Secure Socket Layer Implementations-A Review

Alphonsa Johny
Department of Computer Science &Engineering
Sree Chitra Thirunal College of Engineering
Trivandrum, India
alphonsajohny1@gmail.com

Dr. Jayasudha J.S
Department of Computer Science &Engineering
Sree Chitra Thirunal College of Engineering
Trivandrum, India
jayasudhajs@gmail.com

Abstract— Secure Socket Layer (SSL), is the protocol developed by Netscape for transmitting private documents securely over the Internet. SSL can be effectively used to protect the data in transmission. SSL protocol comes in between the application layer protocol (e.g., HTTPS (Hyper Text Transfer Protocol Secure)) and the Transport layer protocol. The http application interfaces with SSL nearly in the same way as it would with TCP in the absence of security. As far as TCP is concerned, SSL is just another application protocol using its services.SSL consist of two sub-protocols, the Hand-shake protocol and Record protocol. The strength of the SSL and hence the performance offered by the secure communication is determined by the selection of the cipher suite. The cipher suite itself has got four components and they are technique for key exchange, authentication, encryption and the method to compute the message digest hash. Different methods are available for all the above said technique and we need to select a cipher suite that will perfectly match our performance requirements, speed and memory constraint. Different built in libraries are there for the developers to use. This paper provides a brief comparison and description regarding the most commonly used SSL implementations such as OpenSSL, CyaSSL and MatrixSSL.

Keywords- Secure Socket Layer (SSL); Transport Layer Security (TLS); Datagram Transport Layer Security; Hand-shake Protocol; Record Protocol; Advanced Encryption Standard (AES).

I. INTRODUCTION

The internet uses TCP/IP for data transfer and the TCP/IP protocol does not have an inherent security/protection mechanism. So the data in transition may get affected by active attacks such as message forgery and message alteration [1]. In order to protect the data in transmit, Netscape introduced the concept of SSL protocol in 1994 and later other companies such as Microsoft began to develop their own security protocols. Then Internet Engineering Task Force (IETF) intervened to define a standard for an encryption-layer protocol. With the input from multiple vendors, the IETF created Transport Layer Security standard. Previous versions of SSL are SSL 2.0 and SSL 3.0.Transport Layer Security (TLS) is a protocol based on SSL 3.0 and TLS 1.0 is the same as SSL 3.1.Eventhough TLS and SSL protocols differ slightly in their implementation, the application developer and user cannot detect any differences at all [2].

Any network that needs to transmit data over an unsecured network such as Internet can make use of this SSL protocol to ensure security. Today SSL is widely used in online banking transactions (where we need to protect the valuable user credentials such as password, PIN number etc), military purposes, embedded systems etc. It provides secure communication between the client and server .Here, the Client is our browser and server is the web server with which we are communicating [3]. A communication is said to be secure if it ensures confidentiality, authentication and integrity of the message [4]. Confidentiality is used to ensure that only authorized persons are reading the message. By integrity property, it means that the message is not modified in transmit. Authentication is achieved by using digital certificates and it is used to ensure that we are communicating with the real users and not to any imposter. In this paper we have given a detailed description of SSL protocol. Begin with the structure of SSL, its position in the network architecture, security threats on SSL and the later portion of this paper covers the various SSL implementations such as MatrixSSL, OpenSSL, and CysSSL and a comparative study of them.

II. SSL STRUCTURE

SSL comes in between the http application layer and TCP. Figure 1 shows the position of SSL in Network Architecture. SSL requires few changes to the protocols above and below it [5]. The http application interfaces with SSL nearly in the same way it would with TCP in the absence of security. As far as TCP is concerned, SSL is just another application using its services. SSL is composed of two sub-protocols, the Hand-shake protocol and the Record protocol.

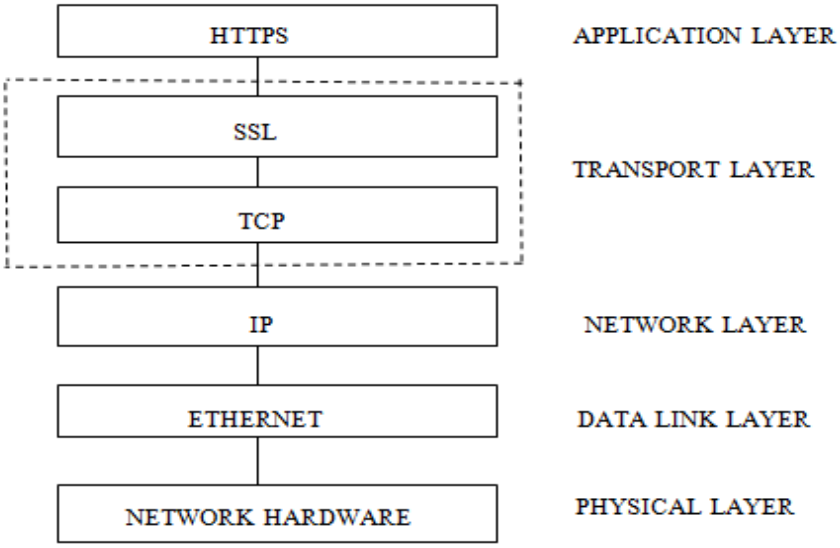

Figure 1. SSL position in the network architecture

A. *Hand-shake protocol*

Hand-shake protocol is called so because it performs the initial handshaking operations such as certificates exchanging, key materials exchanging, and identity authentication. The various step involved in the hand-shake protocol are given below [6],

- A client sends a *ClientHello* message to a server to ask for a SSL connection. This *ClientHello* message specifies the highest version of SSL it supports, a random number, a list of suggested cipher suits and a compression method the client can support.
- The Server responds with a *ServerHello* message, containing the chosen protocol version, a random number, cipher suite, and compression method from the choices offered by the client. The server may also send a session ID as part of the message to perform a resumed handshake.
- The Server sends its *Certificate message* (depending on the selected cipher suite, this may be omitted by the Server).
- The Server sends a *ServerHelloDone* message, to indicate that handshake negotiation is completed.
- The Client responds with a *ClientKeyExchange* message, which may contain a pre-master secret, public key, or nothing. (It depends on the selected cipher.)
- The Client and Server use the random numbers and Pre-Master Secret to compute a common secret, called the *"master secret"*, and then the both parties use the master secret to compute a key-block. All other key data for this connection is derived from this key-block (the client and server-generated random numbers), which is passed through a carefully designed *"pseudorandom function"*. The key data includes two session keys: *client_write_key* and *server_write_key*.
- The Client now sends a *ChangeCipherSpec* record, essentially telling the Server, "Everything I tell you from now on will be encrypted." The *ChangeCipherSpec* is itself a record-level protocol.
- Finally, the Client generates a *finished* message containing a hash and MAC over the previous handshake messages, and encrypts it using the *client_write_key* before sending it.

- The Server will attempt to decrypt the Client's *finished* message by the *client_write_key*, and verify the hash and MAC. If the decryption or verification fails, the handshake is considered to have failed and the connection should be torn down.
- Finally, the Server sends a *ChangeCipherSpec* and its encrypted *finished* message, and the Client performs the same decryption and verification.

At this point, the "handshake" is complete and the application protocol is enabled. Application messages exchanged between Client and Server will be encrypted.

B. Record protocol

The record protocol uses the session keys produced in hand-shake protocol to encapsulate the data to be exchanged. The encapsulation can provide confidentiality and integrity for data.

C. Cipher suites

Before sending data, the sender and receiver must reach at a conclusion regarding the components of the selected cipher suite [7]. The four components of the cipher suite are key exchange algorithm; authentication technique, encryption method and the algorithm to compute the message digest hash. The key exchange algorithm specifies how the peers agree upon a common symmetric key that can be used to encrypt the message after the handshaking is done. The two common key exchange algorithms are DHE (Diffie-HellmanKeyExchange) and RSA (Rivest-Shamir-Adelman).The authentication algorithm indicates how the Client and Server will prove their identities to each other. Authentication options include RSA (Rivest-Shamir-Adelman),DSA(Digital signature algorithm), Elliptic curve DSA, PreShared Key(PSH) and anonymous(when no authentication mechanism is used) .Next, the encryption component indicates which symmetric algorithm is to be used to encrypt the data before transmission.RC4(RivestCipher 4),DES(Data Encryption Standard),3des(triple des),AES (Advanced Encryption Standard) are the commonly used ones. The RC4 algorithm is the fastest and smallest of the cipher algorithms, and therefore is ideal for embedded processors. And finally the digest hash component is used to ensure that the receiver receives what the sender has transmitted, i.e. the message is not modified in transmission. SHA-1 is the most commonly used checksum algorithm to confirm the integrity of the exchanged data.

III. THREATS TO SSL PROTOCOL

SSL protocol is prone to some attacks or security threats .Cipher suite rollback attack, version rollback attack, exchange algorithm rollback attack and dropping change cipher spec attack are more popular. The subsequent sections give a brief description of such at tacks and possible counter measures.

A. Cipher suite rollback attack

This type of attack mostly happens in SSL 2.0 exchange protocol where the attacker intercepts the client's Hello message and modifies the list of cipher suites which is stored in a plaintext. When the cipher suites are modified, now the attacker can actually force the user to use the export-weakened encryption, even if both the server and clients preferred and supports stronger grade algorithm. We can mitigate this problem by using a higher version of SSL (SSL 3.0) and by including hash values for all the messages during SSL handshake protocol and authenticating it in the finished messages by verifying the hash value and the messages. Both the server and the client are not supposed to accept the application data until the finished message is verified from both sides [7].

B. Version rollback attack

Version Rollback attack is an attack that uses a mechanism that makes the higher version capable clients (V 3.0) and server to fall back to a lower version (V 2.0). The server is fooled into thinking that he is communicating with a client who supports only SSL 2.0. This is because the SSL 2.0 has its own fallacies and defect. The SSL 2.0 version of the protocol does not include "Finished" messages which make the attacker to exploit numerous SSL 2.0 vulnerabilities. This problem can be detected by making the client to implement fixed redundancy RSA PKCS padding bytes which supports SSL 3.0. The server which supports the SSL 3.0 will disagree to accept RSA encrypted key exchange over SSL 2.0 only if those padding bytes are included in the RSA encryption [7].

C. Exchange algorithm rollback

In this type of attack, the attacker intercepts the client hello message if it's a plaintext and will try to replace it with a weak or randomly selected key exchange algorithm. The attacker repeats the same in the message from a server to a client and replaces with an algorithm he wants [8].We can recover from this type of attack by transmitting signed parameters so that the attacker can never replace an algorithm.

D. Dropping change cipher spec attack

The Change Cipher Spec message is sent by both the server and the client to notify each other that the subsequent messages will be protected by the newly negotiated Cipher Spec and keys. It is a fact that the Cipher Spec is excluded from the calculation of a MAC from previous handshake messages, which is used as an authentication code for the finished message. The attacker can actually drop the Change Cipher Spec message making the client and the server to never change from read pending state to read current state, disabling the message authentication and encryption at the record layer. We can mitigate this problem by checking the Change Cipher Spec message before the finished message is verified and also with the use of SSL version3.0 which was designed to correct this flaw [9].

IV. COMMON SSL LIBRARIES

The most popular implementation of TSL (Transport Layer Security) protocols are CyaSSL, OpenSSL, and MatrixSSL. The MatrixSSL itself comes in two versions-open and commercial. A brief description of each one of is given below.

A. OpenSSL

OpenSSL is based on the excellent SSLeay library developed by Eric A. Young and Tim J. Hudson. The main feature of OpenSSL Project is that it is full-featured and Open Source toolkit implementing the Secure Socket Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols [10]. The latest version of OpenSSL is OpenSSL 1.0.1c and it is licensed under an Apache-style license, and hence the developers are free to get and use it for commercial and non-commercial purposes subject to some simple license conditions. OpenSSL toolkit can be used for the following purposes,

- Creation and management of private keys, public keys and parameters.
- Public key cryptographic operations.
- Creation of X.509 certificates, CSRs and CRLs.
- Calculation of Message Digests.
- Encryption and Decryption with Ciphers.
- SSL/TLS Client and Server Tests.
- Handling of S/MIME (Multipurpose Internet Mail Extension) signed or encrypted mail.
- Time Stamp requests, generation and verification

OpenSSL supports a number of different cryptographic algorithms: Ciphers supported by OpenSSL are AES(Advanced Encryption Standard),Blowfish, Camellia, SEED(It is a block cipher developed by Korean security Agency), CAST-128(alternatively known as CAST5,it is a block cipher gets its name from the inventors Carlisle Adams and Stafford Tavares),DES(Data Encryption Standard), IDEA(International Data Encryption Algorithm),RC2(Rivest-Cipher 2) RC4, RC5, Triple-DES and GOST 28147-89. The following Cryptographic hash functions can be used with OpenSSL MD5 (Message Digest5), MD2, SHA-1(, SHA-2, RIPEMD-160, MDC-2, GOST R 34.11-94.

B. CyaSSL

The CyaSSL is an embedded SSL library, which is lightweight and written in ANSI C [11]. It is targeted for embedded and Real Time Operating System (RTOS) environments-mainly because of its small size, speed, and feature set. CyaSSL supports industry standards up to the current TLS 1.2 level and DTLS (Datagram Transport Layer Security). It is up to 20 times smaller than OpenSSL, and offers progressive ciphers such as HC-128, RABBIT, and NTRU .It offers full client and server support. OCSP and CRL support is also there in CyaSSL. Its size is in the range 30-100kB.Runtime memory requirement is 3-36kB.The first major user of CyaSSL/yaSSL was MySQL, the world's most popular open source database. CyaSSL is currently available for Win32/64, Linux, Mac OS X, Solaris, FreeBSD, NetBSD, OpenBSD, embedded Linux, Haiku, Open wrt,iPhone , Android, Nintendo Wii and GameCube through DevKitPro support, QNX, VxWorks, Monta Vista, ThreadX, Tron variants, OpenCL, Micrium'sMicroC-OS/II, FreeRTOS, Freescale MQX, and Nucleus.

C. MatrixSSL

MatrixSSL is an embedded SSL and TLS implementation designed for small footprint applications and devices [12]. It is available as an open source resource and it perfectly match with the requirements of embedded systems (memory constraint, processing capability and battery power constraints). MatrixSSL has been ported to operating systems including VxWorks, uClinux, eCos, FreeRTOS, ThreadX, ARM, MIPS32, PowerPC, H-8, SH3, i386 and x86-64.Important features of MatrixSSL are given below

- Total footprint of MatrixSSL with crypto provider is less than 50KB.
- SSL 3.0, TLS 1.0 and 1.1 servers and client support.

- Included crypto library - RSA, ECC, 3DES, AES, ARC4, SHA1, MD5, and RC2.
- Cipher Suites - RC4-MD5, RC4-SHA, DES-CBC3-SHA, AES128-SHA, AES256-SHA.
- Full support for session resumption/caching.
- Session re-keying and cipher renegotiation.
- Server and client X.509 certificate chain authentication.
- Parsing of X.509 pem and ASN.1 DER certificate formats.
- PKCS#1.5, PKCS#5 PKCS#8 and PKCS#12 support for key formatting.
- SSH command line support.
- Fully cross platform, portable codebase; minimum use of system calls.
- Pluggable cipher suite interface.
- Pluggable crypto provider interface.
- Pluggable operating system and malloc interface.
- TCP/IP optional.
- Multithreading optional.
- Only a handful of external APIs, all non-blocking.
- Clean, heavily commented code in portable C.

V. COMPARISON OF SSL/TLS LIBRARIES

The commonly used SSL libraries varies significantly in their memory requirements ,security features offered, cost and difficulty in implementation etc. This section gives a comparative study of them. “Table 1” gives an overview of the above said three libraries.

TABLE I. OVERVIEW

Implementation	Developed By	Open Source	Software License	Latest Stable Version	Release Date
CyaSSL	CyaSSL	Yes	GPLv2 &commercial license	2.4.0	10-10-12
OpenSSL	OpenSSL project	Yes	OpenSSL/SSLeay dual license	1.0.1c	10-05-12
MatrixSSL	PeerSec Networks	Yes	Proprietary	3.3	22-02-12
MatrixSSL-open	PeerSec Networks	Yes	GPLv2(General public License version 2)	3.3	22-10-12

A. protocol support

Several versions of the TLS protocol such as SSL 2.0, SSL 3.0, TLS1.0 exists, but the performance offered by each of them is considerably different. For example the SSL 2.0 is a deprecated protocol, vulnerable to several attacks. In comparison to SSL 2.0, SSL 3.0 andTLS1.0 offers better performance and it does not have any major known vulnerabilities. TLS 1.2 is the latest published version, introducing new features. Datagram TLS (DTLS) is a modification of TLS 1.1 for a packet-oriented transport layer; here we need to consider the packet loss and packet reordering. Comparison of SSL libraries based on the protocols they support is given in “table 2”.

TABLE II. PROTOCOL SUPPORT

Implementation	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	DTLS1.0	DTLS 1.2
CyaSSL	No	Yes	Yes	Yes	Yes	Yes	No
OpenSSL	Yes	Yes	Yes	Yes	Yes	Yes	No
MatrixSSL-open	No	Yes	Yes	Yes	No	No	No
MatrixSSL	No	Yes	Yes	Yes	Yes	Yes	Yes

REFERENCES

[1] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[2] S.Thomas, "SSL and TLS essential", John Wiley & Sons, inc, 2000.

[3] W.B.Mao,"Modern Cryptography:Theory and Practice,"Publishing House of Electronics Indudtry,Beijing,2004

[4] Zhai Xuefeng, "The security analysis of SSL and the research and realization of its being hijacked," Sichuan University,2004.

[5] Qianqian Ge,Feng Chen,"Strategies for Implementing SSL on Embedded System",2008 International Seminar on Future BioMedical Information Engineering.

[6] G. Apostolopoulos, V. Peris V,D. Saha, "Transport Layer Security How much does it really cost," Infocom'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, IEEE,1999,pp. 717-725

[7] David Wagner, Bruce Schneier, "Analysis of the SSL 3.0 protocol", USENIX Workshop on ElectronicCommerce,ACM.

[8] Korea Information Security Agency, "A development of modules for Improving an ability and security of SSL/TLS", January, 2002

[9] www.jucs.org Embedded Monitors for Cryptographic Protocol Intrusion Detection and Prevention http://www.jucs.org/jucs_11_1/protomon_embedded_monitors_for/Joglekar_S_P.html.Retrieved:December5,2009.

[10] The Open Source Toolkit for SSL/TLS .www.openssl.org.

[11] Embedded SSL Library for Applications, Devices, and the Cloud, http://www.yassl.com/yaSSL/Home.html

[12] MatrixSSLSpecification,www.matrixssl.org