

### **4.8.Experiment no.08**

**Title:**

To study the SSL protocol by capturing the packets using Wireshark tool while visiting any SSL secured website (banking, e-commerce etc.)

**Theory:**

**SSL, or Secure Sockets Layer**, is an encryption-based Internet security protocol. It was first developed by Netscape in 1995 for the purpose of ensuring privacy, authentication, and data integrity in Internet communications. SSL is the predecessor to the modern TLS encryption used today.

**How does SSL/TLS work?**

In order to provide a high degree of privacy, SSL encrypts data that is transmitted across the web. This means that anyone who tries to intercept this data will only see a garbled mix of characters that is nearly impossible to decrypt.

SSL initiates an authentication process called a handshake between two communicating devices to ensure that both devices are really who they claim to be.

SSL also digitally signs data in order to provide data integrity, verifying that the data is not tampered with before reaching its intended recipient.

There have been several iterations of SSL, each more secure than the last. In 1999 SSL was updated to become TLS.

**Why is SSL/TLS important?**

Originally, data on the Web was transmitted in plaintext that anyone could read if they intercepted the message. For example, if a consumer visited a shopping website, placed an order, and entered their credit card number on the website, that credit card number would travel across the Internet unconcealed.

SSL was created to correct this problem and protect user privacy. By encrypting any

data that goes between a user and a web server, SSL ensures that anyone who intercepts the data can only see a scrambled mess of characters. The consumer's credit card number is now safe, only visible to the shopping website where they entered it.

SSL also stops certain kinds of cyber attacks: It authenticates web servers, which is important because attackers will often try to set up fake websites to trick users and steal data. It also prevents attackers from tampering with data in transit, like a tamper-proof seal on a medicine container.

### **Are SSL and TLS the same thing?**

SSL is the direct predecessor of another protocol called TLS (Transport Layer Security). In 1999 the Internet Engineering Task Force (IETF) proposed an update to SSL. Since this update was being developed by the IETF and Netscape was no longer involved, the name was changed to TLS. The differences between the final version of SSL (3.0) and the first version of TLS are not drastic; the name change was applied to signify the change in ownership.

Since they are so closely related, the two terms are often used interchangeably and confused. Some people still use SSL to refer to TLS, others use the term "SSL/TLS encryption" because SSL still has so much name recognition.

### ***SSL PROTOCOL Wireshark :***

#### **Step 1: Open a Trace**

##### ***1. Open the Wireshark trace***

You should see the following trace.

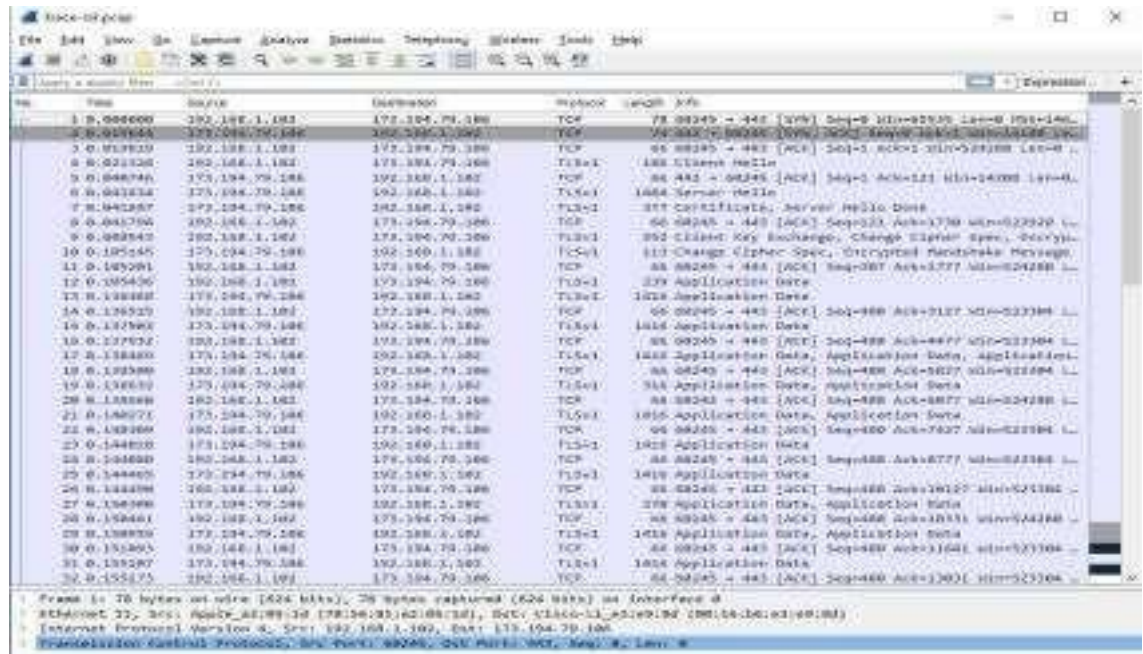


Figure 1: Trace of “HTTPS” traffic

1

## Step 2: Inspect the Trace

Now we are ready to look at the details of some “SSL” messages.

2. To begin, enter and apply a display filter of “ssl”. (see below)

This filter will help to simplify the display by showing only SSL and TLS messages. It will exclude other TCP segments that are part of the trace, such as Acks and connection open/close.

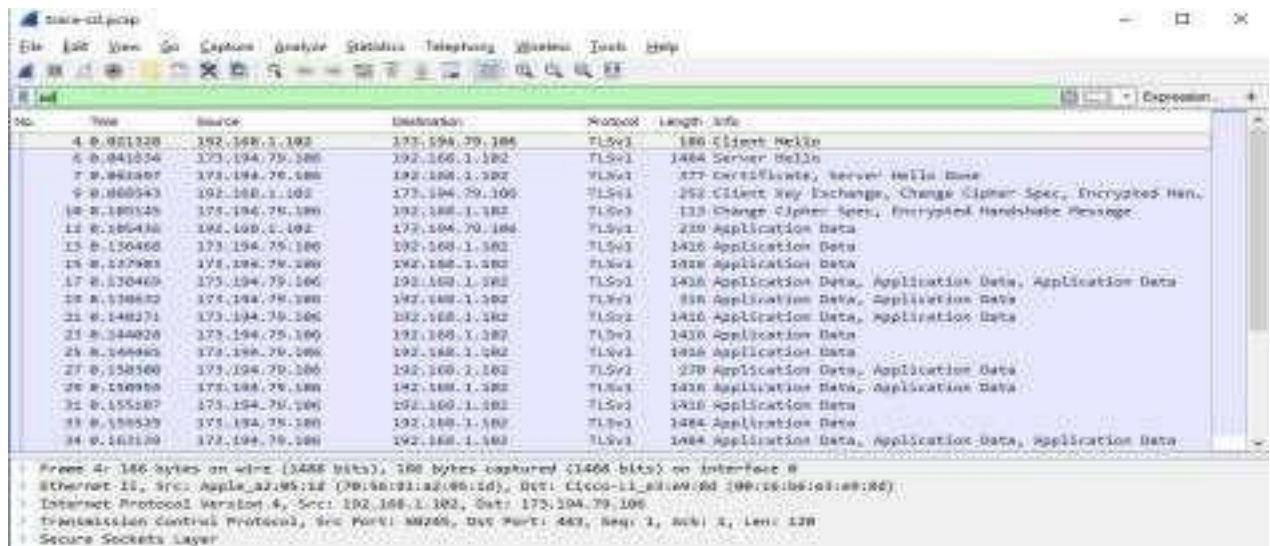
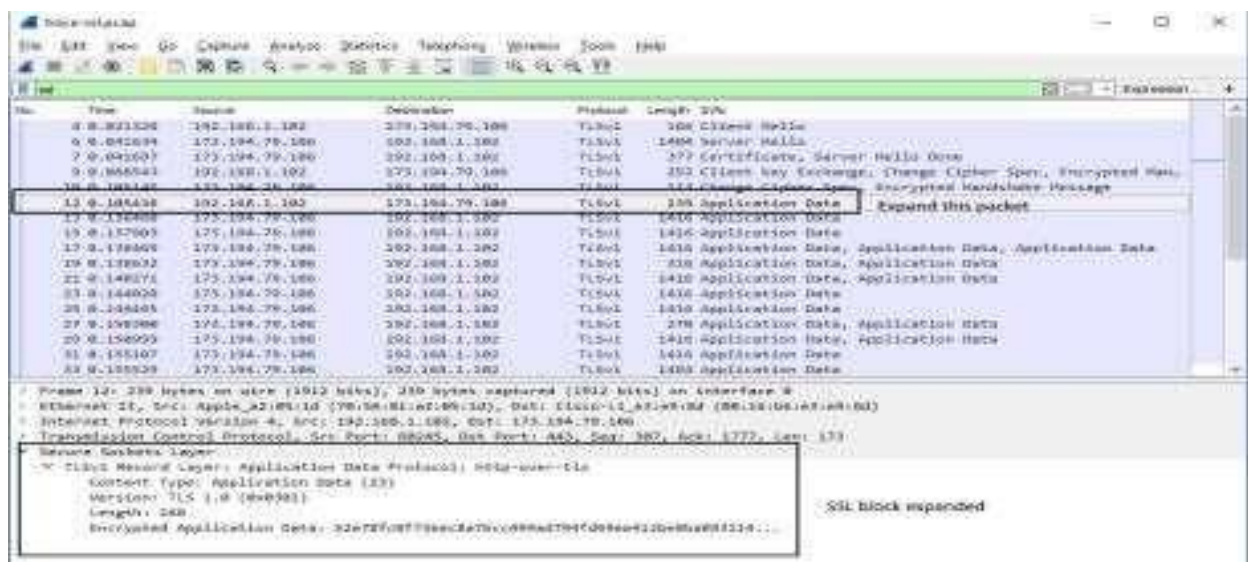


Figure 2: Trace of “SSL” traffic showing the details of the SSL header

3. Select a TLS message somewhere in the middle of your trace for which the Info reads “Application Data” & expand its Secure Sockets Layer block (by using the “+” expander or icon). For instance, packet #12 (see below).



2

Application Data is a generic TLS message carrying contents for the application, such as the web page. It is a good place for us to start looking at TLS messages.

The lower layer protocol blocks are TCP and IP because SSL runs on top of TCP/IP. The SSL layer contains a “TLS Record Layer”. This is the foundational sublayer for

TLS. All messages contain records. Expand this block to see its details. Each record starts with a Content Type field. This tells us what is in the contents of the record. Then comes a Version identifier. It will be a constant value for the SSL connection. It is followed by a Length field giving the length of the record. Last comes the contents of the record. Application Data records are sent after SSL has secured the connection, so the contents will show up as encrypted data. To see within this block, we could configure Wireshark with the decryption key. This is possible, but outside of our scope. Note that, unlike other protocols we will see such as DNS, there may be multiple records in a single message. Each record will show up as its own block. Look at the Info column, and you will see messages with more than one block.

The Content-Type for a record containing “Application Data” is 23. The version constant used in this trace is 0x0301 which represents TLS 1.0. The Length covers only the payload of the Record Layer.

### Step 3: The SSL Handshake

An important part of SSL is the initial handshake that establishes a secure connection. The handshake proceeds in several phases. There are slight differences for different versions of TLS and depending on the encryption scheme that is in use. The usual outline for a brand-new connection is:

- a. Client (the browser) and Server (the web server) both send their Hellos
- b. Server sends its certificate to Client to authenticate (and optionally asks for Client Certificate)
- c. Client sends keying information and signals a switch to encrypted data.
- d. Server signals a switch to encrypted data.
- e. Both Client and Server send encrypted data.
- f. An Alert is used to tell the other party that the connection is closing.

Note that there is also a mechanism to resume sessions for repeat connections between the same client and server to skip most of steps b and c. However, we will not study session resumption.

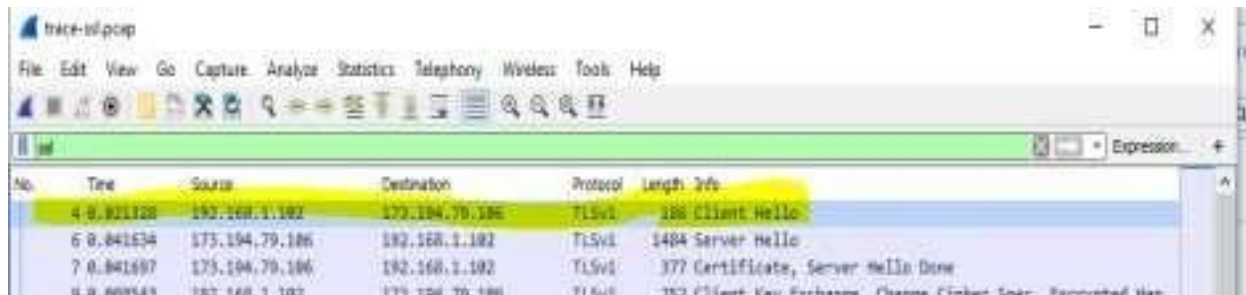
### Hello Messages

*Next we will find and inspect the details of the Client Hello and Server Hello messages, including expanding the Handshake protocol block within the TLS Record.* For these initial messages, an encryption scheme is not yet established so the contents of the record are visible to us. They contain details of the secure connection setup in a

Handshake protocol format.

3

*4. Select packet #4, which is a TLS Client Hello message*



No.	Time	Source	Destination	Protocol	Length	Info
4	0.021128	192.168.1.102	192.168.1.106	TLSv1	188	Client Hello
6	0.041534	192.168.1.106	192.168.1.102	TLSv1	1484	Server Hello
7	0.041697	192.168.1.106	192.168.1.102	TLSv1	377	Certificate, Server Hello Done
8	0.000141	192.168.1.102	192.168.1.106	TLSv1	252	Client Key Exchange, Change Cipher Spec, Finished

We can see several important fields here worth mentioning. First, the time (GMT seconds since midnight Jan 1, 1970) and random bytes (size 28) are included. This will be used later in the protocol to generate our symmetric encryption key. The client can send an optional session ID to quickly resume a previous TLS connection and skip portions of the TLS handshake. Arguably the most important part of the ClientHello message is the list of cipher suites, which dictate the key exchange algorithm, bulk encryption algorithm (with key length), MAC, and a pseudo-random function. The list should be ordered by client preference. The collection of these choices is a “cipher suite”, and the server is responsible for choosing a secure one it supports or return an error if it doesn’t support any. The final field specified in the specification is for compression methods. However, secure clients will advertise that they do not support compression (by passing “null” as the only algorithm) to avoid the CRIME attack.

Finally, the ClientHello can have a number of different extensions. A common one is server\_name, which specifies the host name the connection is meant for, so web servers hosting multiple sites can present the correct certificate.

*5. Select packet #6, which is a TLS Server Hello message*

The session ID sent by the server is 32 bytes long. This identifier allows later resumption of the session with an abbreviated handshake when both the client and server indicate the same value. In our case, the client likely sent no session ID as there was nothing to resume (see below)



No.	Time	Source	Destination	Protocol	Length	Info
4	0.021328	192.168.1.102	173.194.79.106	TLSv1	186	Client Hello
5	0.041634	173.194.79.106	192.168.1.102	TLSv1	1404	Server Hello
7	0.041697	173.194.79.106	192.168.1.102	TLSv1	372	Certificate, Server Hello Done

```

Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 83
  Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 81
    Version: TLS 1.0 (0x0301)
    Random: 561778d3d52d55ed28e072f638f8a51e9724d66ef5f1376...
      GMT Unix Time: Jul 31, 2012 07:18:59.000000000 GMT Daylight Time
      Random Bytes: d52d55ed28e072f638f8a51e9724d66ef5f13769d3a52e0...
    Session ID Length: 32
    Session ID: 8530bdac95116ccb343798b36cb2fd79c1e278cba1af4145...

```

4

The Cipher method chosen by the Server is *TLS\_RSA\_WITH\_RC4\_128\_SHA* (0x0005). The Client will list the different cipher methods it supports, and the Server will pick one of these methods to use.

```

Session ID Length: 32
Session ID: 8530bdac95116ccb343798b36cb2fd79c1e278cba1af4145...
Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
Compression Method: null (0)
Extensions Length: 9

```

### Certificate Messages

- Next, find and inspect the details of the Certificate message including expanding the Handshake protocol block within the TLS Record (see below for expansion of packet #7).

As with the Hellos, the contents of the Certificate message are visible because an encryption scheme is not yet established. It should come after the Hello messages.

Note it is the server that sends a certificate to the client, since it is the browser that wants to verify the identity of the server. It is also possible for the server to request certificates from the client, but this behavior is not normally used by web applications.

A Certificate message will contain one or more certificates, as needed for one party to verify the identity of the other party from its roots of trust certificates. You can inspect those certificates in your browser.

5

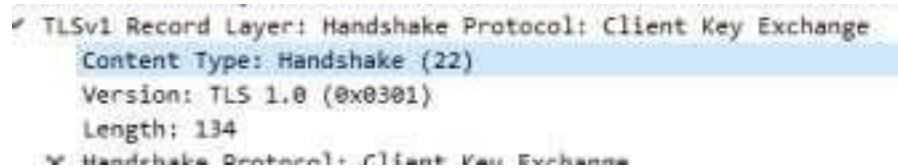
7. Find and inspect the details of the Client Key Exchange and Change Cipher messages i.e. packet #9 (see below)

The key exchange message is sent to pass keying information so that both sides will have the same secret session key. The change cipher message signals a switch to a new encryption scheme to the other party. This means that it is the last unencrypted

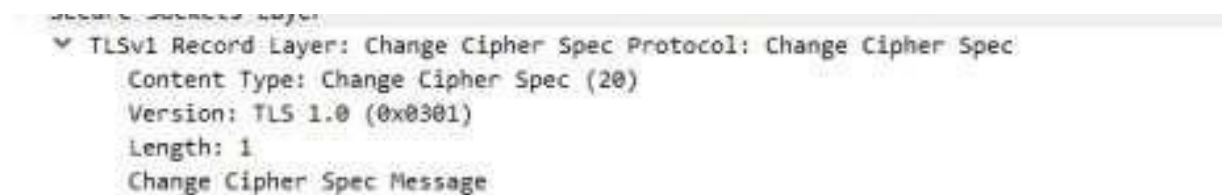


message sent by the party.

Note how the Client Key Exchange has a Content-Type of 22, indicating the Handshake protocol. This is the same as for the Hello and Certificate messages, as they are part of the Handshake protocol.



The Change Cipher Spec message has a Content-Type of 20, indicating the Change Cipher Spec protocol (see packet #10 – see below).



That is, this message is part of its own protocol and not the Handshake protocol. Both sides send the Change Cipher Spec message immediately before they switch to sending encrypted contents. The message is an indication to the other side. The contents of the Change Cipher Spec message are simply the value 1 as a single byte. Actually, it is the value “1” encrypted under the current scheme, which uses no encryption for the handshake so that we can see it.

### Alert Message

8. Finally, find and inspect the details of an Alert message at the end of the trace (packet #42).

The Alert message is sent to signal a condition, such as notification that one party is closing the connection. You should find an Alert after the Application Data messages that make up the secure web fetch.



Note, the Content-Type value is 21 for Alert. This is a new protocol, different from the Handshake, Change Cipher Spec and Application Data values that we have already seen.

The alert is encrypted; we cannot see its contents. Wireshark also describes the message as an “Encrypted Alert”. Presumably it is a “close\_notify” alert to signal that the connection is ending, but we cannot be certain.

### Conclusion:

Hence we had studied the SSL protocol by capturing the packets using Wireshark tool while visiting any SSL secured website (banking, e-commerce etc.)

### Outcomes:

Retrieve SSL protocol by capturing the packets using Wireshark.

### FAQs:

- 1) What is SSL Protocol?
- 2) On which layer SSL Protocol works?