CS315: Lab Assignment 13

B Siddharth Prabhu
200010003@iitdh.ac.in

11 April 2023

1 Answers to Part 1: Capturing Packets in a TLS Session

In this assignment, we explore TLS (Transport Layer Security), which is the successor to the now-deprecated Secure Sockets Layer (SSL). We'll investigate TLS by analyzing a Wireshark packet trace captured during the retrieval of a web page via HTTPS - a secure version of HTTP, which implements TLS on top of HTTP. We'll look at TLS's client-server handshaking protocol in some detail, since that's where most of the interesting action happens. Below is a screenshot of the packet trace captured:

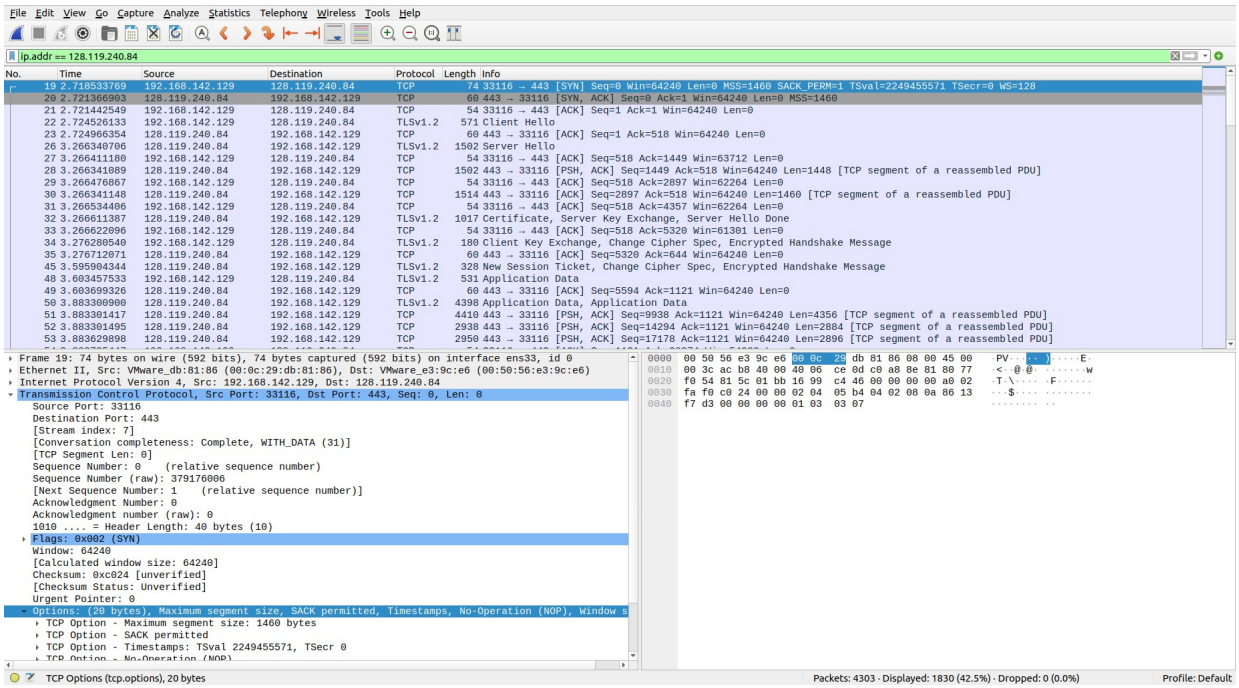

Figure 1: Packet Trace Obtained, with filter `ip.addr == 128.119.240.84`

2 Answers to Part 2: First Look at the Captured Trace

(1) What is the packet number in your trace that contains the initial TCP SYN message?

Based on the trace in Figure (1), the packet number with the initial TCP SYN message is **Packet Number 19**.

(2) Is the TCP connection set up before or after the first TLS message is sent from client to server?

The TCP connection is observed to be set up before the first TLS message is sent from client to server. Hence, TLS waits for the initial TCP connection to be set up. (Note that TLS works 'on top of' TCP.)

3 Answers to Part 3: TLS Handshake - Client Hello Message

In this part, we focus on the Client Hello message, as shown in Figure (2) below.

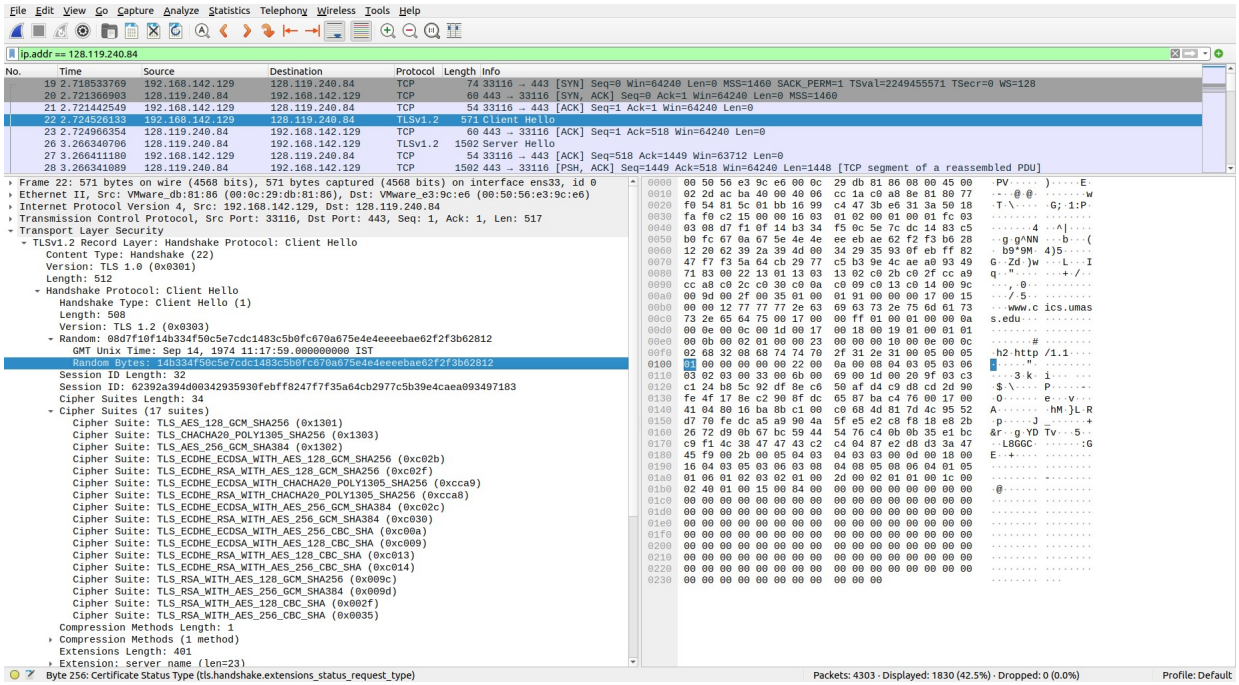

Figure 2: 'Client Hello' Packet Details

(1) What is the packet number in your trace that contains the TLS Client Hello message?

Based on the trace in Figure (2), the packet number with the TLS Client Hello message is **Packet Number 22**.

(2) What version of TLS is your client running, as declared in the Client Hello message?

As seen in Figure (2), the TLS version running in the client seems to be `TLSv1.2` inside the handshake protocol subfield, and TLSv1.0 mentioned inside the record layer. Most likely, it is v1.2 itself, since the rest of the communication between client and server occurs using TLSv1.2.

(3) How many cipher suites are supported by your client, as declared in the Client Hello message?

As seen in Figure (2), there are `17 Cipher Suites` declared in the Client Hello message.

(4) Your client generates and sends a string of "random bytes" to the server in the Client Hello message. What are the first two hexadecimal digits in the random bytes field of the Client Hello message? Enter the two hexadecimal digits.

As observed in Figure (2), the first two hexadecimal digits in the random bytes field of the Client Hello message are `14` .

(5) What is the purpose(s) of the "random bytes" field in the Client Hello message?

The "Random bytes" field in the Client Hello message is used to prevent replay attacks on the network. (It is a form of network attack in which valid data transmission is maliciously or fraudulently repeated or delayed.) These bytes are randomized for each TLS session, making it unique.

4 Answers to Part 4: TLS Handshake - Server Hello Message

Now, let's look at the Server Hello message, as seen in Figure (3) below:

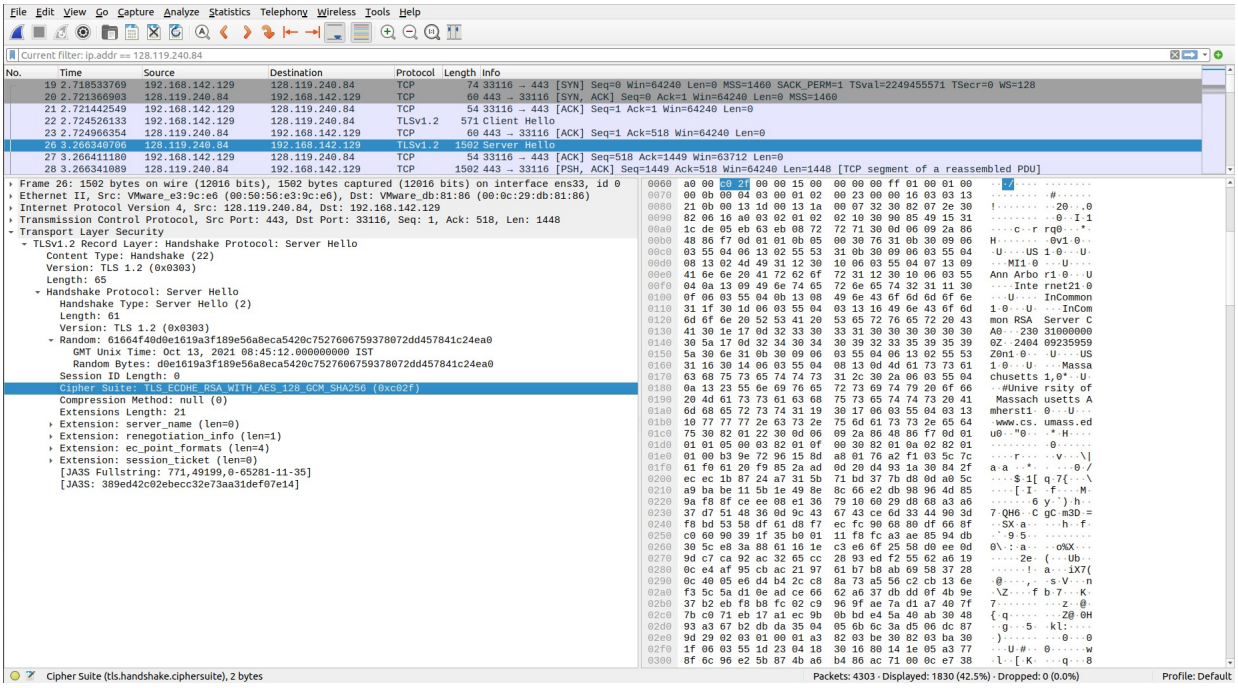

Figure 3: 'Server Hello' Packet Details

(1) What is the packet number in your trace that contains the TLS Server Hello message?

Based on the trace in Figure (2), the packet number with the TLS Client Hello message is **Packet Number 26**.

(2) Which cipher suite has been chosen by the server from among those offered in the earlier Client Hello message?

The cipher suite chosen by the server among those offered in the earlier Client Hello message is `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256` which has hex code (0xc02f).

(3) Does the Server Hello message contain random bytes, similar to how the Client Hello message contained random bytes? And if so, what is/are their purpose(s)?

Yes, there is a "Random bytes" field in the Server Hello message. It is used to prevent replay attacks on the network. (It is a form of network attack in which valid data transmission is maliciously or fraudulently repeated or delayed.)

Next, let's look at the TLS message where the Certificates are being sent. This is important for authentication. The packet details are in Figure (4) below.

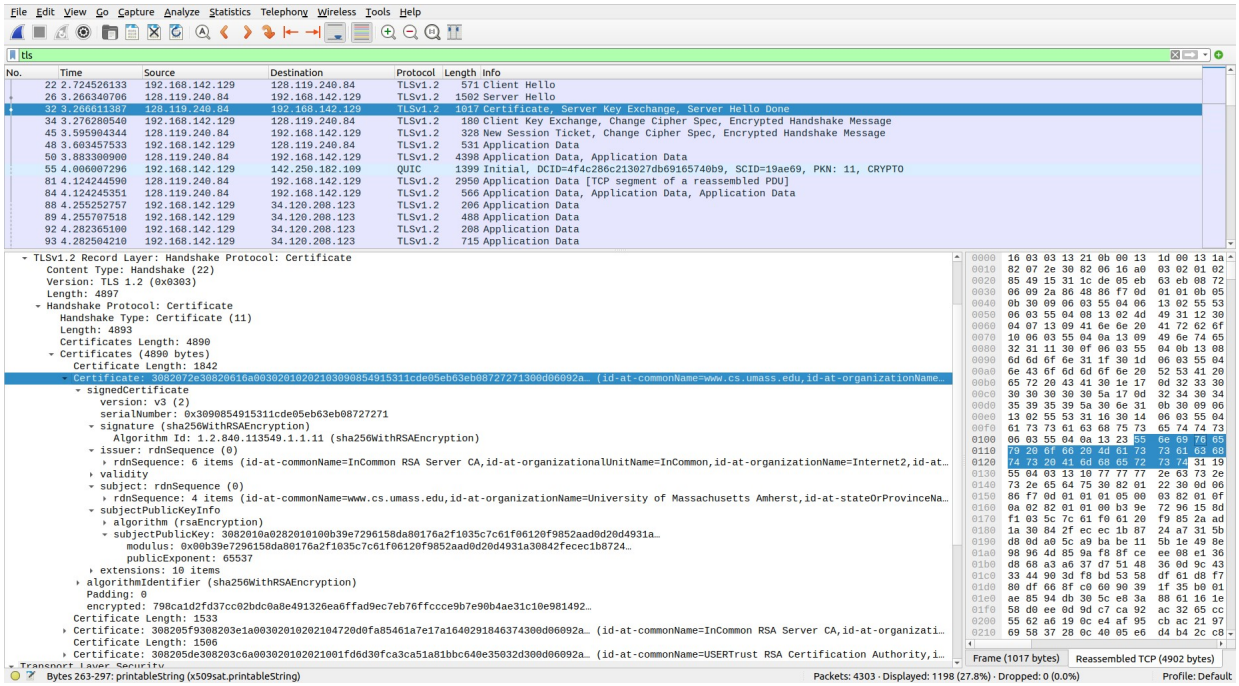

Figure 4: Certificate Packet Details

(4) What is the packet number in your trace for the TLS message part that contains the public key certificate for the www.cics.umass.edu server (actually the www.cs.umass.edu server)?

As seen in Figure (4), the packet number in the trace for the TLS message part that contains the public key certificate for the www.cics.umass.edu server is **Packet Number 32**.

(5) A server may return more than one certificate. If more than one certificate is returned, are all of these certificates for www.cs.umass.edu? If not all are for www.cs.umass.edu, then who are these other certificates for?

Three certificates are observed to have been returned by the server, as seen in Figure (4). Not all of them are for `www.cs.umass.edu` , but are for higher authorities that manage authentication certification for digital signatures, viz., `InCommon RSA Server CA` and `USERTrust RSA Certification Authority` .

(6) What is the name of the certification authority that issued the certificate for id-at-commonName=www.cs.umass.edu?

As seen in Figure (4), The certification authority that issued the certificate for the given site is `InCommon RSA Server CA` , which is seen to be one step up the hierarchy in terms of digital signatures.

(7) What digital signature algorithm is used by the CA to sign this certificate?

As seen in Figure (4), CA (Certification Authority) has used the SHA-256 algorithm with RSA encryption, listed with algorithm Id: `1.2.840.113549.1.1.11` (sha256WithRSAEncryption).

(8) What are the first four hexadecimal digits of the modulus of the public key being used by www.cics.umass.edu?

The first four hexadecimal digits of the modulus of the public key being used by `www.cics.umass.edu` are observed to be `00b3` .

(9) Look in your trace to find messages between the client and a CA to get the CA's public key information, so that the client can verify that the CA-signed certificate sent by the server is indeed valid and has not been forged or altered. Do you see such a message in your trace? If so, what is the number in the trace of the first packet sent from your client to the CA? If not, explain why the client did not contact the CA.

No, there doesn't seem to be such a message. It is possible that the client has cached the required information, and hence did not contact the CA.

(10) What is the packet number in your trace for the TLS message part that contains the Server Hello Done TLS record?

Without expanding any further details, we can observe from Figure (4) that the packet number for the TLS message part that contains the Server Hello Done TLS record is **Packet Number 32**.

5 Answers to Part 5: TLS Handshake - Wrapping up

After the exchange of Hello messages, the client responds to the TLS Server Hello message with its own public key information, a declaration (via a Change Cipher Spec record) that all further communication will be encrypted via the negotiated algorithm and key, and an Encrypted Handshake Message record that contains encrypted information (e.g., a cryptographic hash of all messages exchanged during this handshake) to prevent man-in-the-middle replay attacks.

(1) What is the packet number in your trace for the TLS message that contains the public key information, Change Cipher Spec, and Encrypted Handshake message, being sent from client to server?

As can be seen in Figure (4), The packet number in the trace for the TLS message that contains the public key information, Change Cipher Spec, and Encrypted Handshake message is **Packet Number 34**.

(2) Does the client provide its own CA-signed public key certificate back to the server? If so, what is the packet number in your trace containing your client's certificate?

No, the client does not appear to provide its own CA-signed public key certificate back to the server. This could be because the goal of the authentication in this case is for the server's authenticity, the client is not obligated to authenticate.

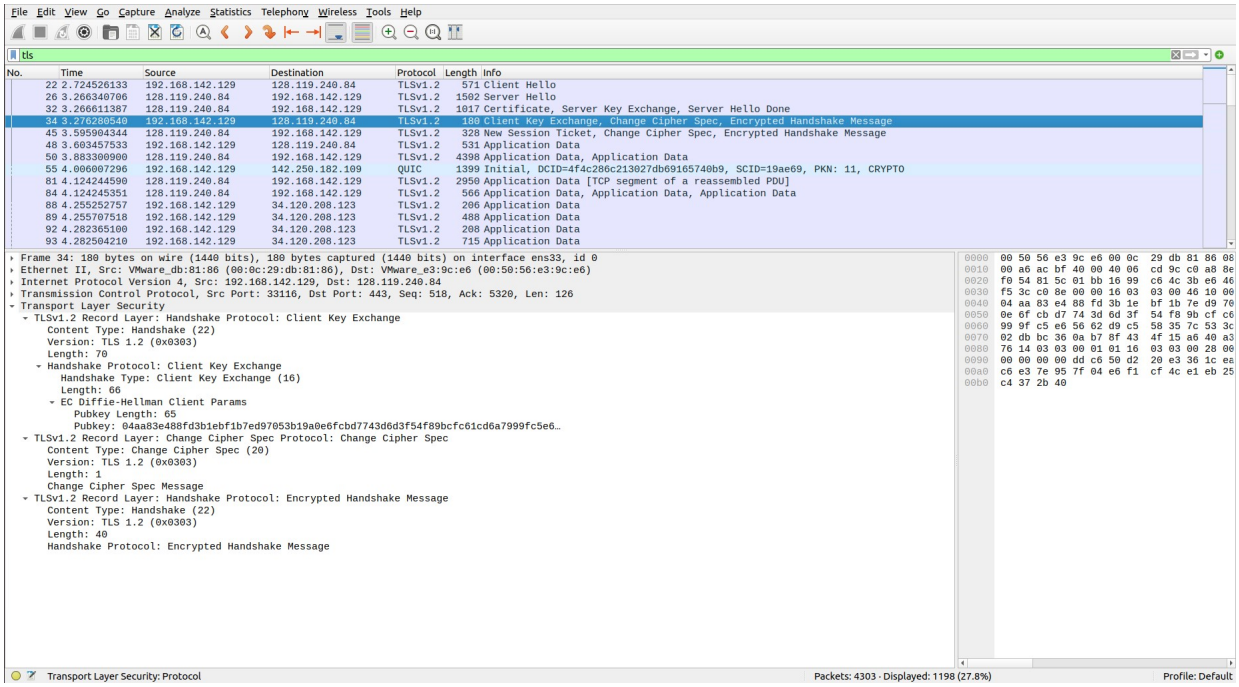

Figure 5: View of the trace

6 Answers to Part 6: Application Data

Once the TLS handshaking has completed, the encrypted application data can begin to flow over the HTTP-over-TLS-over-TCP connection. Of course, since this data is encrypted we can't actually examine the contents of these encrypted messages. But we shall answer a few questions about the encrypted traffic below.

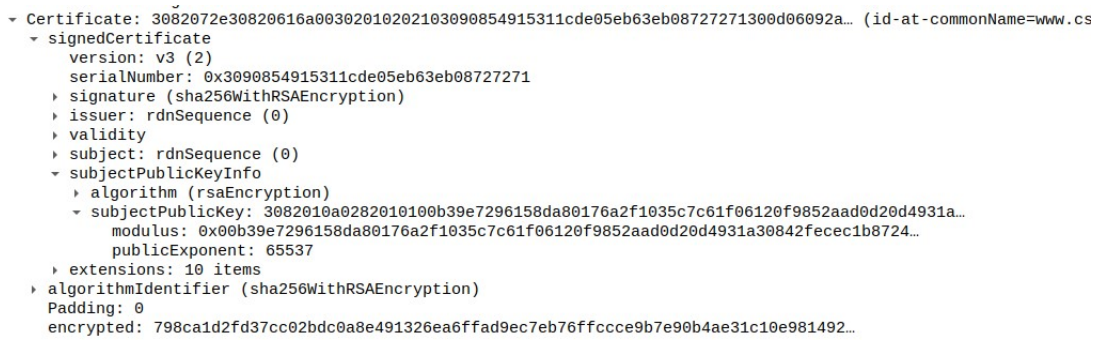

Figure 6: A look into the encryption

(1) What symmetric key cryptography algorithm is being used by the client and server to encrypt application data?

As seen in Figure (6), SHA-256 Algorithm is used with RSA Encrytption by the client and server to encrypt application data.

(2) In which of the TLS messages is this symmetric key cryptography algorithm finally decided and declared?

This symmetric key cryptography algorithm was finally decided and declared in the TLS **Server Hello Done** message.

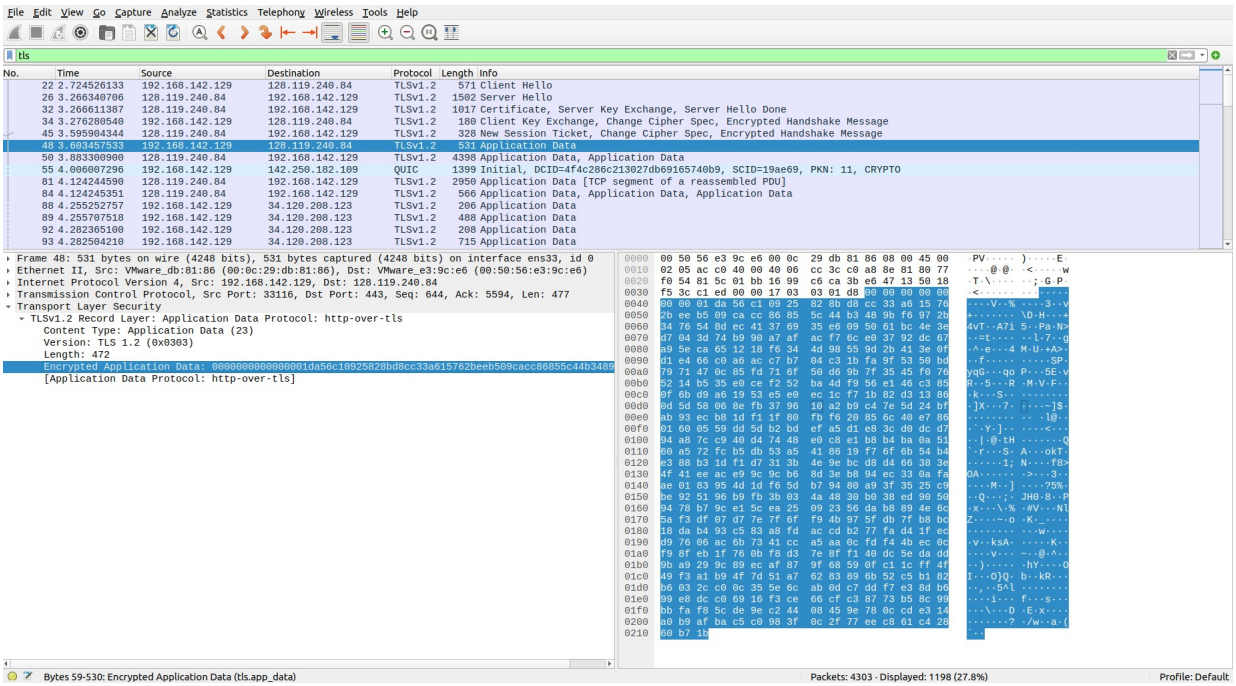

Figure 7: Application Data

(3) What is the packet number in your trace for the first encrypted message carrying application data from client to server?

The first encrypted message carrying application data from client to server is **Packet Number 48**.

(4) What do you think the content of this encrypted application-data is, given that this trace was generated by fetching the homepage of www.cics.umass.edu?

Since that particular message is sent from client to the server, and HTTP is working on top of TLS, considering the TCP and TLS starting earlier, this encrypted application data must contain the **HTTP GET Request**.

(5) What packet number contains the client-to-server TLS message that shuts down the TLS connection?

Since TLS messages are encrypted in our Wireshark traces, we can't actually look inside a TLS message and so we'll have to make an educated guess here. A likely possibility is shown in Figure (8), where **Packet Number 3785** is shown. It has `FIN` flag set in its TCP part, and the expert info hints that the packet may be shutting down the connection. This seems to go from server to the client, but it may be in response to another message sent from client to server.

Packet Number 3806 is seen to be a strong possibility, considering Figure (9).

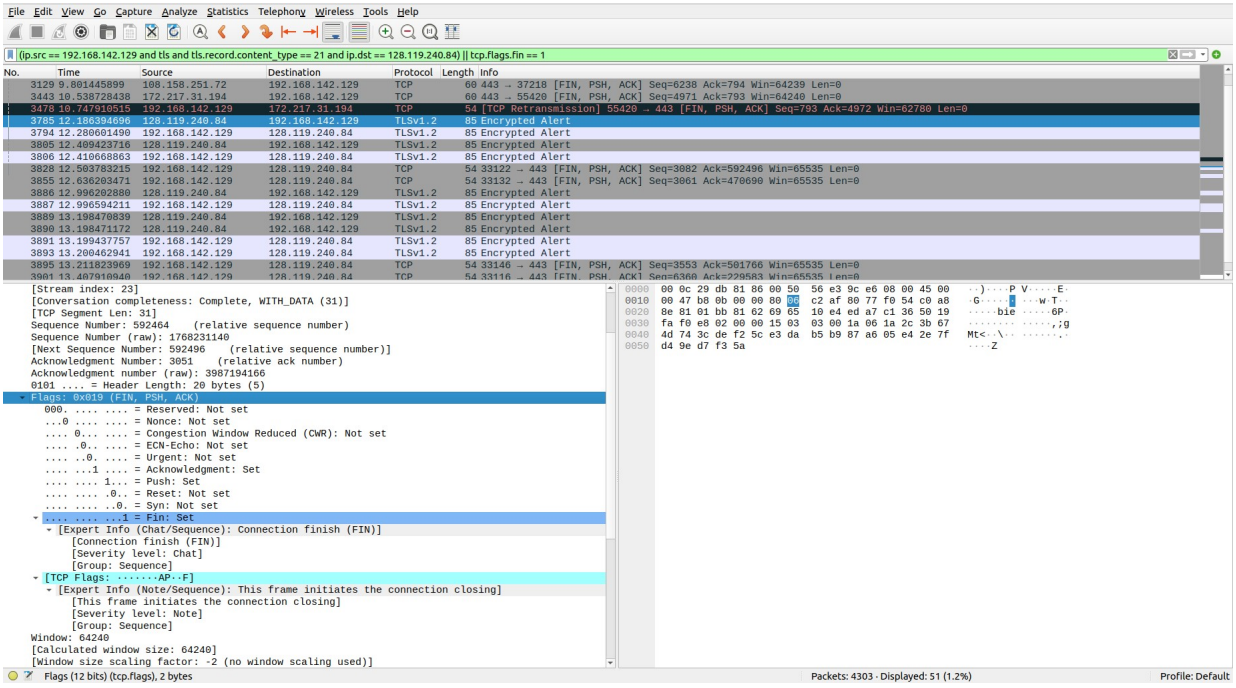

Figure 8: Possible TLS connection shutdown initiation

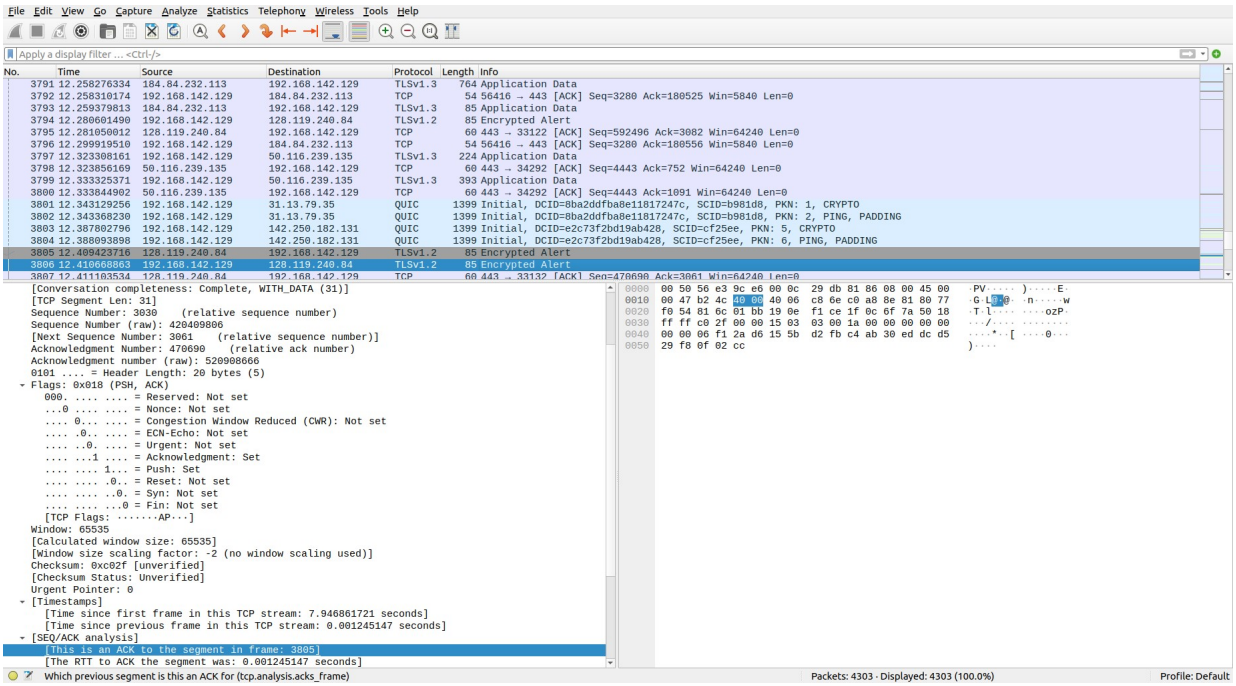

Figure 9: Possible TLS connection shutdown initiation