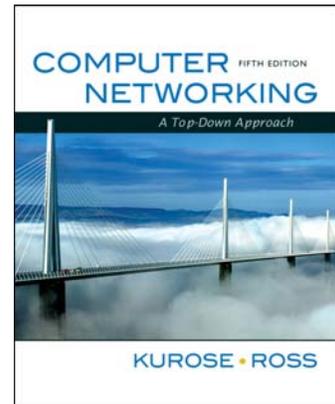


Wireshark Lab: SSL



Version: 2.0
© 2009 J.F. Kurose, K.W. Ross. All Rights Reserved

Computer Networking: A Top-down Approach, 5th edition.

In this lab, we'll investigate the Secure Sockets Layer (SSL) protocol, focusing on the SSL records sent over a TCP connection. We'll do so by analyzing a trace of the SSL records sent between your host and an e-commerce server. We'll investigate the various SSL record types as well as the fields in the SSL messages.

No.	Time	Source	Destination	Protocol	Info
106	21.805705	128.238.38.162	216.75.194.220	SSLv2	Client Hello
108	21.850201	216.75.194.220	128.238.38.162	SSLv3	Server Hello,
111	21.853520	216.75.194.220	128.238.38.162	SSLv3	Certificate
112	21.876168	128.238.38.162	216.75.194.220	SSLv3	Client Key Exchange, Change Cipher S
113	21.945667	216.75.194.220	128.238.38.162	SSLv3	Change Cipher Spec, Encrypted Handsh
114	21.954189	128.238.38.162	216.75.194.220	SSLv3	Application Data
122	23.480352	216.75.194.220	128.238.38.162	SSLv3	Application Data
123	23.481632	216.75.194.220	128.238.38.162	TCP	[TCP segment of a reassembled PDU]
127	23.482041	216.75.194.220	128.238.38.162	SSLv3	[TCP out-of-order] Application Data
129	23.482615	216.75.194.220	128.238.38.162	TCP	[TCP segment of a reassembled PDU]
138	23.537378	216.75.194.220	128.238.38.162	SSLv3	[TCP out-of-order] Application Data
140	23.537671	216.75.194.220	128.238.38.162	TCP	[TCP segment of a reassembled PDU]

Frame 106 (132 bytes on wire, 132 bytes captured)
Ethernet II, Src: Ibm_10:60:99 (00:09:6b:10:60:99), Dst: All-HSRP-routers_00 (00:00:0c:07:ac:00)
Internet Protocol, Src: 128.238.38.162 (128.238.38.162), Dst: 216.75.194.220 (216.75.194.220)
Transmission Control Protocol, Src Port: 2271 (2271), Dst Port: https (443), Seq: 1, Ack: 1, Len
Secure Socket Layer
SSLv2 Record Layer: Client Hello
Length: 76
Handshake Message Type: client Hello (1)
Version: SSL 3.0 (0x0300)
Cipher Spec Length: 51
Session ID Length: 0
Challenge Length: 16
Cipher Specs (17 specs)

```
0000 00 00 0c 07 ac 00 00 09 6b 10 60 99 08 00 45 00 .....k....E.  
0010 00 76 48 28 40 00 80 06 6f a1 80 ee 26 a2 d8 4b .vH(@...o...&...K  
0020 c2 dc 08 df 01 bb 56 d2 08 c5 4c 9e 64 9f 50 18 .....V...L.d.P.  
0030 ff ff e7 55 00 00 80 4c 01 03 00 00 33 00 00 00 .....U...L...3...  
0040 10 00 00 04 00 00 05 00 00 0a 01 00 80 07 00 c0 .....@...d..b.  
0050 03 00 80 00 00 09 06 00 40 00 00 64 00 00 62 00 .....@...d..b.  
0060 00 03 00 00 06 02 00 80 04 00 80 00 00 13 00 00 .....@...d..b.  
0070 12 00 00 63 06 df 78 4c 04 8c d6 04 35 dc 44 89 .....cf.xL...5.D.  
0080 89 46 99 09 .....E.....
```

Challenge data used to authenticate server (ssl.handshake.challenge), 16 bytes P: 336 D: 70 M: 0

1. Capturing packets in an SSL session

The first step is to capture the packets in an SSL session. To do this, you should go to your favorite e-commerce site and begin the process of purchasing an item (but terminating before making the actual purchase!). After capturing the packets with Wireshark, you should set the filter so that it displays only the Ethernet frames that contain SSL records sent from and received by your host. (An SSL record is the same thing as an SSL message.) You should obtain something like screenshot on the previous page.

If you have difficulty creating a trace, you should download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip> and extract the *ssl-ethereal-trace-1* packet trace.

2. A look at the captured trace

Your Wireshark GUI should be displaying only the Ethernet frames that have SSL records. It is important to keep in mind that an Ethernet frame may contain one or more SSL records. (This is very different from HTTP, for which each frame contains either one complete HTTP message or a portion of a HTTP message.) Also, an SSL record may not completely fit into an Ethernet frame, in which case multiple frames will be needed to carry the record.

1. For each of the first 8 Ethernet frames, specify the source of the frame (client or server), determine the number of SSL records that are included in the frame, and list the SSL record types that are included in the frame. Draw a timing diagram between client and server, with one arrow for each SSL record.
2. Each of the SSL records begins with the same three fields (with possibly different values). One of these fields is “content type” and has length of one byte. List all three fields and their lengths.

ClientHello Record:

3. Expand the ClientHello record. (If your trace contains multiple ClientHello records, expand the frame that contains the first one.) What is the value of the content type?
4. Does the ClientHello record contain a nonce (also known as a “challenge”)? If so, what is the value of the challenge in hexadecimal notation?
5. Does the ClientHello record advertise the cipher suites it supports? If so, in the first listed suite, what are the public-key algorithm, the symmetric-key algorithm, and the hash algorithm?

ServerHello Record:

6. Locate the ServerHello SSL record. Does this record specify a chosen cipher suite? What are the algorithms in the chosen cipher suite?

7. Does this record include a nonce? If so, how long is it? What is the purpose of the client and server nonces in SSL?
8. Does this record include a session ID? What is the purpose of the session ID?
9. Does this record contain a certificate, or is the certificate included in a separate record. Does the certificate fit into a single Ethernet frame?

Client Key Exchange Record:

10. Locate the client key exchange record. Does this record contain a pre-master secret? What is this secret used for? Is the secret encrypted? If so, how? How long is the encrypted secret?

Change Cipher Spec Record (sent by client) and Encrypted Handshake Record:

11. What is the purpose of the Change Cipher Spec record? How many bytes is the record in your trace?
12. In the encrypted handshake record, what is being encrypted? How?
13. Does the server also send a change cipher record and an encrypted handshake record to the client? How are those records different from those sent by the client?

Application Data

14. How is the application data being encrypted? Do the records containing application data include a MAC? Does Wireshark distinguish between the encrypted application data and the MAC?
15. Comment on and explain anything else that you found interesting in the trace.