer paquete TCP de la primera sesión podremos ver información encriptada y la información del certificado (SSL autofirmado en este caso) que lo firma.



Time	Source	Destination	Protocol	Length
1 0.000000	192.168.56.1	192.168.56.103	TCP	66
2 0.000364	192.168.56.103	192.168.56.1	TCP	66
3 0.000519	192.168.56.1	192.168.56.103	TCP	54
4 0.005111	192.168.56.1	192.168.56.103	TLSv1.2	201
5 0.005442	192.168.56.103	192.168.56.1	TCP	60
6 0.007707	192.168.56.103	192.168.56.1	TLSv1.2	1482
7 0.009932	192.168.56.1	192.168.56.103	TLSv1.2	147
8 0.010760	192.168.56.103	192.168.56.1	TLSv1.2	296
9 0.015362	192.168.56.1	192.168.56.103	TLSv1.2	173
10 0.018583	192.168.56.103	192.168.56.1	TLSv1.2	435
11 0.019548	192.168.56.1	192.168.56.103	TLSv1.2	85
12 0.019746	192.168.56.1	192.168.56.103	TCP	54
13 0.020026	192.168.56.103	192.168.56.1	TCP	60
14 0.020104	192.168.56.1	192.168.56.103	TCP	54

No nos será posible comprobar el contenido de lo que sirve las máquinas del backend gracias al certificado SSL. También podemos observar “triple handshake” del protocolo TCP.



Como hemos comprobado es una herramienta muy potente para este propósito. Hemos visto menos del 1% de la herramienta, también hay cosas muy interesantes en *Statistics* (donde podremos ver el tráfico desde diferentes perspectivas como “flujos de entrada/salida, o “diagramas de flujo” y una serie de estadísticas; también en *Tools* podremos configurar el cortafuegos de Wireshark, y una infinidad de opciones que nos brinda esta herramienta.