

Assignment No .

12

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

Theory :

Wireshark: This uses Wireshark to capture or examine a packet trace. A packet trace is a record of traffic at some location on the network, as if a snapshot was taken of all the bits that passed across a particular wire. The packet trace records a timestamp for each packet, along with the bits that make up the packet, from the low-layer headers to the higher-layer contents. Wireshark runs on most operating systems, including Windows, Mac and Linux. It provides a graphical UI that shows the sequence of packets and the meaning of the bits when interpreted as protocol headers and data. The packets are color-coded to convey their meaning, and Wireshark includes various ways to filter and analyze them to let you investigate different aspects of behavior. It is widely used to troubleshoot networks. You can download Wireshark from www.wireshark.org if it is not already installed on your computer. We highly recommend that you watch the short, 5 minute video “Introduction to Wireshark” that is on the site.

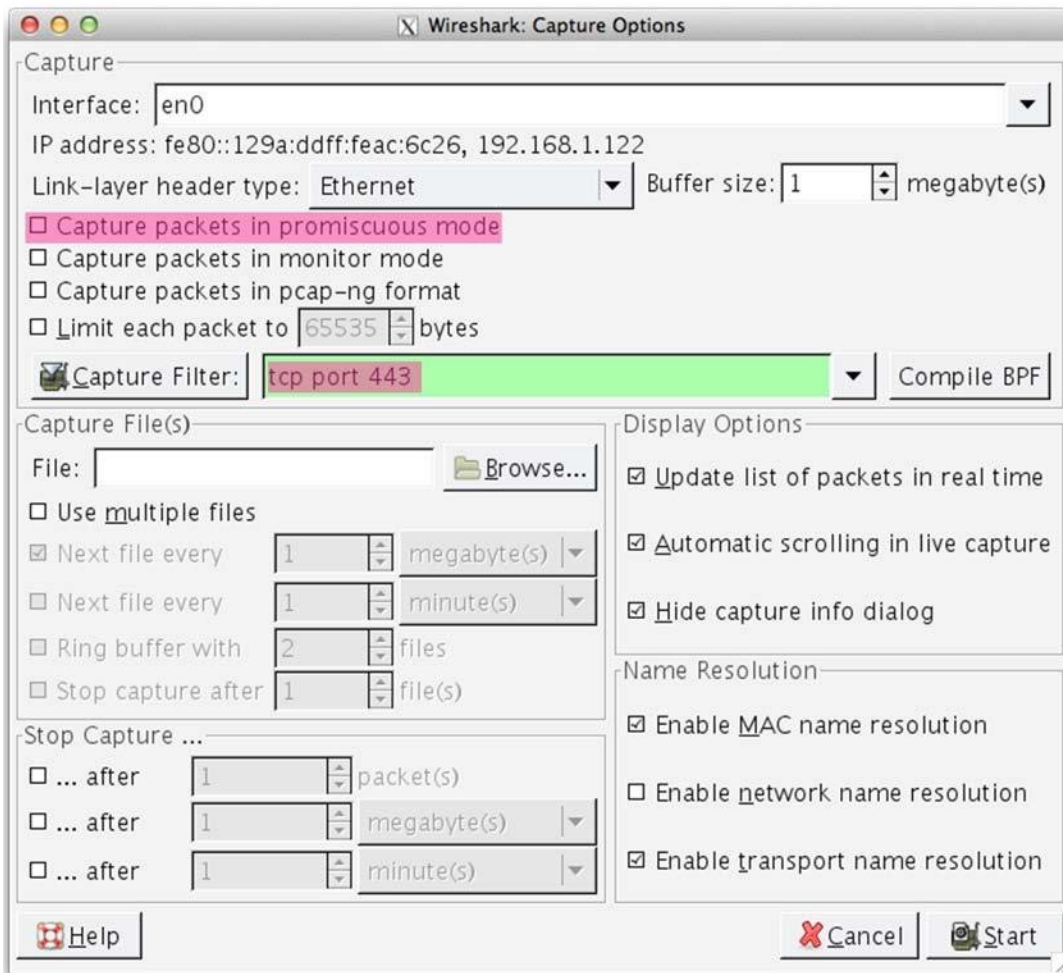
wget / curl: This lab uses wget (Linux and Windows) and curl (Mac) to fetch web resources. wget and curl are command-line programs that let you fetch a URL. Unlike a web browser, which fetches and executes entire pages, wget and curl give you control over exactly which URLs you fetch and when you fetch them. Under Linux, wget can be installed via your package manager. Under Windows, wget is available as a binary; look for download information on <http://www.gnu.org/software/wget/>. Under Mac, curl comes installed with the OS. Both have many options (try “wget --help” or “curl --help” to see) but a URL can be fetched simply with “wget URL” or “curl URL”.

Step 1: Capture a Trace

Proceed as follows to capture a trace of SSL traffic; alternatively, you may use a supplied trace. The easiest way for us to produce SSL traffic is to fetch web pages with HTTPS. Any URL with HTTPS will do, e.g., <https://www.google.com>. However, web browsers have complex behaviors that can lead to a complex trace. Instead, we will use wget / curl to fetch a single HTTPS resource.

1. Close all unnecessary browser tabs and windows. Browsing web sites may generate HTTPS traffic. We want to minimize browser activity so that we capture only the intended DNS traffic. (You may want to redo your trace if you see unexpected HTTPS traffic due to background processes on your computer.)
2. Launch Wireshark and start a capture with a filter of “tcp port 443”. We use this filter because there is no shorthand for SSL, but SSL is normally carried on port 443 in the case of secure web pages. Your capture window should be similar to the one pictured below, other than our highlighting. Select the interface from which to capture as the main wired or wireless interface

used by your computer to connect to the Internet. If unsure, guess and revisit this step later if your capture is not successful. Uncheck “capture packets in promiscuous mode”. This mode is useful to overhear packets sent to/from other computers on broadcast networks. We only want to record packets sent to/from your computer. Leave other options at their default values. The capture filter, if present, is used to prevent the capture of other traffic your computer may send or receive. On Wireshark 1.8, the capture filter box is present directly on the options screen, but on Wireshark 1.9, you set a capture filter by double-clicking on the interface.



3.

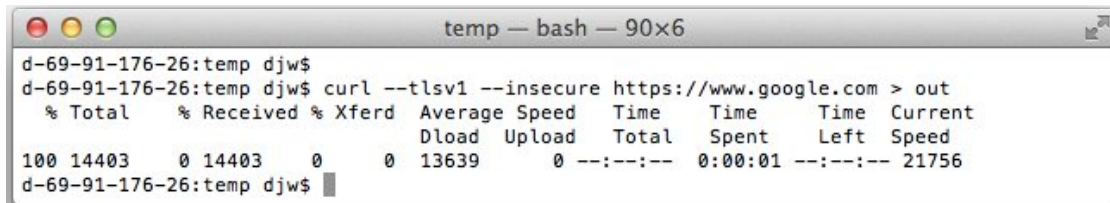
ol. This
to the

various versions of SSL and TLS as SSL 3. We ask you to explicitly select TLS 1.0 because there are several variations of the protocol that operate with slight differences. As part of the fetch, wget and curl may attempt to check the validity of the certificate to authenticate the server. This check will fail if your installation is not configured with an appropriate set of certificates. To avoid this hitch, you can turn off certificate checking with a command-line option.

Examples of the full command are:

```
curl --tlsv1 --insecure https://www.google.com > out
wget --secure-protocol=TLsv1 --no-check-certificate https://www.google.com
```

Note the “long dash” is two hyphens. We give an example fetch in the figure below. Besure to check that your HTTPS fetch succeeds, and if not try with another web page.



```
temp — bash — 90x6
d-69-91-176-26:temp djw$
d-69-91-176-26:temp djw$ curl --tlsv1 --insecure https://www.google.com > out
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload    Total   Spent    Left   Speed
100 14403    0 14403    0    0 13639    0 --:--:--  0:00:01 --:--:-- 21756
d-69-91-176-26:temp djw$
```

Figure 2: Fetching an HTTPS resource with curl

4. Stop the capture after you have completed the HTTPS fetch. We are ready to inspect it.

Step 2: Inspect the Trace

Now we are ready to look at the details of some “SSL” messages. There are several different versions of SSL and TLS that are in widesread use, including SSL version 2, SSL version 3, and TLS version 1; TLS is the open standard version of the protocol and TLS 1.0 comes after SSL 3.0. Following common practice, we will informally refer to all of them as “SSL”. A complication for this lab is that they all behave slightly differently. To minimize variation, we asked you to gather TLS 1.0 traffic as part of the steps above.

To begin, enter and apply a display filter of “ssl”. 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. Your display should be similar to the one shownin our figure.

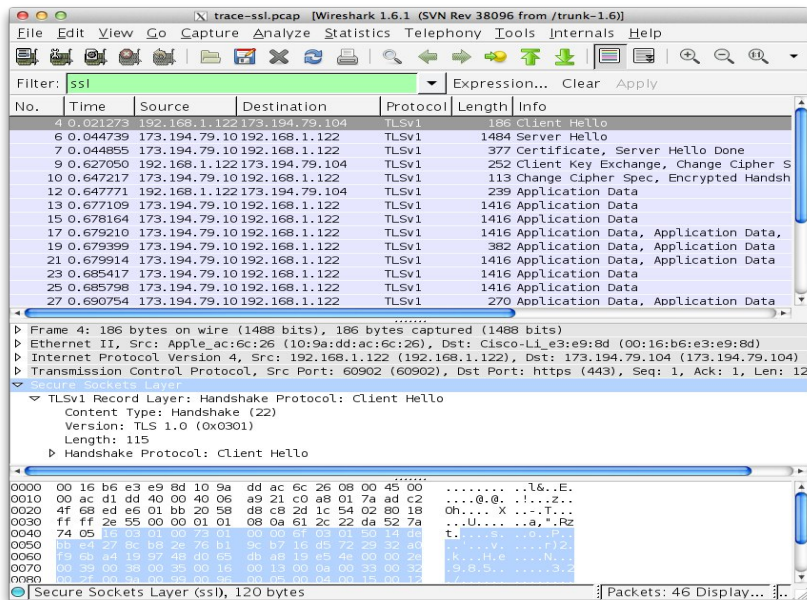


Figure 3: Trace of “SSL” traffic showing the details of the SSLheader

Select a TLS message somewhere in the middle of your trace for which the Info reads “Application Data”, and expand its Secure Sockets Layer block (by using the “+” expander or icon). 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.

Look for the following protocol blocks and fields in the message:

- 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.

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 ClientCertificate)
- 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.

Overall Handshake

To get a sense of the overall SSL connection behavior, draw a timeline showing and naming the SSL messages sent between the client and the server. As usual, draw vertical lines to represent the client and the server, with time running down the page. To work out the names of the SSL messages you should include, use the description given in the Info column of your Wireshark display. You should cover the entire connection, but elide most of the encrypted SSL messages in the middle of the connection (Application Data) to keep the figure simple. There is no need for you to understand the details at this stage; our next step is to look at each of the phases of the

connection in more detail.

Hello Messages

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.

Answer the following questions:

1. How long in bytes is the random data in the Hellos? Both the Client and Server include this random data (a nonce) to allow the establishment of session keys.
2. How long in bytes is the session identifier sent by the server? 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.
3. What Cipher method is chosen by the Server? Give its name and value. The Client will list the different cipher methods it supports, and the Server will pick one of these methods to use.

Certificate Messages

Next, find and inspect the details of the Certificate message, including expanding the Handshake protocol block within the TLS Record. 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.

Answer the following questions:

4. Who sends the Certificate, the client, the server, or both? A certificate is sent by one party to let the other party authenticate that it is who it claims to be. Based on this usage, you should be able to guess who sends the certificate and check the messages in your trace.

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.

Client Key Exchange and Change Cipher Messages

Find and inspect the details of the Client Key Exchange and Change Cipher messages, expanding their various details. 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.

Alert Message

Finally, find and inspect the details of an Alert message at the end of the trace. 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.

- Remove the display filter for “ssl” to see the other TCP segments that are part of the connection. In this way you will see when TCP connection setup/teardown happens relative to the SSL handshake and close_notify alert.
- Look to see what encryption schemes are requested by the client and selected by the server. Do they all provide good security, or are some much better than others?
- Explore the session resumption mechanism that abbreviates the SSL handshake. It is widely used in practice to speed up SSL connections to web servers.
- Capture a trace of HTTPS traffic generated by using your browser. It should be consistent with the broad protocol features we studied, but will likely exhibit a wider variety of behaviors.
- Try to fetch secure web resources using SSL versions 2 and 3 as well as TLS version 1, and look to see the differences. You can also study SSL version negotiation: it is common practice to start with an SSL version 2 message and negotiate up to SSL version 3 or TLS version 1.
- Configure Wireshark with a key to let you look inside encrypted SSL messages. You can read on the web how to do this. Once decrypted, you will be able to observe the HTTP protocol running on top of SSL, as well as the details of other SSL messages such as Alerts.

Observation :

(1) Identify a secure server which uses TLS for secure transactions. It appears with a green padlock in the URL window of the browser. Some of the websites may appear with a grey padlock. There is not much practical difference except these websites do not use extended validation certificates. For this worksheet, a website which comes up with a green padlock is recommended.

(2) For this worksheet, hsbc.co.in website of HSBC bank is selected. It comes up with a green padlock when personal internet banking log on is pressed from its main page (3) Identify the IP address of the website running commands like `ping` or `tracert` from command prompt (windows) or `run -> cmd` (Linux terminal). The IP address for HSBC personal banking log on is 203.112.92.107 as retrieved below. This IP address will be used to identify the TLS server in the Wireshark captured packets

4) Identify the IP address of your PC/laptop from where you are accessing this web server using `ipconfig` command from command prompt (or, `ifconfig` from the Linux terminal). The IP address of this experimental laptop is 192.168.1.2 as retrieved below. This IP address will be used to identify the TLS client in the Wireshark captured packets

Conclusion : Thus we have studied SSL protocol using Wireshark successfully

Output:

The screenshot shows the Wireshark interface with a packet capture of an SSL/TLS session. The packet list on the left shows several packets, including the TLS handshake (1421-1447) and application data (1452-1521). The packet details pane on the right shows the selected packet (1452) as a TLSv1.2 record layer, containing application data (23 bytes) and a protected payload (511 bytes). The packet bytes pane on the right shows the raw data of the selected packet, including the TLS record structure and the encrypted application data.

No.	Time	Source	Destination	Protocol	Length	Info
1421	244.799794	44.236.232.139	10.30.74.125	TLSv1.2	159	Application Data, Application Data, Encrypted Alert
1441	248.032968	34.160.144.191	10.30.74.125	TLSv1.2	129	Application Data
1441	248.318285	34.117.237.239	10.30.74.125	TLSv1.2	122	Application Data
1447	249.374335	34.120.115.102	10.30.74.125	TLSv1.2	122	Application Data
1452	250.046947	10.30.75.94	8.8.4.4	QUIC	1292	Initial, DCID=d4ec6f53bc86983f, PKN: 1, PADDING, CRYPTO, PADDING, CRYPTO, PADDING, CRYPTO, PADDING, CRY...
1452	250.114897	8.8.4.4	10.30.75.94	QUIC	1292	Protected Payload (KP0)
1453	250.133372	10.30.75.94	172.217.167.174	QUIC	1292	Initial, DCID=fbcabfdb1b27db42, PKN: 1, CRYPTO, PING, PADDING, CRYPTO, CRYPTO, CRYPTO, CRYPTO, CRYPTO, ...
1453	250.223252	172.217.167.174	10.30.75.94	QUIC	1292	Protected Payload (KP0)
1453	250.331707	34.120.237.76	10.30.74.125	TLSv1.2	139	Application Data
1470	253.544491	10.30.75.94	52.213.210.125	TLSv1.2	132	Application Data
1470	253.544535	10.30.75.94	52.213.210.125	TLSv1.2	100	Application Data
1470	253.544554	10.30.75.94	52.213.210.125	TLSv1.2	435	Application Data
1470	253.688105	52.213.210.125	10.30.75.94	TLSv1.2	100	Application Data
1470	253.690392	52.213.210.125	10.30.75.94	TLSv1.2	248	Application Data
1475	253.908562	10.30.75.94	142.250.183.99	QUIC	1292	Initial, DCID=4f4d8344e0664f0c, PKN: 1, PING, CRYPTO, CRYPTO, PADDING, PING, PADDING, PING, CRYPTO, CRY...
1475	253.934363	34.120.208.123	10.30.74.125	TLSv1.2	139	Application Data
1475	254.007494	142.250.183.99	10.30.75.94	QUIC	1292	Protected Payload (KP0)
1481	255.331836	34.120.237.76	10.30.74.125	TLSv1.2	122	Application Data
1503	258.935092	34.120.208.123	10.30.74.125	TLSv1.2	122	Application Data
1504	259.200795	108.158.43.85	10.30.75.94	TLSv1.2	93	Application Data
1510	260.597890	108.159.59.176	10.30.75.94	TLSv1.2	93	Application Data
1510	260.597891	18.66.40.93	10.30.75.94	TLSv1.2	93	Application Data
1521	262.619738	151.101.153.108	10.30.74.125	TLSv1.2	97	Encrypted Alert
1521	262.632537	151.101.153.108	10.30.74.125	TLSv1.2	97	Encrypted Alert

Packet 1452 details:

- Internet Protocol Version 4, Src: 104.81.24.166, Dst: 10.30.75.94
- Transmission Control Protocol, Src Port: 443, Dst Port: 50320, Seq: 1, Ack: 1, Len: 516
- Transport Layer Security
 - TLSv1.2 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
 - Content Type: Application Data (23)
 - Version: TLS 1.2 (0x0303)
 - Length: 511
 - Encrypted Application Data: 2a7333548e4f7b8b081fab5d0f5216e4d7aa57049de4d77b5657 [Application Data Protocol: Hypertext Transfer Protocol]

Packet 1452 bytes:

```

01a0 27 c7 0a ea be aa 0a 57 c5 07 99 b3 49 f8 76 b8 .....W ....I-v-
01b0 62 82 a0 26 7a 7e ec 60 66 06 b6 f9 78 fd e9 6a b-82-... f-x-
01c0 20 dd d2 57 23 91 7a 60 e7 91 ae ad 8d a7 ea ab ...M-z- .....
01d0 da 9f b1 09 6b cd 4e 3f 79 22 4b 0b 14 05 22 68 ....k-N? y"K-...
01e0 49 07 2c 74 7f 3d d1 d6 79 88 8e ca 6d 43 b9 2b I-... y-mC-...
01f0 ca 0e 17 73 7c 86 51 e1 03 2f c6 47 9b d8 a3 22 ...s]-Q- -/G-...
0200 d4 34 cc ef fd 1b 0b 8c 1c 52 33 38 a4 5a c5 0a .4-.....-R3B-Z-
0210 77 73 cd 7d 3b e0 66 28 62 9b e6 cf 49 1b 98 05 ws-);f( b-...I-...
0220 60 82 87 e2 01 08 b7 d6 77 70 6f 8c 1e bc bf c1 .....wpo-....
  
```