



Encrypted Protocol Analysis: Decrypting HTTPS Write Up

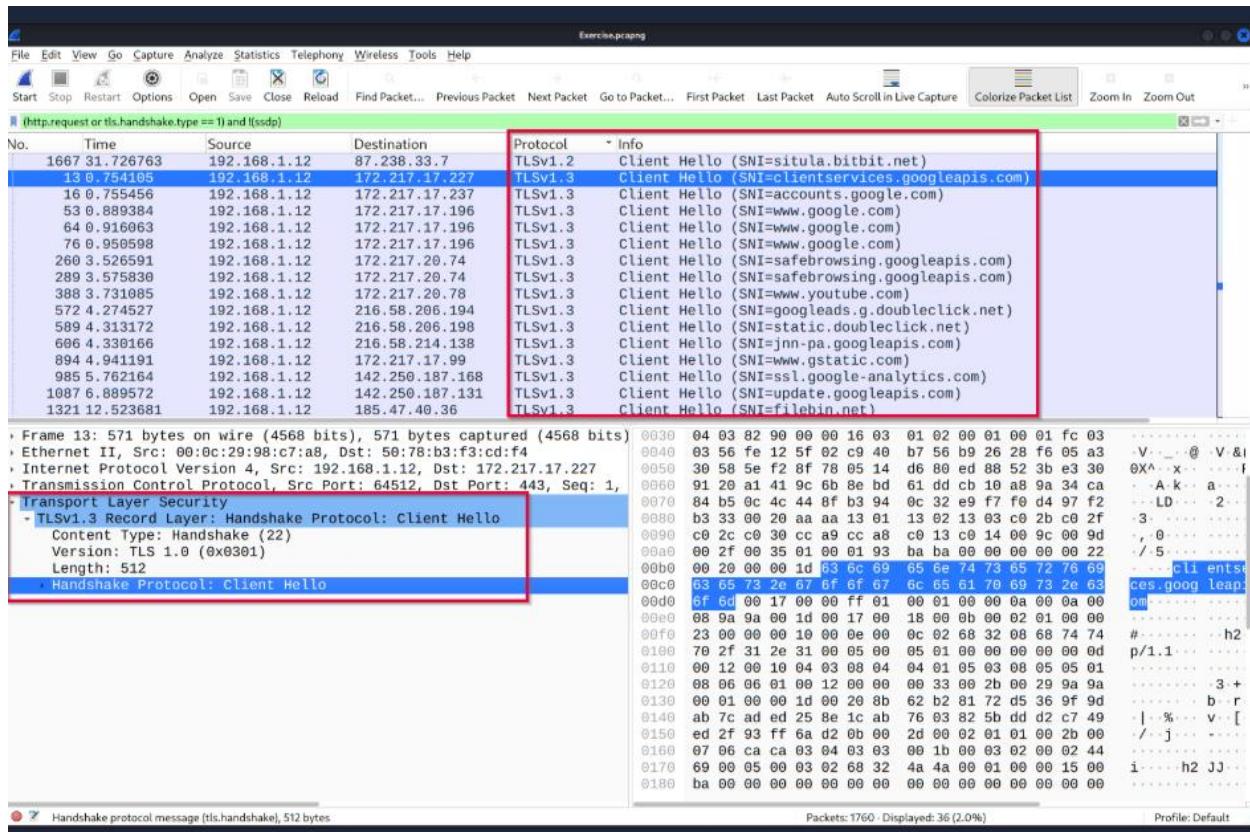
- Sometime, malicious activity is also done in HTTPS (Hypertext Transfer Protocol Secure)
- HTTPS encrypts the data while it is being transferred to the network.
- Without having the encryption and decryption pairs, it will be difficult to read what is being transferred.

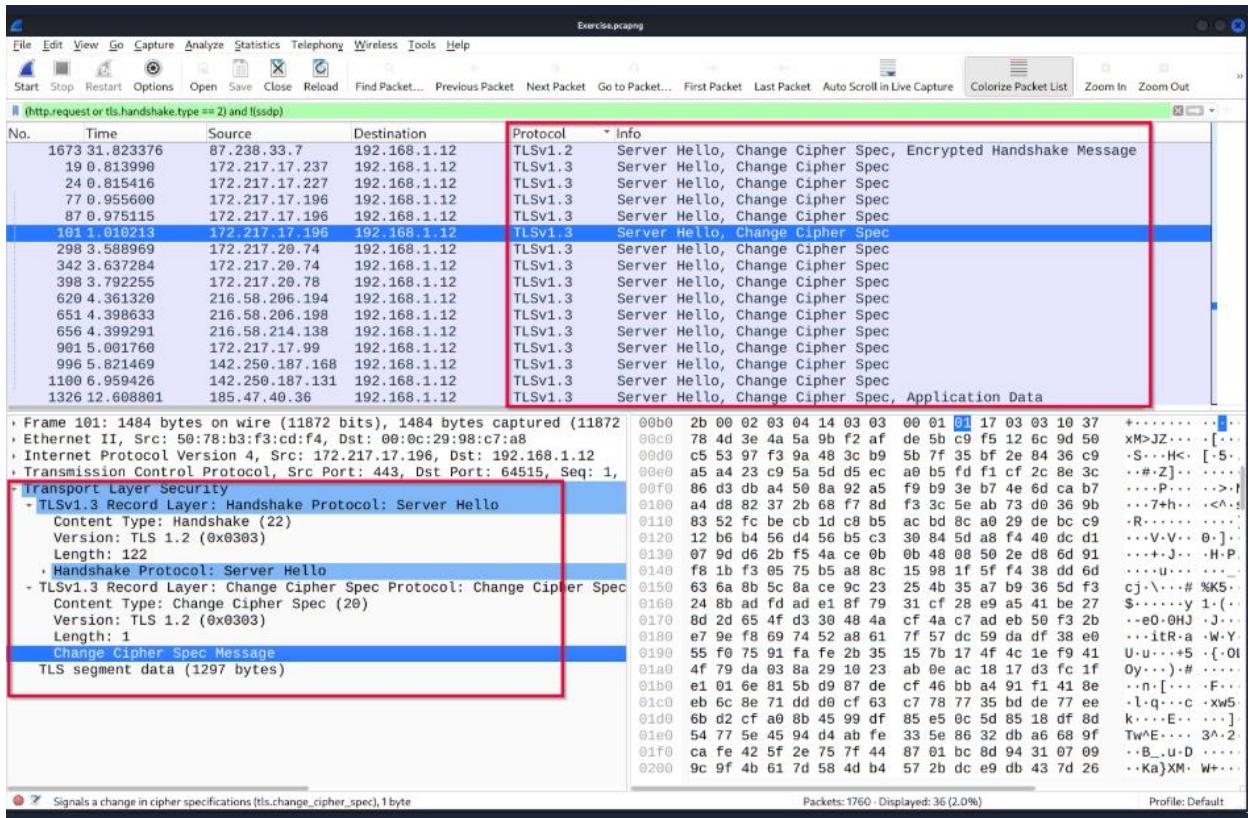
Notes	Wireshark Filter
<p>"HTTPS Parameters" for grabbing the low-hanging fruits:</p> <ul style="list-style-type: none"> • Request: Listing all requests • TLS: Global <u>TLS</u> search • <u>TLS</u> Client Request • <u>TLS</u> Server response • Local Simple Service Discovery Protocol (SSDP) <p>Note: SSDP is a network protocol that provides advertisement and discovery of network services.</p>	<pre>http.request tls tls.handshake.type == 1 tls.handshake.type == 2 ssdp</pre>

Time	Source	Destination	Protocol	Info
7 0.726443	192.168.1.12	239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
8 0.727099	192.168.1.12	172.217.17.237	TCP	64511 -> 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SA.
9 0.727225	192.168.1.12	172.217.17.237	TCP	64512 -> 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SA.
10 0.727688	192.168.1.1	192.168.1.12	DNS	Standard query response 0x4065 A clientservices.googleapis...
11 0.753562	172.217.17.227	192.168.1.12	TCP	443 -> 64512 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=143
12 0.753642	192.168.1.12	172.217.17.227	TCP	64512 -> 443 [ACK] Seq=1 Ack=1 Win=262912 Len=0
13 0.754105	192.168.1.12	172.217.17.227	TLSv1.3	Client Hello (SNI=clientservices.googleapis.com)
14 0.755128	172.217.17.237	192.168.1.12	TCP	443 -> 64511 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=143
15 0.755173	192.168.1.12	172.217.17.237	TCP	64511 -> 443 [ACK] Seq=1 Ack=1 Win=262912 Len=0
16 0.755456	192.168.1.12	172.217.17.237	TLSv1.3	Client Hello (SNI=accounts.google.com)
17 0.781804	172.217.17.227	192.168.1.12	TCP	443 -> 64512 [ACK] Seq=1 Ack=518 Win=66816 Len=0
18 0.785015	172.217.17.237	192.168.1.12	TCP	443 -> 64511 [ACK] Seq=1 Ack=518 Win=66816 Len=0
19 0.813990	172.217.17.237	192.168.1.12	TLSv1.3	Server Hello, Change Cipher Spec
20 0.814226	172.217.17.237	192.168.1.12	TCP	443 -> 64511 [PSH, ACK] Seq=1431 Ack=518 Win=66816 Len=1430
21 0.814255	192.168.1.12	172.217.17.237	TCP	64511 -> 443 [ACK] Seq=518 Ack=2861 Win=262912 Len=0
22 0.815416	172.217.17.237	192.168.1.12	TCP	443 -> 64511 [ACK] Seq=2861 Ack=518 Win=66816 Len=1430 TCP

- Like a TCP three-way-handshake, a TLS also has handshakes between the client and the server.

- Client Hello: (`http.request or tls.handshake.type ==1`) and `!(ssdp)` <--- this is to filter the start of the conversation between the client and the server for both unencrypted (http) and encrypted web traffic (https) and ignores the ssdp (Simple Service Discovery Protocol) that is used by devices such as smart tv's, printers, routers , etc.
- Server Hello: (`http.request or tls.handshake.type ==2`) and `!(ssdp)` <--- this is to filter out responses from the server during an unencrypted or an encrypted connection. This allows us to see the communication of a connection by capturing the client's intent for http or the server response for the TLS.

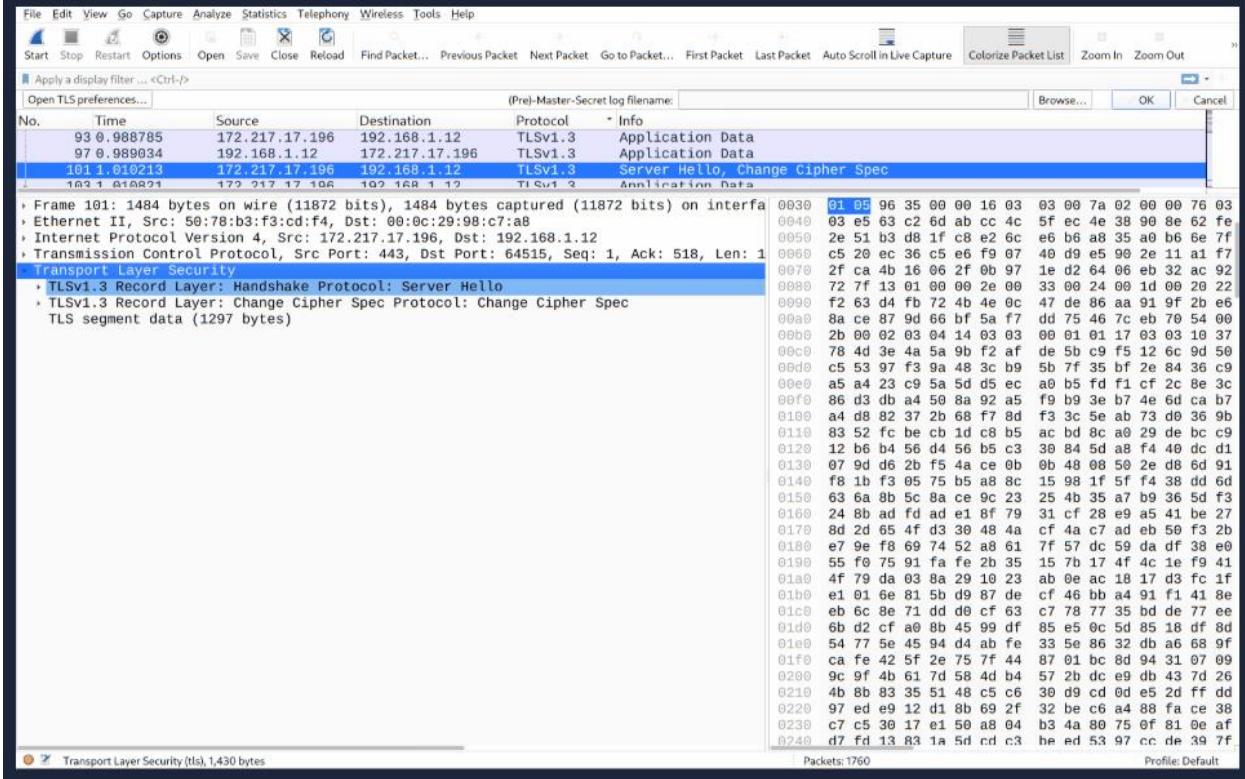




Encrypted Key Log Files

- It is a text file that has unique key pairs to decrypt the encrypted traffic session.
- These key pairs are automatically created when a connection is established with an SSL/TLS-enabled webpage.

Adding key log files with the "right-click" menu:



Adding key log files with the "Edit --> Preferences --> Protocols --> TLS"

Screenshot of Wireshark showing a network capture of a TLS handshake between two hosts. The packet list pane shows several TLSv1.3 frames, with frame 101 highlighted. The details pane shows the handshake process, and the bytes pane shows the raw hex and ASCII data of the selected frame.

Frame 101: 1484 bytes on wire (11872 bits), 1484 bytes captured (11872 bits) on interface

- Ethernet II, Src: 50:78:b3:f3:cd:f4, Dst: 00:0c:29:98:c7:a8
- Internet Protocol Version 4, Src: 172.217.17.196, Dst: 192.168.1.12
- Transmission Control Protocol, Src Port: 443, Dst Port: 64515, Seq: 1, Ack: 518, Len: 1
- Transport Layer Security**
 - TLSv1.3 Record Layer: Handshake Protocol: Server Hello
 - TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

TLS segment data (1297 bytes)

```

0030  01 05 96 35 00 00 16 03  03 00 7a 02 00 00 76 03
0040  03 e5 63 c2 6d ab cc 4c  5f ec 4e 38 90 8e 62 fe
0050  2e 51 b3 d8 1f c8 e2 6c  e6 b6 a8 35 a0 b6 6e 7f
0060  c5 20 ec 36 c5 e6 f9 07  40 d9 e5 90 2e 11 a1 f7
0070  2f ca 4b 16 06 2f 0b 97  1e d2 64 06 eb 32 ac 92
0080  72 7f 13 01 00 00 2e 00  33 00 24 00 1d 00 20 22
0090  f2 63 d4 fb 72 4b 4e 0c  47 de 86 aa 91 9f 2b e6
00a0  8a ce 87 9d 66 bf 5a f7  dd 75 46 7c eb 70 54 00
00b0  2b 00 02 03 04 14 03 03  00 01 01 17 03 03 10 37
00c0  78 4d 3e 4a 5a 9b f2 af  de 5b c9 f5 12 6c 9d 50
00d0  c5 53 97 f3 9a 48 3c b9  5b 7f 35 bf 2e 84 36 c9
00e0  a5 a4 23 c9 5a 5d d5 ec  a0 b5 fd f1 cf 2c 8e 3c
00f0  86 d3 db a4 50 8a 92 a5  f9 b9 3e b7 4e 6d ca b7
0100  a4 d8 82 37 2b 68 f7 8d  f3 3c 5e ab 73 d0 36 9b
0110  83 52 fc be cb 1d c8 b5  ac bd 8c a0 29 dc bc c9
0120  12 b6 b4 56 d4 56 b5 c3  30 84 5d a8 f4 40 dc d1
0130  07 9d d6 2b f5 4a ce 0b  0b 48 08 50 2e d8 6d 91
0140  f8 1b f3 95 75 b5 a8 8c  15 98 1f 5f f4 38 dd 60
0150  63 6a 8b 5c 8a ce 9c 23  25 4b 35 a7 b9 36 5d f3
0160  24 8b ad fd ad e1 8f 79  31 cf 28 e9 a5 41 be 27
0170  8d 2d 65 4f d3 30 48 4a  cf 4a c7 ad eb 50 f3 2b
0180  e7 9e f8 69 74 52 a8 61  7f 57 dc 59 da df 38 e0

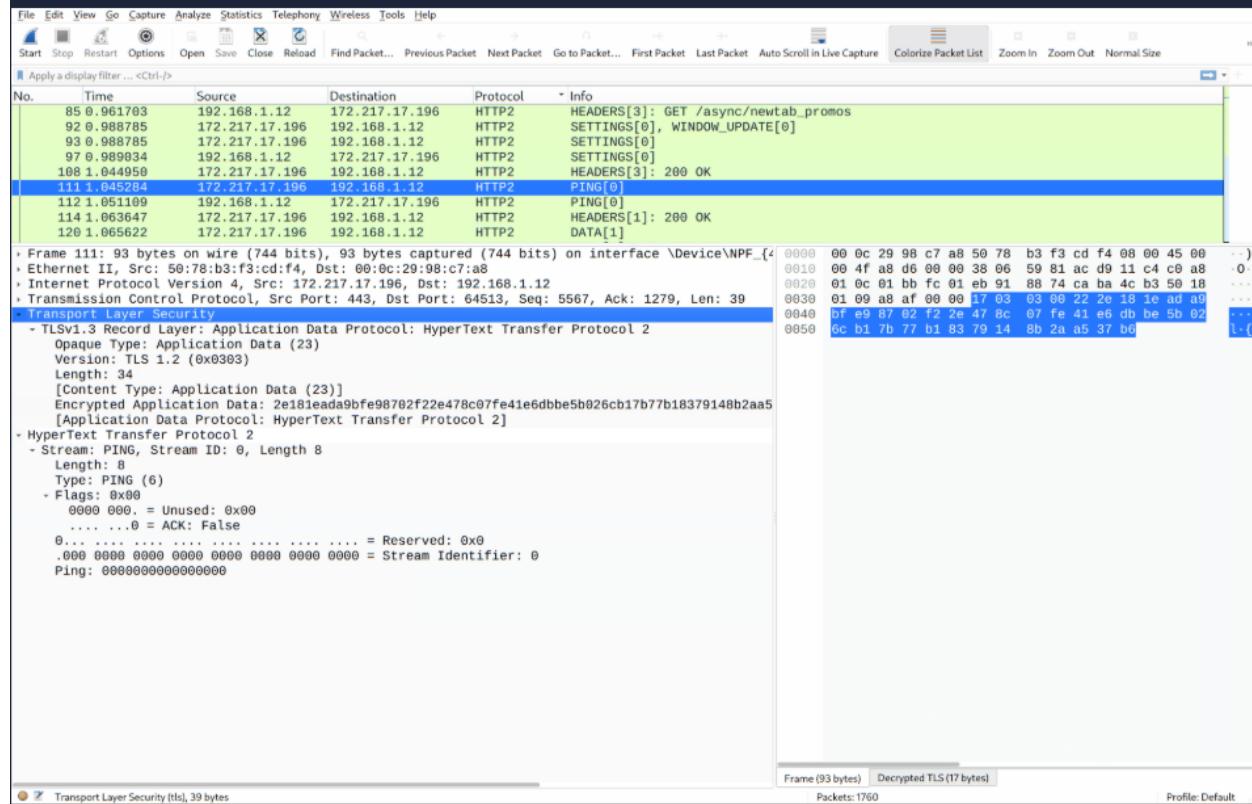
```

Exercise.pcapng

Profile: Default

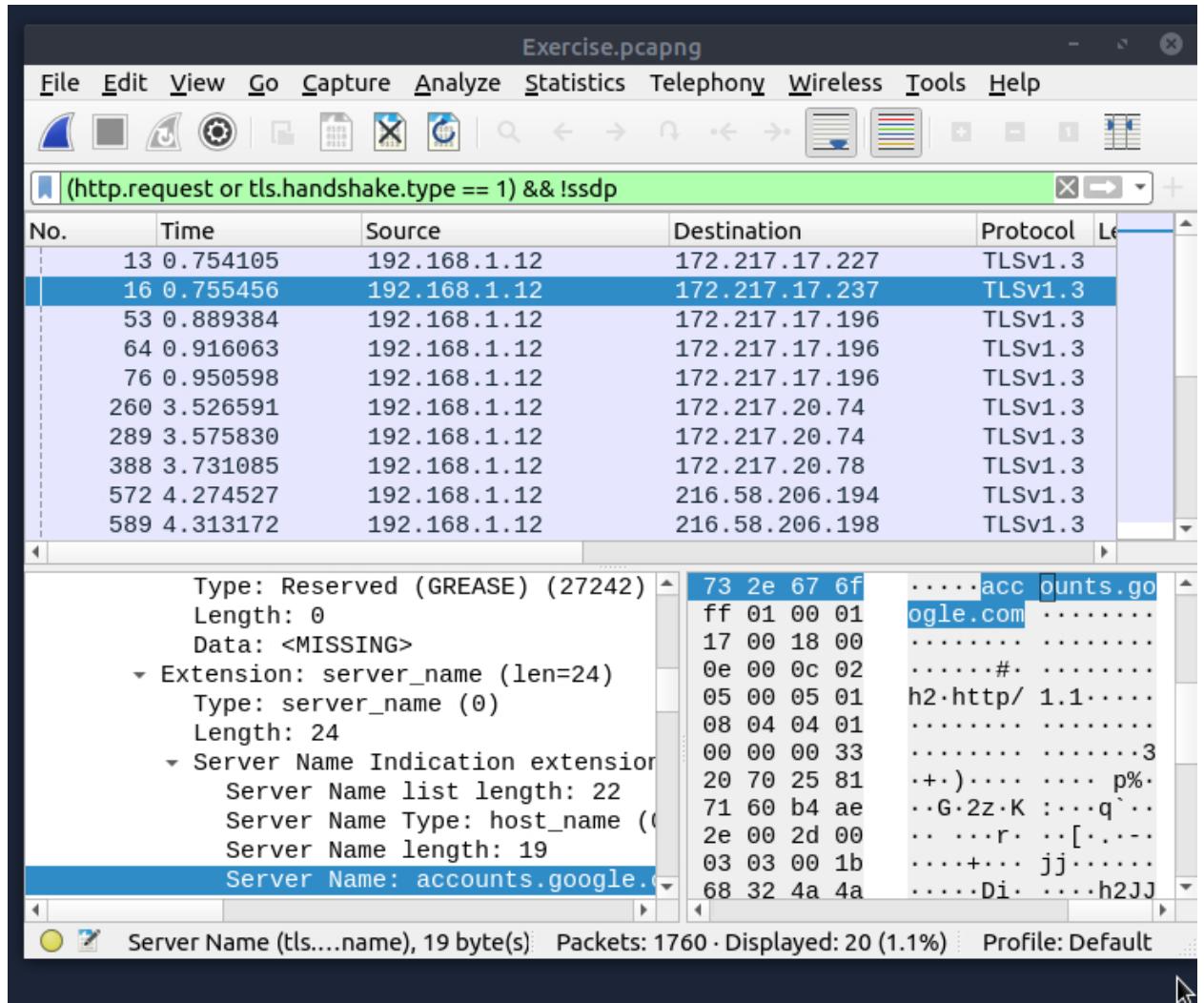
packets: 1760

Viewing the traffic with/without the key log files:



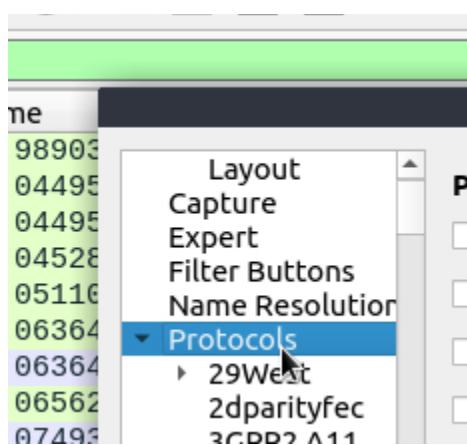
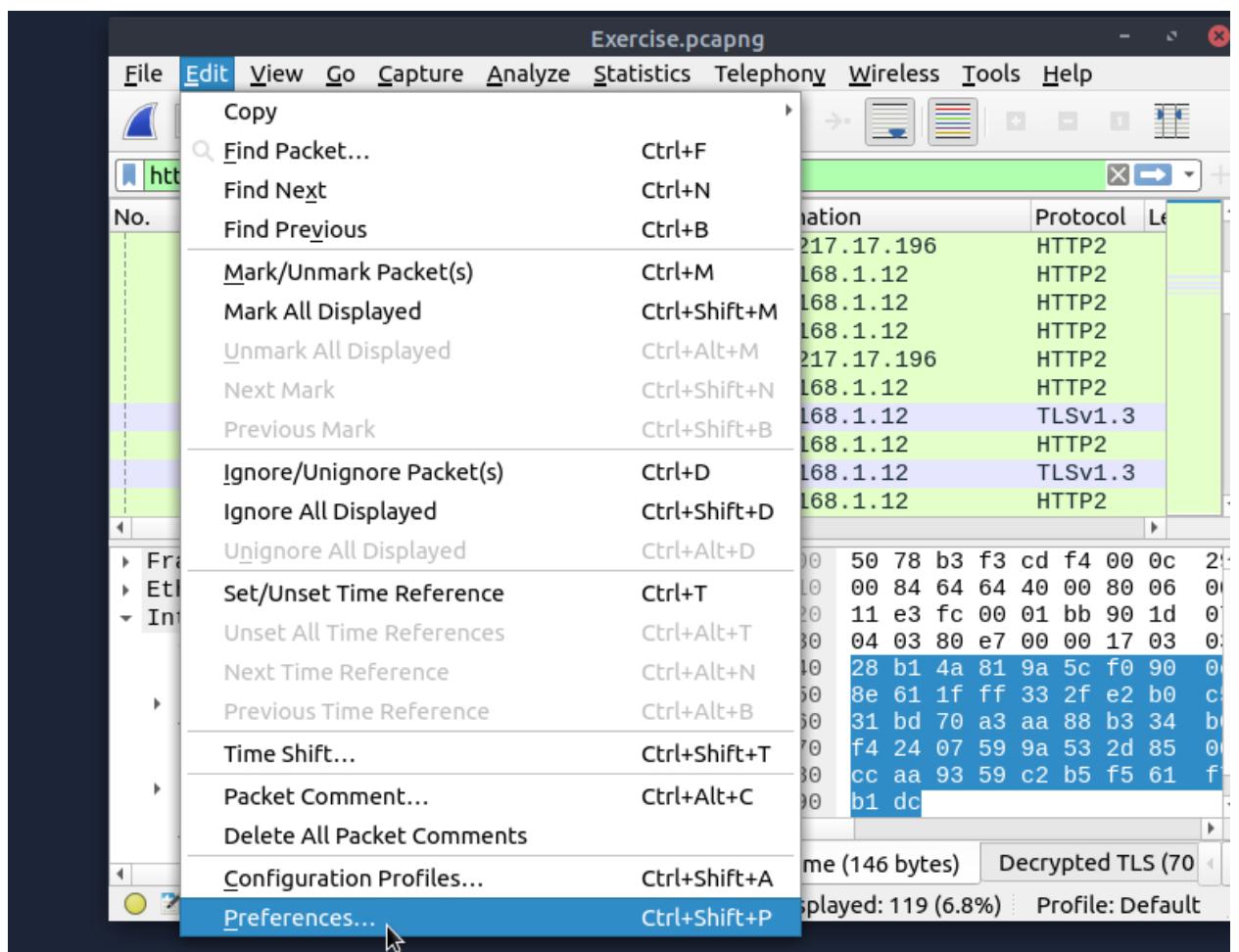
1. What is the frame number of the “Client Hello” message sent to “account.google.com”?

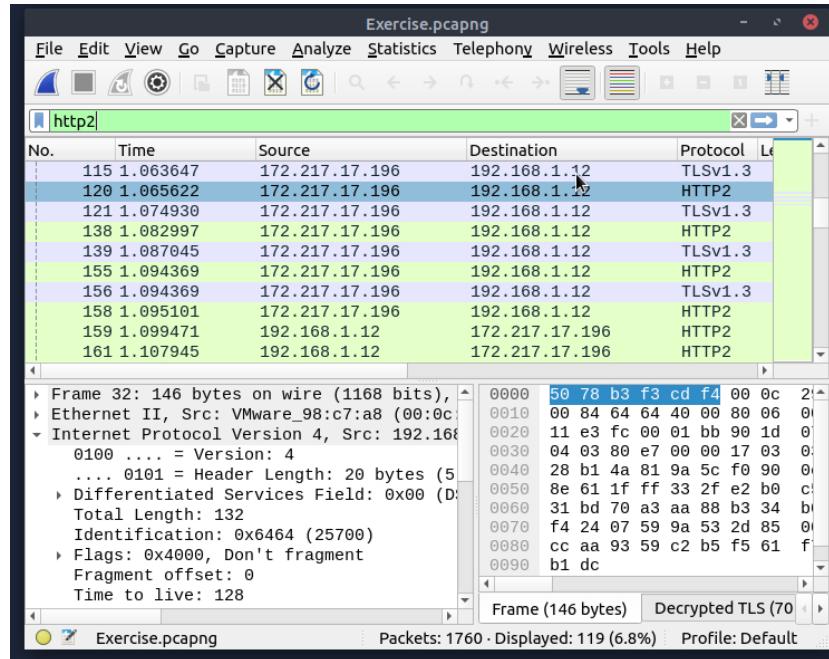
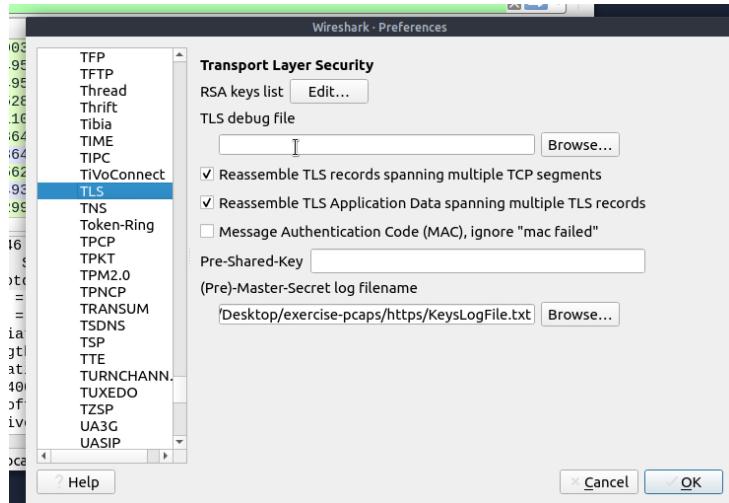
- We will input the filter associating with the client sending a request during a TLS handshake
- ([http.request or tls.handshake.type ==1](#)) && !ssdp <--- filtering the start of a web connection and isolating additional traffic.



Answer: 16

2. Decrypt the Traffic with the “KeysLogFile.txt” file. What is the number of HTTP2 packets?
 - First, we need to add the KeyLogFile.txt



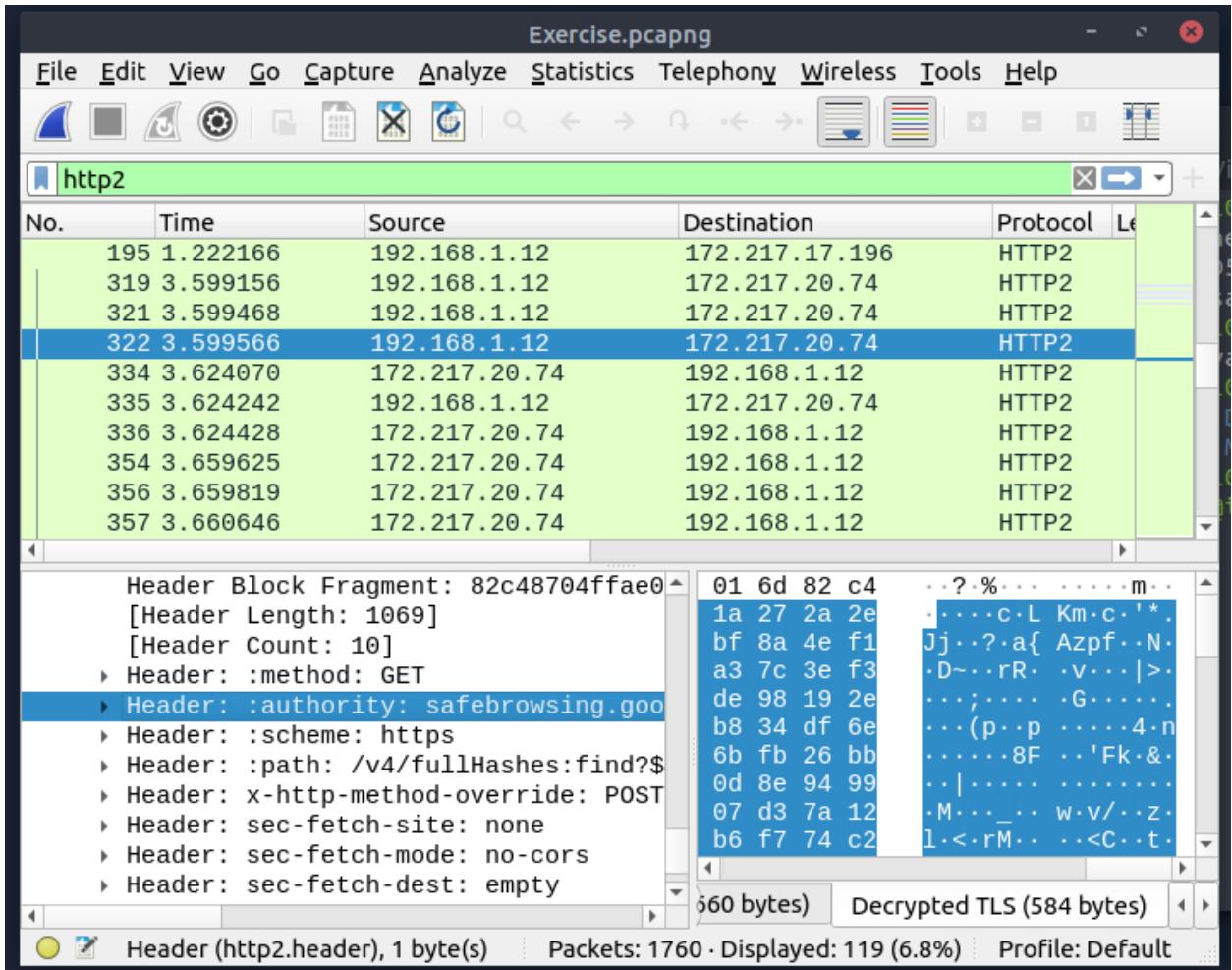


Answer: 115

- There is a total of 119 packets, but notice there are 4 TLSv1.3 packets. We need to deduct the 4 from the 119 displayed packets.

3. Go to Frame 322. What is the authority header of the HTTP2 packet? (Defanged the address)

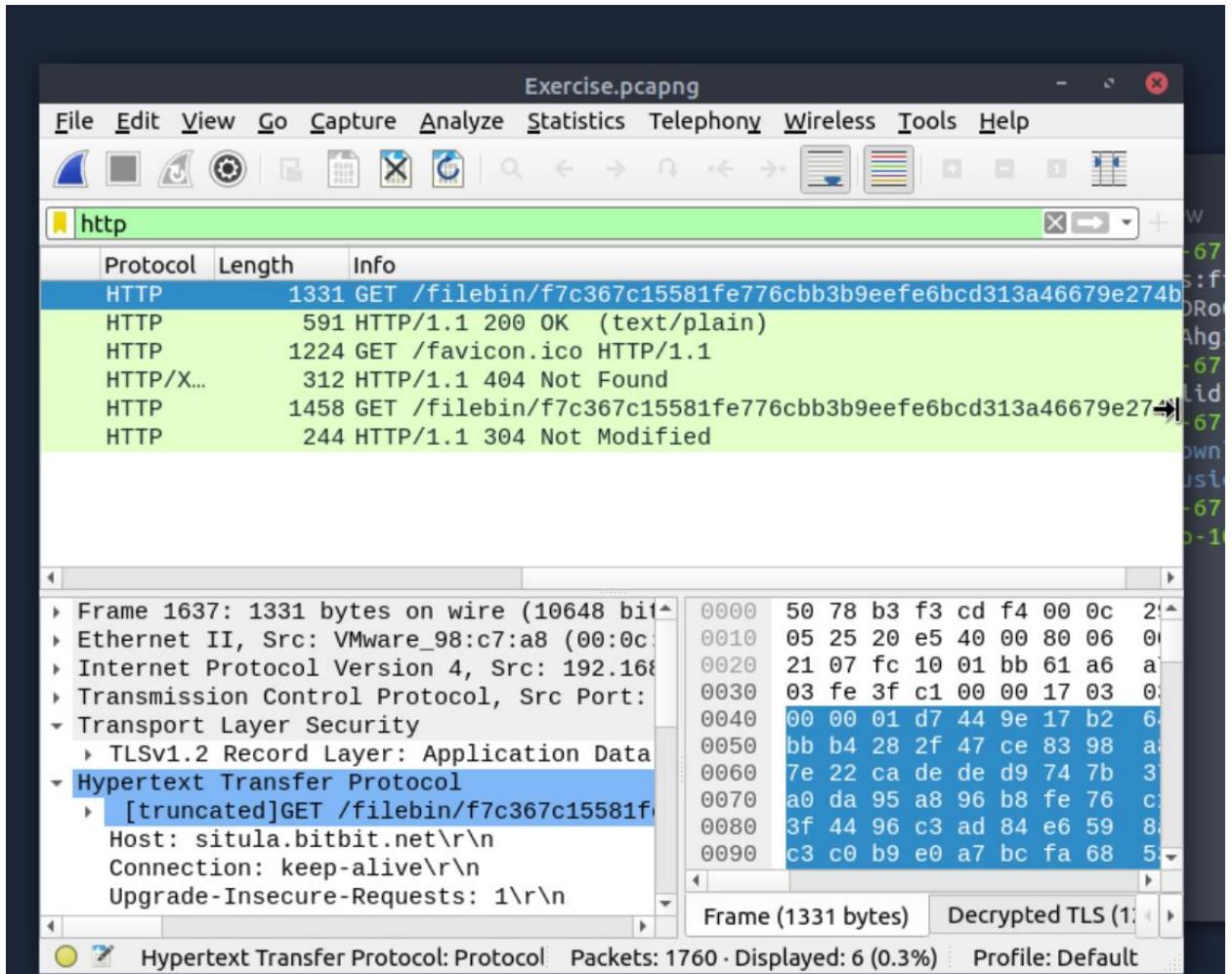
- On the same http2 traffic go to frame 322 –> Hypertext Transfer Protocol 2 -> Stream Headers -> Headers



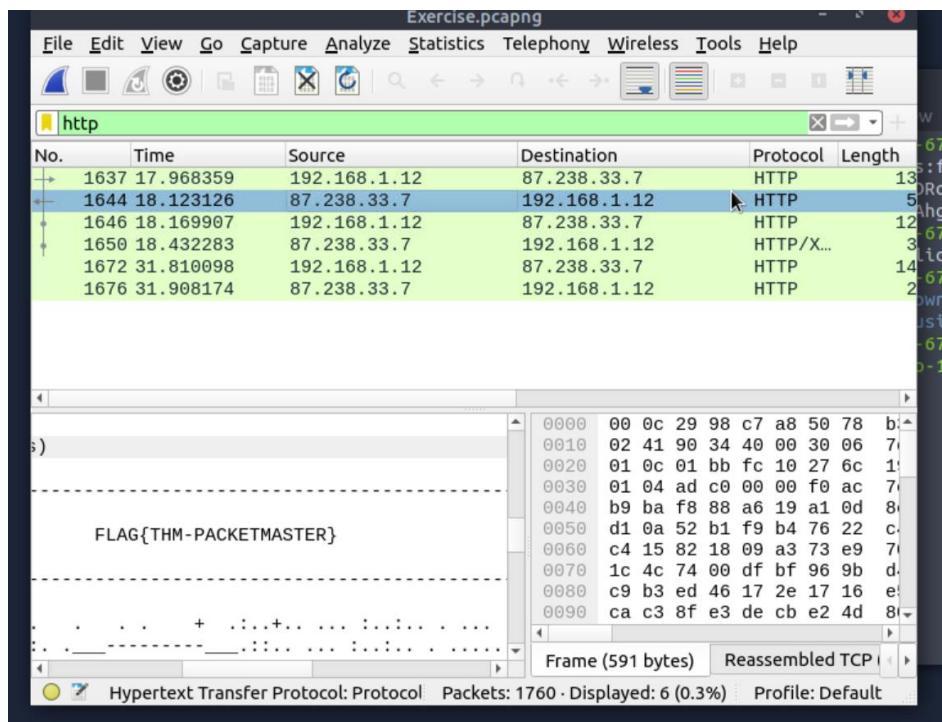
Answer: safebrowsing[.]google[.]com

4. Investigate the decrypted packets and find the flag. What is the flag?

- To find the flag, I filtered to http, since the file should be a readable file. I found a few results



- Here there is a text/plain file from packet 1644



Answer: FLAG{THM-PACKETMASTER}