

RETO 7

TÚNELES Y REDES PRIVADAS VIRTUALES

Pablo Ramiro Foronda

Marianela Estévez Bosso

Ignacio García García

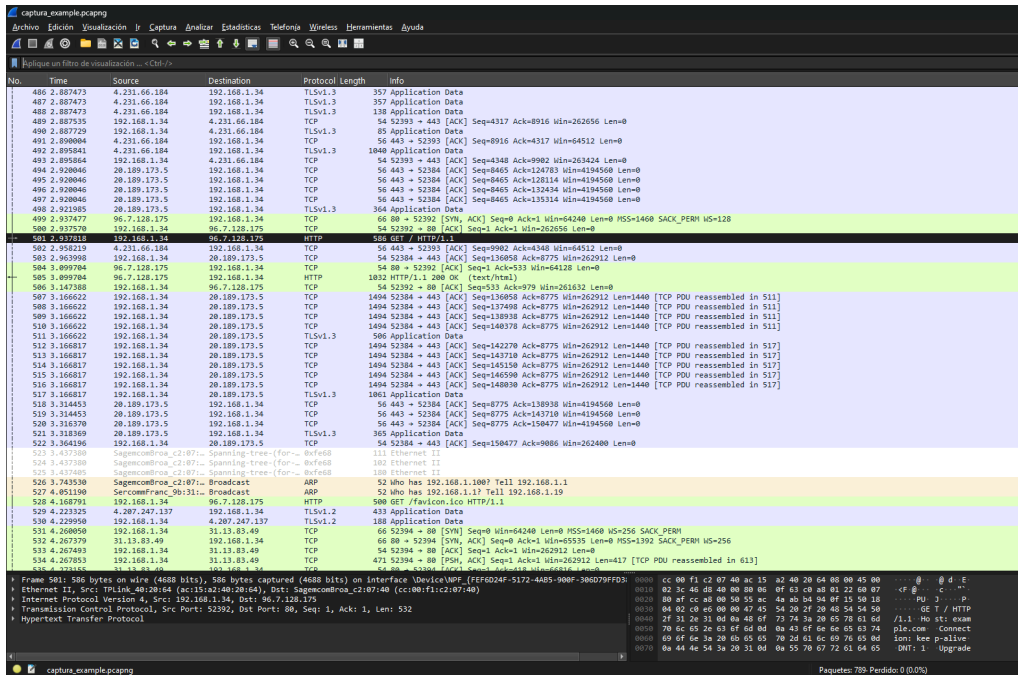
Guillermo Rojo Martín

Realización de capturas de tráfico:

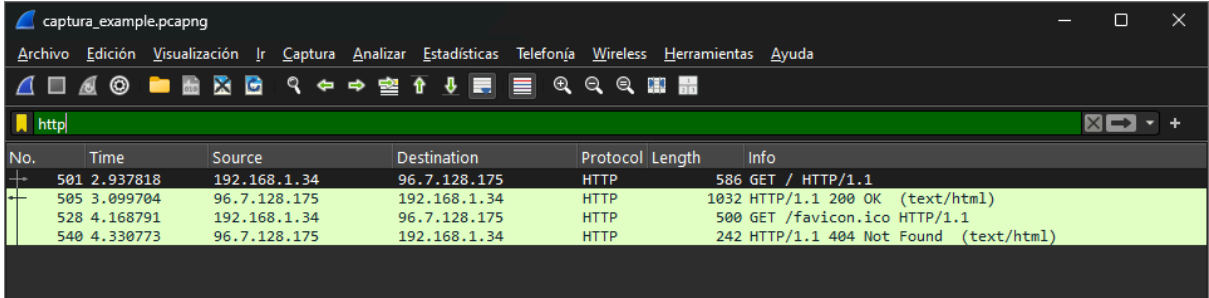
1. Primer escenario: Navegación web tradicional

- HTTP. Ejemplo: <http://example.com/>

La dirección IP de nuestro pc es 192.168.1.34 y la de la página es 96.7.128.198. Es una IP diferente pero muy cercana en el rango a la que capturó Wireshark (96.7.128.175). Esto es normal para sitios que pueden usar múltiples IPs o balanceadores de carga. Lo importante es a qué IP se conectó mi navegador en el momento de la captura, y Wireshark nos lo dice.



No.	Time	Source	Destination	Protocol	Length	Info
486	2.887473	4.231.66.184	192.168.1.34	TLsv1.3	357	Application Data
487	2.887473	4.231.66.184	192.168.1.34	TLsv1.3	357	Application Data
488	2.887473	4.231.66.184	192.168.1.34	TLsv1.3	138	Application Data
489	2.887535	192.168.1.34	4.231.66.184	TCP	54	52393 → 443 [ACK] Seq=4317 Win=26256 Len=0
490	2.887729	192.168.1.34	4.231.66.184	TLsv1.3	85	Application Data
491	2.890004	4.231.66.184	192.168.1.34	TCP	56	443 → 52393 [ACK] Seq=8916 Ack=4317 Win=0 Len=0
492	2.895641	4.231.66.184	192.168.1.34	TLsv1.3	1840	Application Data
493	2.895864	192.168.1.34	4.231.66.184	TCP	54	52393 → 443 [ACK] Seq=4348 Ack=9982 Win=263424 Len=0
494	2.920046	20.189.173.5	192.168.1.34	TCP	56	443 → 52394 [ACK] Seq=8465 Ack=124783 Win=4194568 Len=0
495	2.920046	20.189.173.5	192.168.1.34	TCP	56	443 → 52394 [ACK] Seq=8465 Ack=124783 Win=4194568 Len=0
496	2.920046	20.189.173.5	192.168.1.34	TCP	56	443 → 52394 [ACK] Seq=8465 Ack=124783 Win=4194568 Len=0
497	2.920046	20.189.173.5	192.168.1.34	TCP	56	443 → 52394 [ACK] Seq=8465 Ack=124783 Win=4194568 Len=0
498	2.921585	20.189.173.5	192.168.1.34	TLsv1.3	364	Application Data
499	3.937477	96.7.128.175	192.168.1.34	TCP	66	80 → 52392 [SYN, ACK] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM WS=256
500	3.937570	192.168.1.34	96.7.128.175	TCP	54	52392 → 80 [ACK] Seq=1 Ack=1 Win=26256 Len=0
501	2.937818	192.168.1.34	96.7.128.175	HTTP	586	GET / HTTP/1.1
502	3.952119	4.231.66.184	192.168.1.34	TCP	56	443 → 52393 [ACK] Seq=9982 Ack=4348 Win=0 Len=0
503	3.963980	192.168.1.34	20.189.173.5	TCP	54	52384 → 443 [ACK] Seq=136058 Ack=8775 Win=262912 Len=0
504	3.969784	96.7.128.175	192.168.1.34	TCP	54	80 → 52392 [ACK] Seq=1 Ack=533 Win=64128 Len=0
505	3.969784	96.7.128.175	192.168.1.34	HTTP	1032	HTTP/1.1 200 OK (text/html)
506	3.147380	192.168.1.34	96.7.128.175	TCP	54	52392 → 80 [ACK] Seq=535 Ack=979 Win=261832 Len=0
507	3.166622	192.168.1.34	20.189.173.5	TCP	1494	52384 → 443 [ACK] Seq=137498 Ack=8775 Win=262912 Len=1440 [TCP PDU reassembled in 511]
508	3.166622	192.168.1.34	20.189.173.5	TCP	1494	52384 → 443 [ACK] Seq=137498 Ack=8775 Win=262912 Len=1440 [TCP PDU reassembled in 511]
509	3.166622	192.168.1.34	20.189.173.5	TCP	1494	52384 → 443 [ACK] Seq=137498 Ack=8775 Win=262912 Len=1440 [TCP PDU reassembled in 511]
510	3.166622	192.168.1.34	20.189.173.5	TCP	1494	52384 → 443 [ACK] Seq=137498 Ack=8775 Win=262912 Len=1440 [TCP PDU reassembled in 511]
511	3.166622	192.168.1.34	20.189.173.5	TLsv1.3	506	Application Data
512	3.166817	192.168.1.34	20.189.173.5	TCP	1494	52384 → 443 [ACK] Seq=142278 Ack=8775 Win=262912 Len=1440 [TCP PDU reassembled in 517]
513	3.166817	192.168.1.34	20.189.173.5	TCP	1494	52384 → 443 [ACK] Seq=142278 Ack=8775 Win=262912 Len=1440 [TCP PDU reassembled in 517]
514	3.166817	192.168.1.34	20.189.173.5	TCP	1494	52384 → 443 [ACK] Seq=142278 Ack=8775 Win=262912 Len=1440 [TCP PDU reassembled in 517]
515	3.166817	192.168.1.34	20.189.173.5	TCP	1494	52384 → 443 [ACK] Seq=142278 Ack=8775 Win=262912 Len=1440 [TCP PDU reassembled in 517]
516	3.166817	192.168.1.34	20.189.173.5	TCP	1494	52384 → 443 [ACK] Seq=142278 Ack=8775 Win=262912 Len=1440 [TCP PDU reassembled in 517]
517	3.166817	192.168.1.34	20.189.173.5	TLsv1.3	1061	Application Data
518	3.314453	20.189.173.5	192.168.1.34	TCP	56	443 → 52394 [ACK] Seq=8775 Ack=139338 Win=4194568 Len=0
519	3.314453	20.189.173.5	192.168.1.34	TCP	56	443 → 52394 [ACK] Seq=8775 Ack=139338 Win=4194568 Len=0
520	3.316370	20.189.173.5	192.168.1.34	TCP	56	443 → 52394 [ACK] Seq=8775 Ack=139338 Win=4194568 Len=0
521	3.318369	20.189.173.5	192.168.1.34	TLsv1.3	365	Application Data
522	3.364196	192.168.1.34	20.189.173.5	TCP	54	52384 → 443 [ACK] Seq=136058 Ack=8775 Win=262912 Len=0
523	3.457380	Sagemcombro_c2:07...	Spanning-tree (For-...	111	Ethernet II	
524	3.457380	Sagemcombro_c2:07...	Spanning-tree (For-...	182	Ethernet II	
525	3.457485	Sagemcombro_c2:07...	Spanning-tree (For-...	180	Ethernet II	
526	3.743330	Sagemcombro_c2:07...	Broadcast	52	Who has 192.168.1.100? Tell 192.168.1.1	
527	3.051100	SercomFrame_0b:31...	Broadcast	52	Who has 192.168.1.1? Tell 192.168.1.1	
528	4.168791	192.168.1.34	96.7.128.175	HTTP	500	GET /favicon.ico HTTP/1.1
529	4.223325	4.207.247.137	192.168.1.34	TLsv1.2	433	Application Data
530	4.223950	192.168.1.34	4.207.247.137	TLsv1.2	188	Application Data
531	4.260050	192.168.1.34	31.13.83.49	TCP	66	52394 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
532	4.267379	31.13.83.49	192.168.1.34	TCP	66	80 → 52394 [SYN, ACK] Seq=0 Win=0 Len=0 MSS=1460 WS=256 SACK_PERM WS=256
533	4.267493	192.168.1.34	31.13.83.49	TCP	54	52394 → 80 [ACK] Seq=1 Ack=1 Win=262912 Len=0
534	4.267853	192.168.1.34	31.13.83.49	TCP	471	52394 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262912 Len=417 [TCP PDU reassembled in 613]
535	4.273355	192.168.1.34	31.13.83.49	TCP	54	80 → 52394 [ACK] Seq=1 Ack=418 Win=69816 Len=0



No.	Time	Source	Destination	Protocol	Length	Info
501	2.937818	192.168.1.34	96.7.128.175	HTTP	586	GET / HTTP/1.1
505	3.099704	192.168.1.34	192.168.1.34	HTTP	1032	HTTP/1.1 200 OK (text/html)
528	4.168791	192.168.1.34	96.7.128.175	HTTP	500	GET /favicon.ico HTTP/1.1
540	4.330773	96.7.128.175	192.168.1.34	HTTP	242	HTTP/1.1 404 Not Found (text/html)

- Paquete 501: Mi petición GET

- Este es nuestro navegador pidiendo la página principal de example.com. Si seleccionamos este paquete en el panel de detalles de paquetes muestra claramente el host.

```

0000 cc 00 f1 c2 07 40 ac 15 a2 40 20 64 08 00 45 00 .....@...@ d·E·
0010 02 3c 46 d8 40 00 80 06 0f 63 c0 a8 01 22 60 07 ·<F·@...·c...·`·
0020 80 af cc a8 00 50 55 ac 4a ab b4 94 0f 15 50 18 .....PU· J·...·P·
0030 04 02 c0 e6 00 00 47 45 54 20 2f 20 48 54 54 50 .....GE T / HTTP
0040 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 65 78 61 6d /1.1·Ho st: exam
0050 70 6c 65 2e 63 6f 6d 0d 0a 43 6f 6e 6e 65 63 74 ple.com· ·Connect
0060 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d ion: kee p-alive
0070 0a 44 4e 54 3a 20 31 0d 0a 55 70 67 72 61 64 65 ·DNT: 1· ·Upgrade
0080 2d 49 6e 73 65 63 75 72 65 2d 52 65 71 75 65 73 ·Insecur e-Reques
0090 74 73 3a 20 31 0d 0a 55 73 65 72 2d 41 67 65 6e ts: 1· ·U ser-Agen
00a0 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 t: Mozil la/5.0 (
00b0 57 69 6e 64 6f 77 73 20 4e 54 20 31 30 2e 30 3b Windows NT 10.0;
00c0 20 57 69 6e 36 34 3b 20 78 36 34 29 20 41 70 70 Win64; x64) App
00d0 6c 65 57 65 62 4b 69 74 2f 35 33 37 2e 33 36 20 leWebKit /537.36
00e0 28 4b 48 54 4d 4c 2c 20 6c 69 6b 65 20 47 65 63 (KHTML, like Gec
00f0 6b 6f 29 20 43 68 72 6f 6d 65 2f 31 33 36 2e 30 ko) Chro me/136.0
0100 2e 30 2e 30 20 53 61 66 61 72 69 2f 35 33 37 2e .0.0 Saf ari/537.
0110 33 36 20 45 64 67 2f 31 33 36 2e 30 2e 30 2e 30 36 Edg/1 36.0.0.0
0120 0d 0a 41 63 63 65 70 74 3a 20 74 65 78 74 2f 68 ··Accept : text/h
0130 74 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f tml,application/
0140 78 68 74 6d 6c 2b 78 6d 6c 2c 61 70 70 6c 69 63 xhtml+xm l,applic
0150 61 74 69 6f 6e 2f 78 6d 6c 3b 71 3d 30 2e 39 2c ation/xm l;q=0.9,
0160 69 6d 61 67 65 2f 61 76 69 66 2c 69 6d 61 67 65 image/av if,image
0170 2f 77 65 62 70 2c 69 6d 61 67 65 2f 61 70 6e 67 /webp,im age/apng
0180 2c 2a 2f 2a 3b 71 3d 30 2e 38 2c 61 70 70 6c 69 ,/*;q=0 .8,appli
0190 63 61 74 69 6f 6e 2f 73 69 67 6e 65 64 2d 65 78 cation/s igned-ex
01a0 63 68 61 6e 67 65 3b 76 3d 62 33 3b 71 3d 30 2e change;v =b3;q=0.
01b0 37 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 7 ··Accep t-Encodi
01c0 6e 67 3a 20 67 7a 69 70 2c 20 64 65 66 6c 61 74 ng: gzip , deflat
01d0 65 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 e· Accep t-Langua
01e0 67 65 3a 20 65 73 2d 34 31 39 2c 65 73 3b 71 3d ge: es-4 19,es;q=
01f0 30 2e 39 2c 65 73 2d 45 53 3b 71 3d 30 2e 38 2c 0.9,es-E S;q=0.8,

```

No: 501 · Time: 2.937818 · Source: 192.168.1.34 · Destination: 96.7.128.175 · Protocol: HTTP · Length: 586 · Info: GET / HTTP/1.1

- Paquete 505: Respuesta del Servidor 200 OK.
 - El servidor nos responde que todo está bien y nos envía la página, que es de tipo text/html.
- Paquete 528: Petición GET para favicon.ico
 - Después de cargar la página principal, los navegadores casi siempre intentan pedir un archivo llamado favicon.ico. Este es el pequeño icono que aparece en la pestaña del navegador. Es una petición HTTP normal.
- Paquete 540 (Respuesta del Servidor 404 Not Found para favicon.ico):
 - El servidor de example.com responde que no tiene un archivo llamado favicon.ico (404 Not Found). Esto es muy común para páginas simples.

El análisis de la captura para <http://example.com/> demuestra claramente cómo las peticiones y respuestas HTTP viajan en texto plano. Se pudo identificar el host solicitado (example.com) a través de la cabecera Host en la petición GET, y tanto la solicitud como el contenido de la respuesta son visibles para cualquiera que capture el tráfico.

- HTTPS: Navego a un sitio HTTPS (WEB SEGURA)

La conversación entre nuestro ordenador y el servidor web estará cifrada, como si la "postal" fuera dentro de un sobre cerrado con un código secreto. Wireshark seguirá viendo los sobres, pero no podrá leer fácilmente lo que hay dentro después de que se establezca la seguridad. Voy a usar la pagina HTTPS de prueba: <https://www.wikipedia.org>

The image shows a Wireshark packet capture of a TLS connection. The packet list on the left shows a series of packets, with packet 597 highlighted. The packet details pane on the right shows the structure of the selected packet, which is a TLSv1.3 Client Hello. The packet bytes pane at the bottom shows the raw hex and ASCII data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
544	4.689590	192.168.1.34	20.42.73.28	TLSv1.3	497	Application Data
554	4.689754	192.168.1.34	20.42.73.28	TLSv1.3	630	Application Data
559	4.689961	192.168.1.34	20.42.73.28	TLSv1.3	1384	Application Data
570	4.695817	20.42.73.28	192.168.1.34	TLSv1.3	89	Application Data
572	4.821187	20.42.73.28	192.168.1.34	TLSv1.3	414	Application Data
584	4.956649	192.168.1.34	20.42.73.28	TLSv1.3	512	Application Data
590	4.957094	192.168.1.34	20.42.73.28	TLSv1.3	726	Application Data
597	4.997612	192.168.1.34	185.15.58.224	TLSv1.3	441	Client Hello (SNI=www.wikipedia.org)
599	5.031854	185.15.58.224	192.168.1.34	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
601	5.031854	185.15.58.224	192.168.1.34	TLSv1.3	1800	Application Data, Application Data, Application Data
603	5.034813	192.168.1.34	185.15.58.224	TLSv1.3	118	Change Cipher Spec, Application Data
604	5.035086	192.168.1.34	185.15.58.224	TLSv1.3	146	Application Data
605	5.035169	192.168.1.34	185.15.58.224	TLSv1.3	685	Application Data
609	5.067493	185.15.58.224	192.168.1.34	TLSv1.3	309	Application Data
610	5.067493	185.15.58.224	192.168.1.34	TLSv1.3	313	Application Data
611	5.067493	185.15.58.224	192.168.1.34	TLSv1.3	186	Application Data
613	5.067725	192.168.1.34	185.15.58.224	TLSv1.3	85	Application Data
628	5.101815	185.15.58.224	192.168.1.34	TLSv1.3	1514	Application Data
634	5.101819	185.15.58.224	192.168.1.34	TLSv1.3	1412	Application Data
636	5.108648	192.168.1.34	185.15.58.224	TLSv1.3	344	Application Data
641	5.161284	192.168.1.34	20.42.73.28	TLSv1.3	512	Application Data
647	5.161357	192.168.1.34	20.42.73.28	TLSv1.3	889	Application Data
648	5.162461	192.168.1.34	185.15.58.224	TLSv1.3	158	Application Data
650	5.194513	20.42.73.28	192.168.1.34	TLSv1.3	233	Application Data
665	5.201050	185.15.58.224	192.168.1.34	TLSv1.3	1514	Application Data
666	5.201050	185.15.58.224	192.168.1.34	TLSv1.3	1514	Application Data
674	5.208445	185.15.58.224	192.168.1.34	TLSv1.3	502	Application Data
676	5.211380	192.168.1.34	185.15.58.224	TLSv1.3	149	Application Data
677	5.211620	192.168.1.34	185.15.58.224	TLSv1.3	158	Application Data
683	5.227791	192.168.1.34	185.15.58.224	TLSv1.3	151	Application Data
684	5.229059	192.168.1.34	185.15.58.224	TLSv1.3	152	Application Data
686	5.269552	185.15.58.224	192.168.1.34	TLSv1.3	1398	Application Data
693	5.295923	185.15.58.224	192.168.1.34	TLSv1.3	457	Application Data
700	5.207040	185.15.58.224	192.168.1.34	TLSv1.3	1514	Application Data
712	5.269518	185.15.58.224	192.168.1.34	TLSv1.3	128	Application Data
715	5.270861	185.15.58.224	192.168.1.34	TLSv1.3	1400	Application Data
717	5.326917	20.42.73.28	192.168.1.34	TLSv1.3	233	Application Data
727	5.558396	192.168.1.34	150.171.28.11	TLSv1.2	480	Client Hello (SNI=edge.microsoft.com)
734	5.560805	150.171.28.11	192.168.1.34	TLSv1.2	187	Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
735	5.567664	192.168.1.34	150.171.28.11	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
736	5.567648	192.168.1.34	150.171.28.11	TLSv1.2	153	Application Data
737	5.568803	192.168.1.34	150.171.28.11	TLSv1.2	471	Application Data
739	5.573874	150.171.28.11	192.168.1.34	TLSv1.2	398	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
740	5.575266	150.171.28.11	192.168.1.34	TLSv1.2	113	Application Data
741	5.575266	150.171.28.11	192.168.1.34	TLSv1.2	726	Application Data, Application Data
742	5.575266	150.171.28.11	192.168.1.34	TLSv1.2	92	Application Data
744	5.575500	192.168.1.34	150.171.28.11	TLSv1.2	92	Application Data
747	6.475988	192.168.1.34	185.15.58.224	TLSv1.2	152	Application Data
749	6.508955	185.15.58.224	192.168.1.34	TLSv1.3	397	Application Data

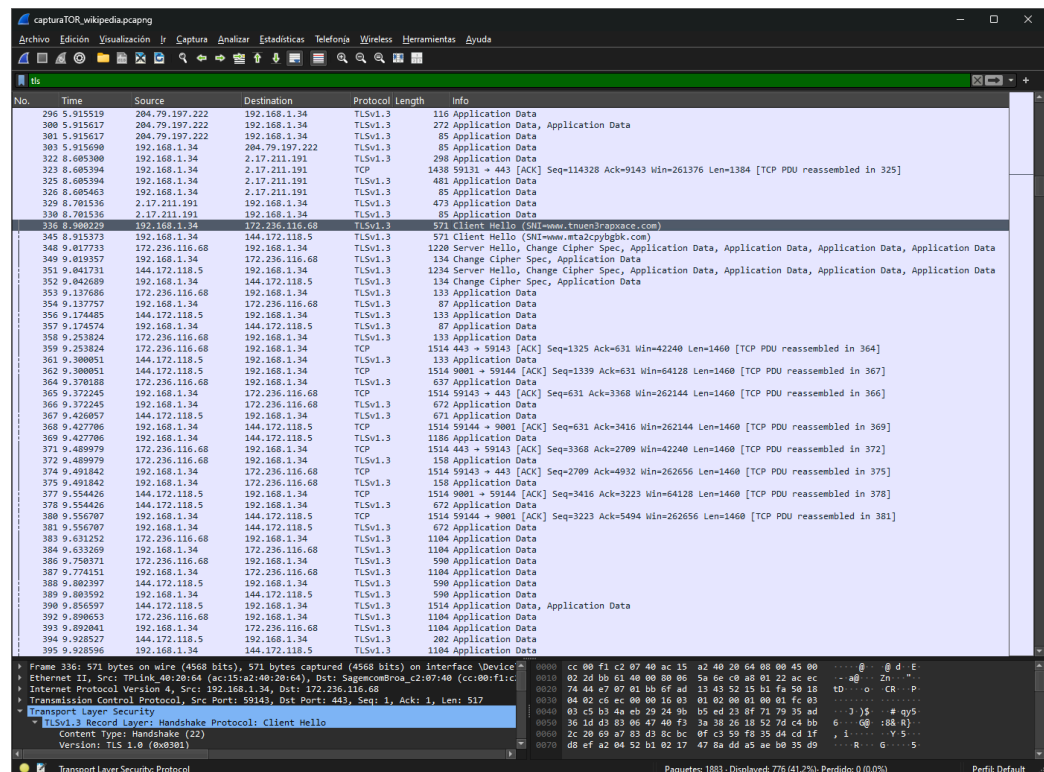
- Muchos paquetes Application Data: Filas 544-598...
- El navegador y el servidor de Wikipedia están intercambiando la información de la página. Todo este contenido está cifrado, no se puede leer directamente en Wireshark. Estos paquetes demuestran que la comunicación principal está protegida
- El Handshake TLS
 - Paquete 597. Este es el Client Hello que envía nuestro navegador. La parte (SNI=www.wikipedia.org) es crucial. Antes de que se establezca el cifrado completo, el navegador le dice al servidor que quiere establecer una conexión segura con la página. Esto es visible en texto plano dentro de este paquete inicial del handshake. Este es el punto donde se puede identificar el sitio web al que intentas acceder, incluso antes de que la comunicación esté totalmente cifrada.

▼	TLSv1.3 Record Layer: Handshake Protocol: Client Hello
	Content Type: Handshake (22)
	Version: TLS 1.0 (0x0301)
	Length: 1818
▼	Handshake Protocol: Client Hello
	Handshake Type: Client Hello (1)
	Length: 1814
▶	Version: TLS 1.2 (0x0303)
	Random: cdfef49809261814b0cf14e3b055376992e8b5a9ea866c28a3b5255d30c6e6daf
	Session ID Length: 32
	Session ID: 7e492ff9a4f1772d7d80ba459cf02a5b515a41a810d21d724a8a04dd4b0dad7c
	Cipher Suites Length: 32
▶	Cipher Suites (16 suites)
	Compression Methods Length: 1
	Compression Methods (1 method)
	Extensions Length: 1709
▶	Extension: Reserved (GREASE) (len=0)
▶	Extension: encrypted_client_hello (len=282)
▶	Extension: extended_master_secret (len=0)
▶	Extension: supported_groups (len=12)
▶	Extension: session_ticket (len=0)
▶	Extension: signature_algorithms (len=18)
▶	Extension: ec_point_formats (len=2)
▼	Extension: server_name (len=22) name=www.wikipedia.org
	Type: server_name (0)
	Length: 22

- Paquete 599
 - El servidor responde al Client Hello. El Server Hello contiene los parámetros de cifrado elegidos por el servidor. Change Cipher Spec indica que el servidor va a empezar a usar cifrado. El Application Data aquí probablemente contiene el resto de los mensajes del handshake del servidor, ya cifrados con las claves temporales del handshake.
- Paquete 601:
 - Nuestro navegador también dice "Ok, yo también empiezo a cifrar (Change Cipher Spec)" y el Application Data es el final del handshake desde nuestro lado, ya cifrado.

2. Segundo Escenario: Navegación Web en TOR hacia Clear Web

Abrimos TOR Browser. Esperamos a que nos indique que está conectado a la red TOR. Luego, en la barra de direcciones, escribimos la dirección de un sitio web normal (de la "clear web"), por ejemplo, <https://www.wikipedia.org>.

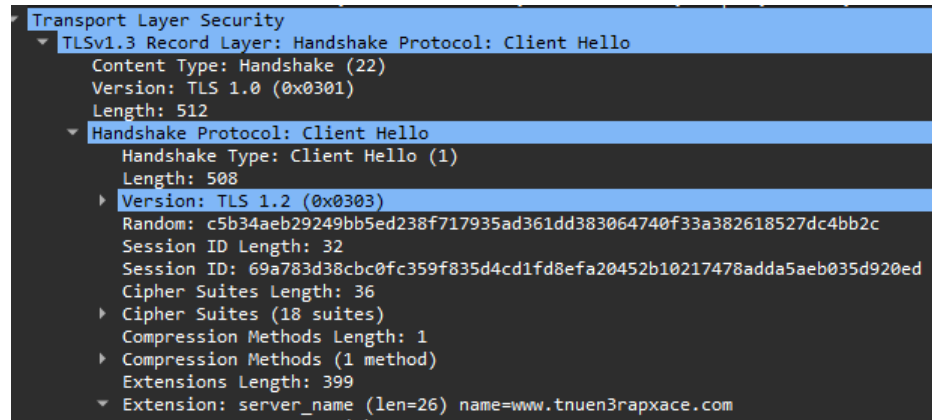


No vemos peticiones DNS directas desde mi IP para wikipedia, ni conexiones TCP/TLS directas desde mi IP a las direcciones IP conocidas de wikipedia (como las que vimos en el escenario HTTPS directo).

Lo que si vemos ahora:

- Conexión inicial a la Red TOR
 - Al principio de la captura, poco después de abrir TOR, vemos tráfico desde mi IP a varias IP desconocidas. El protocolo para estas conexiones es TLSv1.2 o TLSv1.3. Los puertos de destino en estos

servidores remotos suelen ser el 443 o el 9001 (puertos comunes para los nodos TOR). Estas IPs son los nodos de entrada (entry relays o guard relays) de la red TOR. TOR Browser se conecta a uno de estos para entrar a la red.



El SNI aquí no es wikipedia.org, es tnuen3rapxace.com. Cuando un TOR Browser establece la conexión cifrada con el primer nodo TOR, el SNI que se usa es para el propio nodo TOR o un dominio genérico que usa TOR para el handshake, no para el sitio web final que vamos a visitar. Wikipedia se solicitará dentro de este túnel cifrado. Toda la navegación hacia wikipedia está ocurriendo dentro de flujos de Application Data cifrados con los nodos TOR. No podemos ver wikipedia directamente.

- Tráfico de Navegación Encapsulado y Cifrado
 - Una vez que TOR Browser está conectado y empezamos a navegar hacia www.wikipedia.org, todo ese tráfico se enviará primero al nodo de entrada TOR al que te conectaste. En Wireshark, seguimos viendo paquetes TLS y luego Application Data dentro de TLS, entre nuestra IP (192.168.1.34) y la IP de ese nodo de entrada TOR.
 - No se ve ninguna comunicación directa con las IPs de Wikipedia. El contenido de nuestra petición a Wikipedia y la respuesta de Wikipedia están cifrados y encapsulados dentro de la comunicación que tenemos con el nodo TOR.

Se demuestra cómo la red TOR anonimiza el tráfico. En lugar de conexiones directas al servidor de destino, se observan conexiones cifradas a nodos de la red TOR. La identidad del sitio web final y el contenido de la comunicación están protegidos de la observación local, cumpliendo el objetivo de privacidad de TOR.

3. Tercer Escenario: Navegación Web en TOR hacia Dominios .onion

Ahora visitamos una dirección de un sitio .onion. El ejemplo que usamos es el de DuckDuckGo: <https://duckduckgogg42xjoc72x3sjasowoarfbgcmvfimaftt6twagswzczad.onion/>.

The screenshot shows a Wireshark capture of a TLS handshake. The packet list displays a 'Hello Retry Request' from source 204.79.197.222 to destination 192.168.1.34. The packet details pane shows the 'Transport Layer Security' section, specifically 'TLSv1.3 Record Layer: Handshake Protocol: Hello Retry Request'.

Es similar al escenario anterior hacia clear web en cuanto a la conexión iniciar a la red TOR, pero hay algunas diferencias sutiles.

No vemos:

- Peticiones DNS tradicionales: Los dominios .onion no se resuelven mediante el sistema DNS público. Su resolución ocurre dentro de la red TOR utilizando un sistema de directorios distribuidos (Distributed Hash Table - DHT) y servicios de introducción
- Conexiones directas a una IP pública del servicio .onion.

The screenshot shows a Wireshark capture of a DNS query response. The packet list displays a 'Standard query response' from source 192.168.1.1 to destination 192.168.1.34. The packet details pane shows the 'DNS' section, specifically 'Standard query response'.

Hay varias consultas DNS con nombres de dominio variados de clear web, pero hay ausencia de consulta para el dominio .onion. Esto confirma que no se resuelven a través del sistema DNS público tradicional. Se gestiona internamente por la red TOR

Lo que seguimos viendo similar al escenario anterior son las conexiones iniciales a la red TOR y el tráfico de navegación encapsulado y cifrado hacia nodos TOR.