I. J. Computer Network and Information Security, **2016, 2, 15-24**
Published Online February 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2016.02.02

Taxonomy of SSL/TLS Attacks

Keerthi Vasan K. and Arun Raj Kumar P.
National Institute of Technology (NIT) Puducherry, Karaikal, India
E-mail: keerthivasankkv@gmail.com, p.arunrajkumar@nitpy.ac.in

Abstract—Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols use cryptographic algorithms to secure data and ensure security goals such as Data Confidentiality and Integrity in networking. They are used along with other protocols such as HTTP, SMTP, etc. in applications such as web browsing, electronic mail, and VoIP. The existing versions of the protocols as well as the cryptographic algorithms they use have vulnerabilities and is not resistant towards Man-In-The-Middle (MITM) attacks. Exploiting these vulnerabilities, several attacks have been launched on SSL/TLS such as session hijacking, version degradation, heart bleed, Berserk etc. This paper is a comprehensive analysis of the vulnerabilities in the protocol, attacks launched by exploiting the vulnerabilities and techniques to mitigate the flaws in protocols. A novel taxonomy of the attacks against SSL/TLS has been proposed in this paper.

Index Terms—SSL/TLS, vulnerabilities, Man-In-The-Middle (MITM) attack, mitigations, taxonomy of attacks.

I. Introduction

Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols run above the Transport Layer, providing protection to application layer traffic. The protocol achieves secure communication between a pair of nodes by exchanging parameters such as cipher suite, key, etc. and using them to encrypt application layer message. The SSL/TLS protocol is used in electronic mailing services, VOIP, banking, social networking, file transfer and web browsing. SSL protocol [1] was designed by Netscape for providing secure HTTP connection from its browser, Netscape Navigator. Based on SSL v3.0 [2], the TLS protocol was proposed in [3]. HTTPS is designed by making the HTTP protocol running on SSL/TLS on port 443. The STARTTLS command based implementation of TLS is used by various protocols such as POP3, SMTP, FTP, XMPP, LDAP and NNTP. TLS used with stateless transport protocols are referred as Datagram Transport Layer Security (DTLS) and for WAP architecture is called Wireless Transport Layer Security (WTLS).

The communication established through SSL/TLS secures application traffic. As the data is encrypted, sniffing of packets does not reveal actual data, ensuring confidentiality of messages exchanged. The Message exchanged is suffixed with a Message Authentication Code (MAC) before encryption. Hence, the integrity of the message is ensured. Thus, it is difficult to compromise the SSL/TLS protocol. Fig.1. depicts a session of the TLS v1.2 protocol [4]. Steps involved in the working of TLSv1.2 protocol are as follows:

1. *Client hello*: Client initiates communication by sending a hello message to the server. Client hello packet consists of cipher suites, compression methods, and hello extensions (additional parameters required for the protocol).
2. *Server hello*: On receiving the client hello, server selects a cipher suite, compression method, and requisite hello extensions from receives client hello message and sends this information as server hello.
3. *Server SSL certificate*: The server sends an X.509 certificate [**5**] using which the client authenticates the server.
4. *Server key exchange*: Server sends an ephemeral public key to the client. The client uses the key for encrypting information exchanged in the future.
5. *Client certificate request*: The server requests the client to send its SSL certificate. This message is generated only if client authentication is mandatory as per server policy.
6. *Server hello done*: The server indicates the client the end of server hello message, certificate and server public key exchange activities.
7. *Client verify*: The client sends an X.509 certificate or shared secret (authentication information unique to the client, known to server) to the server. This is sent only if client certificate request is sent by server.
8. *Client key exchange*: Client sends either of the following keys to the server:

   i. Premaster secret (48-bit) - client sends the server a premaster secret encrypted using temporary or permanent public key of the server (symmetric key exchange).
   ii. Client public key - Client sends its public key to the server (asymmetric key exchange).

9. *New session ticket:* The server sends a session ticket that is encrypted using shared key/client public key. For every session, a new ticket is generated. The vital information in the session ticket include:

   i. Session ID – unique number to denote the ticket
   ii. Ticket TTL - time till which ticket is valid

10. Change cipher specification: Both server and client agree to work using the decided parameters such as

cipher suites, compression method, encryption key and session ticket. This is indicated by exchanging the CCS message. This is followed by exchange of application data.

Although SSL/TLS protocols secure the data, vulnerabilities in the protocol make it weaker. The SSL/TLS protocol is vulnerable to Man-In-The-Middle (MITM) attack. Several vulnerabilities in the protocols are exploited when the protocol is used in a channel compromised by MITM. The contributions of the paper are as follows: Section 2 discusses the vulnerabilities in the protocol. Section 3 is a taxonomy of attacks against the protocol till date and section 4 discusses on the mitigations available against some attacks. Eventually, the paper concludes in Section 5.

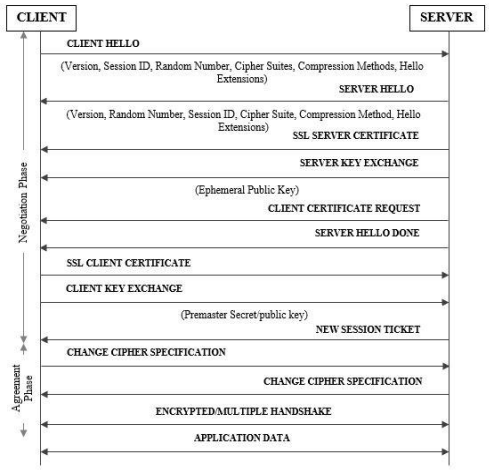

Fig.1. Working of TLS v1.2 Protocol

II. Vulnerabilities in SSL/TLS Protocol

SSL/TLS exchange application data between two communicating nodes in a secure manner. They make use of cryptographic algorithms such as Data Encryption Standards (DES), Advanced Encryption Standards (AES), Message Digest (MD), Secure Hash Algorithm (SHA), etc. As seen in Fig.1, the session of TLS protocol consists of two phases namely, negotiation phase and agreement phase. In the negotiation phase, all information exchanged for establishing a secure channel by SSL/TLS is not protected and can be sniffed by a MITM attack. Using information sniffed from various messages exchanged, an attacker can cryptanalyze the application data exchanged later, in the agreement phase of the same session to steal the data. Moreover, legacy encryption algorithms such as DES, MD4, etc. have become weaker due to their vulnerabilities. Using these weak algorithms in the cipher suites of the protocol makes the protocol vulnerable, leading to failure of the data confidentiality and integrity. Due to insecure negotiation phase, improper implementation of the protocol steps such as CCS exchange and usage of legacy cryptographic algorithms, the SSL/TLS protocol is vulnerable [8]. The vulnerabilities in SSL/TLS are as follows:

A. Client and Server Hello

- *Version Rollback Policy:* Version number in hello messages indicate the latest protocol version supported by the sender. If latest version support of both the nodes differ, the least recent version common in both nodes is selected. So one of the nodes has to rollback its version. Older the negotiated version, weaker the protocol. Hence this policy becomes vulnerability.
- *Cipher Suite/ Legacy Cipher Suites:* Lower versions of protocols have fewer cipher suites which consists of legacy cryptographic algorithms. These algorithms will have become vulnerable over time and using them makes the protocols weaker.
- *Session ID/ Renegotiation Policy:* Session ID is exchanged between client and server when parameters of the session the ID refers to are going to be reused in the current session. This is called Renegotiation Policy. If the session referred by the session ID was compromised, it is likely that current session can also be compromised. Also, even if the previous session was not compromised, using the same parameters in current session will make more data available for cryptanalysis leading to compromising of both sessions.
- *Compression Policy:* Lossless compression algorithms affix redundant substrings to application traffic. If the initial length of data before compression is small, after compression it may be larger than actual length. Moreover, the redundancy can be used by attacker for cryptanalysis which will reveal substring of the text eventually may lead to decoding the whole message.

B. Certificate Exchange

- *Unknown CA:* During certificate exchange, if receiver does not recognize the Certifying Authority (CA), the handshake must be rejected as the CA maybe illegitimate.
- *Self-signed certificate:* Self-signed certificate does not provide authentication and must rejected by receiver as the sender may be a fake or malicious user.

If the receiver overrides either of the policies mentioned above, the connection can be compromised by an attacker who exploits this vulnerability.

C. Change Cipher Specification

Change Cipher Specification (CCS) message exchanged during the agreement phase mutually acknowledges the client and server decision on using the parameters exchanged during negotiation phase. If there is any error in parameter exchange during negotiation, either or both nodes select the default parameters, 0x0000 cipher suite which literally does no encryption on the data or rejection of handshake between the peers.

- CCS message exchanged in-between a negotiation

phase will lead to either no encryption or connection termination. The possibility of introducing an inappropriate CCS is vulnerability.

- CCS message when never exchanged before timeout will lead to rejection of negotiations made earlier. The possibility of blocking a legitimate CCS message is vulnerability.

D. Explicit Alert Message/ Server as Oracle

Alert messages are generated to indicate warning or error during communication. Whenever an error occurs, alert message is generated and all alerts are designated with alert IDs. The some alert messages explicitly mention the nature of the error. A malicious user can use this ID and error description as data about sent message and hence abuse the node as an oracle. Regressively querying the oracle, the malicious user understands the working of it and it can be attacked by the abusive user.

E. Multiple Open Ports at server

Server implementations may allow an application server to run in more than one port, where a few use SSL/TLS while a few do not use SSL/TLS. In such a scenario, if a request to the application server port supporting SSL/TLS is redirected to a port not using SSL/TLS, the connection will not be secure. A malicious user can redirect his target traffic to an unsecure port during any request. This becomes vulnerability.

For instance, when *OpenSSL* is configured for *Apache2* server, by default, the open ports in the configuration file *ports.conf* are 80 and 443. Hence, a MITM can redirect any request from the client(s) to port 80 and make the communication insecure.

F. STARTTLS implementation policy

Several application layer protocols use STARTTLS implementation for SSL/TLS. This implementation requires a node to explicitly send a STARTTLS request to initiate the negotiation phase. If the STARTTLS is overridden by an attacker, SSL/TLS will not be used and hence the communication becomes insecure.

Table 1. SSL/TLS Attacks and Vulnerability

Vulnerability	Attack
Version Rollback policy	Version Rollback attack
Legacy Cipher suites	Beast, RC4, Poodle
Renegotiation Policy	Renegotiation attack, Triple Handshake attack
Compression Policy	Crime, Breach
CCS Message	CCS manipulation
Explicit Alert Message	Padding Oracle, Poodle, Bleichenbacher attack, Berserk, Lucky 13
Multiple Open Ports	SSL Stripping
StartTLS Policy	StartTLS manipulation
TCP fin override	Truncation attack, Triple Handshake attack
Time Delay in response	Time, Lucky 13
Heartbeat Policy	Heart Bleed
Shared certificate policy	Virtual Host Confusion

G. TCP Fin message override

After a transport layer session ends, TCP exchanges messages with *Fin flag* set. This is sent if the SSL/TLS stops negotiating as no more application layer data needs to be exchanged. If a *Fin flag* set message is exchanged and is dropped by the MITM attacker, the transport layer session will remain open. This vulnerability can be exploited either to damage the user resources in the server or launch a broker a new negotiation that might occur in the near future.

H. Time delay in response from server

There exists a delay in response from the server depending on the type of message and correctness of message. Using this delay as a significant parameter, a malicious user can use the server as an oracle to study the semantics of message and use it for cryptanalysis. If this time delay is significant parameter to classify valid and invalid requests, the attacker might be able to decrypt message without server secret key over a period of time.

I. Heartbeat policy

The heartbeat policy makes use of a special kind of message called heartbeat, which checks a machine is alive or not. Each node has a heartbeat message string which is known to other nodes. When the heartbeat request is sent, sender anticipates the heartbeat string of the receiver in the response message. Length of anticipated string is attached to the request. A malicious user can generate a request with larger length value such that the information in the buffer are leaked by the receiver. This vulnerability causes the Heart Bleed vulnerability [6, 7].

J. Shared Certificate Policy

If a service is hosted at a single IP address across different domains, each service is considered as a virtual host. Consider that a common X.509 certificate is shared by the entire server hosted in same IP address. If one of the virtual hosts is compromised by the attacker, using Cross Site Scripting (XSS) all requests can be redirected to the compromised virtual host. Hence all the services become vulnerable.

Based on the vulnerabilities, a malicious user can launch various attacks on the SSL/TLS protocol and the data secured by it. Table 1 relates vulnerabilities and the attacks possible due to them.

III. Taxonomy of SSL/TLS Attacks

SSL/TLS protocol is vulnerable to Man-In-The-Middle (MITM) attack, where the attacker can sniff the data from the communication channel. By exploiting the vulnerabilities in the protocol design, policies used and cryptographic algorithms in the cipher suites, several attacks can be launched on the protocol. In order to launch any of these attacks, sniffing of packets is required which is achieved by MITM attack. From Table 1, one can relate how vulnerability can be exploited to launch an

attack.

Fig.2. depicts the proposed novel taxonomy of attacks in SSL/TLS protocol. The attacks [8, 9] against the protocol are initially classified depending on where the attack is launched, namely (1) Client end, (2) Server end, and (3) During Transit. The attack at client end is classified depending on which layer attack happens, namely (1) Application layer, (2) Presentation layer, where SSL/TLS functions; and (3) Transport layer. Presentation layer is rather depicted as (1) Negotiation phase, and (2) Agreement phase of SSL/TLS protocol. The attack at server end is classified as (1) Side Channel attack, timing based attacks; (2) Manipulation attack, server is abused as an oracle; and (3) Certificate manipulation, where X.509 of a compromised server is used to compromise other servers. The attack during transit is classified as (1) Cipher based, and (2) Compression based. The attacks against SSL/TLS are as follows:

A. SSL stripping

SSL stripping refers to removing away the SSL/TLS data from a request message. It exploits the vulnerability of multiple open ports for same application server. MITM attacker redirects all requests to the application server through unsecure ports. Therefore, the data exchanged are unencrypted and hence unsecure. SSL stripping is also achieved by a penetration testing tool [10, 11] called the SSL strip. This tool removes the SSL/TLS based request messages from a client request and sends it to the server.

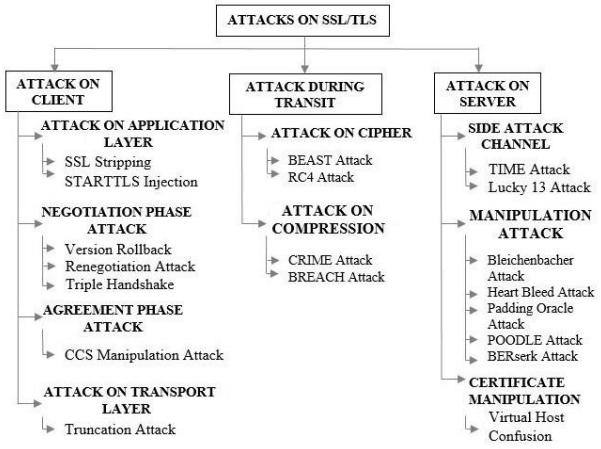

Fig.2. Taxonomy of SSL/TLS Attacks

The server assumes that the client does not support SSL/TLS and hence the established an insecure connection.

B. STARTTLS Manipulation

Several application layer protocols including POP3, SMTP, FTP, XMPP, LDAP, and NNTP use STARTTLS command for initiating SSL/TLS. STARTTLS [12] has a higher priority over other application layer commands. MITM attacker overrides STARTTLS policy of the application layer protocols by doing either of the following:

1. Attacker reads the buffer of the node and removes the command statement or modifying the code to skip execution of the STARTTLS command.
2. The attacker modified the command with weaker parameters which when used to encrypt data can be easily cryptanalyzed by him/her. Although the connection uses SSL/TLS, the communication does not happen secure manner.

The policy of explicitly stating the initiation of the protocol makes the implementation insecure.

C. Version rollback

The backward compatibility of the protocol is exploited to launch version rollback attack. The MITM attacker achieves version rollback by blocking the legitimate hello request from the client and sends a forged hello request with lower version to the server. Server assumes that the received version is the latest version supported by the client browser. Therefore the server does either of the following:

1. If server policy allows the received version, connection is established using a weaker set of parameters which can be easily cryptanalyzed.
2. If the version is unsupported, server hello selects the default suite, 0x0000 cipher suite and hence no encryption is applied to the data.

D. Renegotiation attack

Renegotiation policy [13, 20] in SSL/TLS is the idea of reusing all the parameters used in a previous session. This is done to reduce the traffic generated during negotiation. This requires mentioning the session ID of the session whose parameter are going to be reused in client hello. This policy has the following issues:

1. If the session whose parameters are reused was compromised by the attacker, then the current session can also be compromised.
2. Even if the old session was not compromised by the attacker, both old session and current session information can be used by attacker for cryptanalysis. If the encryption is broken data exchanged in both sessions are revealed.
3. Attacker steals the cookies from client browser and when the client is not active, he can masquerade as the client and connect to the server. Renegotiation allows the attacker to establish a secure connection with the server without authentication done on his machine.

Thus renegotiation policy although reduces complexity of the protocol, fails in providing security.

E. Triple handshake attack

The triple handshake attack [14] is based on the renegotiation vulnerability and TCP fin override. MITM attacker establishes a SSL/TLS connection with the

server using the stolen client cookies impersonating as client. This is achieved by launching a truncation attack. Next, when a request is generated by the actual client, the attacker portrays himself as the server and services the request from the client. Here, the attacker can act as middle man and relays the received message to the destined node. Every message is read by the attacker and confidentiality is lost. Although the overhead of message decryption and encryption during relaying exists, it is effective than cryptanalysis technique.

F. CCS manipulation attack

The negotiation becomes effective only after CCS message is exchanged. Improper usage of CCS message leads to error in the working of the protocol. The MITM attacker can manipulate the CCS message as described below:

1. At the start of a new session, an injected CCS message causes the nodes to select 0x0000 cipher suite.
2. During negotiation phase, injected CCS message confuses the receiver and causes failure of the negotiation.
3. If the CCS is dropped during agreement phase, the session remains passive until timeout after which the negotiation is rejected.

By doing any of the above mentioned, CCS manipulation attack is launched causing the failure of SSL/TLS protocol.

G. Truncation attack

Dropping TCP packets with flag Fin flag set keeps the transport layer session open in an inactive state. This vulnerability is used to launch truncation attack. If the Fin set packet is not delivered, destined receiver of the message (server) assumes the session is open. But to the sender (client), transmission in the session has ended from its end and awaits the server to send a Fin set message. Attacker can cause TCP Fin override and take control of the session. He can do the following:

1. Bring about undesirable change in the client information held by the server.
2. Launch a Triple Handshake Attack.

H. TIME attack

SSL/TLS messages consist of both negotiation messages and encrypted application data. The time taken to process one type of message differs from other [15, 16]. Similarly, time taken for processing and accepting a legitimate message differs from processing and sending an alert message for an error prone message. Using the difference in the response time of various messages, a malicious user tries to gain information about the nature of processing made by the server. Using this vulnerability, the attacker abuses the server as an oracle to identify master secret common to server and client. By a long trial and error method, MITM obtains the master secret. The contributing factors of Timing Information Made Easy (TIME) attack are:

1. Content type of message: Application data, Negotiation message or Alert message.
2. Correctness of message: message with error and message without errors.

I. Lucky 13 attack

Lucky 13 attack [17] is based on explicit alert message and time delay in response vulnerabilities. The server uses Message Encode-then-Encrypt (MEE) policy to achieve a constant response time for both correct and incorrect encryptions. But error in message padding will cause more delay than checking for encryption errors. This delay is used as vulnerability. Server is abused as an oracle to study about message padding and thereby decrypt the message using cryptanalysis.

J. Bleichenbacher attack

Bleichenbacher attack is a chosen ciphertext attack based on server as an oracle vulnerability. The attacker uses server as an oracle to decrypt the ciphertext. A large set of forged messages designed from the regular traffic and is used to study the encryption used by the server. Bleichenbacher attacks [18, 19] has several variants over time namely, PKCS v1.0, PKCS v1.5 and BERserk attack.

PKCS v1.0: A RSA key with a 3072 bit modulus and public exponent 3 is used to demonstrate the attack. The attack is composed of the following three stages:

1. *Blinding:* A preimage attack on ciphertext to map any fragment to its plaintext equivalent
2. *Conforming:* Assuming an arbitrary preimage map, associating several chunks with their plaintext equivalent
3. *Integrating:* Structuring a pattern using which any ciphertext associated with server is decrypted

From the patterns collected, all the data exchanged using the same parameters, which is in the same session, can be decrypted by the attacker.

K. Bleichenbacher certificate forgery

An advancement of bleichenbacher attack is to negotiate using a compromised public key. Attacker forges a X.509 certificate and sends it to client. The client uses public key in the forged certificate to establish a secure SSL/TLS connection [26].

PKCS v1.5: Key pair with public exponent 3 and key size of 1024/2048 bits is used in browser implementations. Also, several implementations fails to check the presence of any data after the FF_{16} octet, which is used as a delimiter. Based on this, the attacker generates a fake X.509 certificate. The following algorithm is used to forge a certificate,

i. Modify necessary fields like serial number,

issue date or validity. Make minimum modifications.

ii. Hash the resultant certificate and calculate

iii. If same, then we have a forged certificate with a signature where most the significant bit is same as the signature of the original certificate. Else restart the process.

Client on receiving the certificate trusts the Certifying Authority (CA). It uses the mentioned public key to encrypt data. The attacker holds the corresponding private key and decrypts the message using it.

L. BERserk attack

In order to fix the problem of bleichenbacher certificate forgery attack, the idea of including a data length field in the message was implemented. The encoding scheme BER with ASN.1 sequence [24] involved using PKCS1v15_BER_Parse_DigestInfo to parse a digest information sequence. The length field is implemented in one of the following ways,

1. Short length - 1 byte
2. Long length - 7 bytes

The first bit of the first byte in the message determines the length field type. If the bit is 0, then it is short length. If the bit is 1, then it is long length field. The MITM modifies the length field type bit to manipulate the server. By launching a bleichenbacher forgery attack, MITM can decrypt the ciphertext and tamper the client information.

M. Heart Bleed attack

In OpenSSL implementation, the client sends a request to the server called heartbeat to check the availability of the server. This message contains a keyword and the length of the keyword. Server allocates a buffer for every user communicating with it and a heartbeat message can fetch a maximum of 2^{16} Bytes from the buffer [7]. The server on receiving the message checks whether the sender is the authentic user by comparing its copy of keyword with the copy received. The attacker modifies the payload field value in such a way that the memory buffer containing the client data at server does not overflow. The server sends back data containing the authentication information of the client. This authentication information leaked fails confidentiality user credentials and the validity of the client is compromised.

N. Padding oracle attack

While using block ciphers for encryption, the data needs to be an integral multiple of block size. The application data is not of the same length always. So the original message is padded with 0s to achieve appropriate block size before encryption [21]. The mathematical expression goes as below:

$$length(data + MAC) = n * block\ size\ of\ cipher \quad (1)$$

As the ciphertext is resistant to preimage attack, server is used as a random oracle. With sufficient information on padding, the ciphertext can be decrypted without the key by cryptanalysis.

O. POODLE attack

Padding Oracle On Downgraded Legacy Encryption (POODLE) [22] is an instance of padding oracle attack. It exploits server as oracle and version rollback vulnerability. The MITM launches a version rollback attack and the client connection is established with SSLv3.0 or lower versions. The cipher suites contain legacy encryption ciphers like CBC and RC4, which are vulnerable. The padding oracle attack provides information on message padding. From the both information obtained and cryptanalysis of legacy ciphers, the ciphertext is decrypted without a proper encryption key.

P. Virtual Host Confusion attack

The use of shared TLS session caches and session tickets across different hosts weakens server authentication. The Same Origin Policy (SOP) allows interactions only between pages from the same origin (domain and application). By cross-site scripting (XSS) attack, user can override SOP. An IP address hosts multiple application servers sharing a common X.509 certificate, hence have common public key. Each service is considered as a virtual host. If one of the services is compromised, then all the traffic can to various virtual hosts can be redirected as per desire of the attacker using XSS. All messages can be decrypted using the corresponding private key.

Q. BEAST attack

Browser Exploit Attack on SSL&TLS (BEAST) [24, 25] is possible on ciphertext generated using Cipher Block Chaining (CBC). The vulnerability in CBC is that, if the encryption key is known, it is possible to retrieve the plaintext from the cipher text without the actual Initialization Vector (IV). At each stage, output from previous stage is used as IV for the current stage. The attacker launches XSS attack to steal the client cookies, which contain the encryption key, So without the actual IV, most of the plaintext is retrieved and the remaining cipher text is decrypted by known text attack.

R. RC4 attack

RC4 based ciphers have weak with confusion property, similar patterns are found redundant in the ciphertext. This makes the encrypted application data vulnerable to known ciphertext attack [24, 25]. The vulnerabilities in RC4 implementation include the following:

1. A broadcast attack launched using several unique keys for encryption on the same plaintext shows similarities in the ciphertext.
2. Pseudo-random number generation algorithm used as seed for RC4 can is vulnerable by performing analysis on the large set of ciphertext obtained as a

result of repeated encryption of same plaintext over the same key.

The MITM understands the nature of encryption in the underlying implementation and launches a XSS attack. The MITM hijacks the cookies and uses the information to cryptanalyze and decrypt the data.

S. CRIME attack

Compression Ratio Information leak Made Easy (CRIME) [27] is based on lossless compression vulnerability. HTTP uses lossless compression algorithms such as GZIP for compressing application data. This algorithm adds redundancy to the compressed plaintext. After encryption, the ciphertext shows some similarity for that traces back to its corresponding plaintext. This fails the property of confusion and makes it vulnerable to cryptanalysis. The MITM hijacks the HTTP cookies from client browser for cryptanalysis, which contains information about compression ratio and redundancy is revealed. This leads to failure of SSL/TLS protocol.

Table 2. CVE ID for Attacks on SSL/TLS Protocol

SSL/TLS Attack	CVE ID
Bleichenbacher #1 attack	CVE-2003-0147
Version rollback	CVE-2005-2969
Renegotiation attack	CVE-2009-3555
Cipher suite attack	CVE-2010-4180
STARTTLS injection	CVE-2011-0411
BEAST attack	CVE-2011-3389
CRIME attack	CVE-2012-4929
Lucky 13 attack	CVE-2013-0169
Heart bleed attack	CVE-2014-0160
CCS injection	CVE-2014-0224
Triple handshake attack	CVE-2014-1295
Poodle attack	CVE-2014-3566
Padding oracle attack	CVE-2014-8730

T. BREACH attack

Browser Reconnaissance and Exfiltration via Adaptive Compression of HTTP (BREACH) attack is an instance of CRIME attack on HTTP compression methods implementing DEFLATE algorithm. After hijacking HTTP cookies from the client browser, the MITM uses the data in the cookies to comprehend the redundancy pattern in the compression. By regressive brute force attacks, the MITM obtains uncompressed plaintext application data. This fails the confidentiality property of the protocol.

The attacks against SSL/TLS protocol are registered and cited with an index called the Common Vulnerabilities and Exploits (CVE) ID [28]. Table 2 lists registered attacks and their CVE IDs. Not all attacks have a CVE index.

IV. Mitigation Against SSL/TLS Attacks

The attacks against the SSL/TLS protocol were discussed in the previous section. Mitigations to the attacks have been devised to various attacks against the protocol. A few policies such as renegotiation and version rollback have avoided, modifications were bought to protocol design and server policy and new rules were added to rectify the vulnerability. Some solutions like avoiding renegotiation, avoiding legacy ciphers, etc. mitigate multiple vulnerabilities.

A. HTTP Strict Transport Security (HSTS)

HSTS enforces the client to use HTTP-over-Secure-Transport policy [29] to exchange application data. The HSTS enforcement are included in its HTTP response header. The STS header field follows an Augmented Backus-Naur Form (ABNF) syntax which is described as follows:

$$STS = "Strict-Transport-Security" ":" [directive] *(";" [directive]) \quad (2)$$

$$directive = directive-name ["=" directive-value] \quad (3)$$

$$directive-name = token \quad (4)$$

$$directive-value = token \mid quoted-string \quad (5)$$

Initially, the client request is sent to the server in one of the two forms:

1. HTTP-over-Secure-Transport Request - Server is a known host
2. HTTP Request – Server is an unknown host.

Also, closing the insecure ports in the server also prevents application layer attack.

B. TLS fallback cipher suite

Server sets a threshold version to SSL/TLS that is permissible. Any request with version older than the threshold version will be rejected. To enforce this, the server expects a cipher suite with ID 0x5600 in client hello. The version rollback is prevented by blocking the hello message with unacceptable version, resulting in timeout. The cipher suite TLS_FALLBACK_SCSV identified with 0x5600 [1] is a special cipher suite used for preventing version rollback. The current threshold is TLS v1.0. Any request/response whose version is below TLS v1.0 will be rejected by the server and client browsers.

C. TLS empty renegotiation cipher suite

Renegotiation policy is denied using a cipher suite with ID 0x00FF. Server on seeing this cipher suite ignores to process any information included for renegotiation. So, for each new session, a fresh negotiation is made

mandatory. The client hello includes the special cipher suite TLS_EMPTY_RENEGOTIATION _INFO_SCSV [30] identified by 0x00FF. This informs the server that client is not interested in session resumption and requests a new negotiation phase. Some server implementations do not support renegotiation and expect the cipher suite in client hello. Failing to include it will cause rejection of handshake.

D. Avoiding data compression

Compression of application layer data at presentation layer leads to (1) Adding of redundant substrings making encryption weak, or (2) Elongation of message rather than being compressed due to redundancy. Therefore, most servers and latest client browsers avoid compression on application layer data. Hence, hello messages do not include any compression methods mentioned.

E. Authenticated Encryption cipher suites

The most prominent ciphers used for encryption are RC4 and CBC based (DES with CBC, AES with CBC and similar ciphers). CBC based ciphers were vulnerable to chosen ciphertext attack and RC4 encrypted ciphertext shows pattern redundancy, making them vulnerable to serious cryptanalysis. Authenticated Encryption ciphers (AEAD) [31, 32] are resistant to cryptanalysis. These ciphers show high degree of randomness and are resistant to preimage attack. The common AEAD ciphers include the following:

1. Advanced Encryption Standards in CC Mode (AES-CCM)
2. Advanced Encryption Standards in Galois Counter Mode (AES-GCM)

F. Forwarded Secrecy

Forwarded secrecy is the property of a cipher suite that the session key, for any given session, is not revealed even if one of the secret key or symmetric key (depends on cipher suite) used for negotiation and exchange is compromised. The following cipher suites provide forwarded secrecy:

1. Diffie-Hellmann Ephemeral (DHE)
2. Elliptic Curve Diffie-Hellmann Ephemeral (ECDHE)

The ephemeral parameters ensure the security of the session key. Hence, these ciphers mitigate BEAST and RC4 vulnerabilities.

G. No explicit alert messages

As the MITM extract some useful information from the alert message describing the nature of error in the ciphertext sent to server, the alert message mechanism is modified. The server responds to existence of error in the incoming message but does not reveal any information on the nature of the error. Hence, the MITM cannot misuse the server as an oracle to cryptanalyze the ciphertext.

H. Enforcing client authentication

Client validation [33, 34] is neglected by several implementations of the protocol as not all service providers have clients holding X.509 certificates. Client verification is neglected and it leads to attacks triple handshake attack and bleichenbacher forgery.

Table 3. Attacks and Mitigations

Attacks	Mitigation
SSL Stripping & STARTTLS injection	HTTP Strict transport security
Version Rollback & Poodle	TLS fallback cipher suite
Renegotiation Attack	TLS no renegotiation cipher suite
Triple Handshake Attack	TLS fallback cipher suite & Enforcing client authentication
BEAST & RC4 Attack	AEAD cipher suites &Forwarded Secrecy
CRIME & BREACH	Avoiding data compression
Bleichenbacher v1.0	AEAD cipher suites
Padding Oracle	No explicit alert messages
Virtual Host Confusion	Enforcing client authentication

The following are the mechanisms to implement client authentication without certificates:

1. Elgamal Encryption
2. Session Aware User Authentication

The following are the steps involved in *Elgamal Encryption*:

1. A random value 'x' is generated by the server and encrypted with client public key as $E(x)$.
2. Salt is calculated as $T_S(x) \bmod N$.
3. $(E(x), T_S(x) \bmod N, N)$ is sent to client.
4. Client decrypts $E(x)$ to find x.
5. Authentication token is calculated

$$AT = hash(salt \,|username|\, password) \tag{6}$$

6. $Tr(Ts(x)) \bmod N$ is the premaster secret.
7. $(AT, T_r(T_S(x)))$ is sent to server by client.

If the specified username exists in server database, communication proceeds. Otherwise, the connection is terminated. In *Session Aware User Authentication*, every user has a personal PIN known to the server.

1. Client sends User Authentication Code (UAC) to the server on certificate request. UAC is generated as given below:

$$UAC = f(NT, PINU) \tag{7}$$

$$NT = HMAC(hash) \quad (8)$$

where, 'hash' is hash values of previously exchanged messages.

2. On authentication, a ticket is framed by the server and sent to client. The ticket is generates as given below:

$$T = \left(\left(K, K^{-1}\right), SN_T, K_T\right) \quad (9)$$

where,

$$K_T = E_{MasterSecret}\left(SN_T\right)$$

$\left(K, K^{-1}\right)$ - Impersonal asymmetric key pair

SN_T - Serial Number of ticket

A separate port is required for exchanging the UAC [34]. One Time Password (OTP) can also be used instead of hash, which is generated by a trusted third party.

Table 3 shows the relationship between an attack and the associated mitigation technique. The following is list of attacks for which proper mitigations are not yet available:

1. CCS Manipulation attack
2. Truncation attack
3. TIME attack
4. Lucky 13 attack
5. Heart bleed attack
6. Berserk attack
7. Bleichenbacher certificate forgery

V. Conclusion

In this paper, we discusses about the vulnerabilities in SSL/TLS protocol, the attacks against the protocol and mitigation techniques available. The vulnerabilities in SSL/TLS are due to the legacy cryptographic algorithms, weakness in some stages of the protocol such as CCS message exchange and policies adopted by the server. Legacy encryption algorithms such as CBC and RC4 make the ciphertext vulnerable to cryptanalysis. Policies such as renegotiation and version rollback make the protocol vulnerable. HTTP, a stateless protocol is transformed into HTTPS (a stateful protocol) using cookies, URL rewriting, etc. Due to stateful nature, the cookies are misused and replayed during SSL/TLS communication. Goals such as data confidentiality and authentication must be achieved by SSL/TLS. In this paper, we have presented a comprehensive survey of vulnerabilities, novel taxonomy of SSL/TLS attacks and their defense solutions. Based on the comprehensive analysis, it is evident that the impacting change is related to several vulnerabilities in the SSL/TLS protocol. Because of the vulnerabilities, none of the current versions of SSL/TLS meets the assured security goals.

References

[1] S. Turner and T. Polk, "Prohibiting Secure Sockets Layer (SSL) Version 2.0", RFC 6176, IETF, March 2011.

[2] A. Freier, P. Karlton and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0", RFC 6101, IETF, August 2011.

[3] T. Dierks and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, IETF, January 1999.

[4] T. Deirks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, IETF, August 2008.

[5] M. Myers et al, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP", RFC 2560, IETF, June 1999.

[6] M.H. Momani and A.Z. Hudaib, "Comparative Analysis of Open-SSL Vulnerabilities & Heartbleed Exploit Detection", International Journal of Computer Science and Security, vol 8, issue 4, 2014, pp 159 -176.

[7] www.heartbleed.com as seen on July 2015.

[8] Y. Sheffer and R. Holz, "Summarizing Known Attacks on Transport Layer Security (TLS)", RFC 7457, IETF, February 2015.

[9] M.L. Das and N. Samdaria, "On the security of SSL/TLS-enabled applications", Elsevier Applied Computing and Informatics, 2014, pp 68 – 81.

[10] M. Marlinspike,"New Tricks for Defeating SSL in Practice", BlackHat Security, 2009

[11] www.thoughtcrime.org/software/sslstrip as seen on July 2015.

[12] en.wikipedia.org/wiki/STARTTLS as seen on July 2015.

[13] Y. Suga, "SSL/TLS Status Survey in Japan – Transitioning against the Renegotiation Vulnerability and Short RSA Key Length Problem", IEEE Seventh Asia Joint Conference on Information Security, 2012, pp 17 – 24.

[14] K. Bhargavan et al, "Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS", IEEE Symposium on Security and Privacy, 2014, pp 1 – 17.

[15] D. Brumley and D. Boneh, "Remote timing attacks are practical," Proceedings of the 12th conference on USENIX Security Symposium - Volume 12, June 2003, pp 950-970.

[16] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology, UK: Springer-Verlag, August 1996, pp 104-113.

[17] N. AlFardan and K.G. Paterson, "Lucky thirteen: Breaking the TLS and DTLS record protocols", IEEE Symposium on Security and Privacy, 2013, pp 526-540.

[18] D. Bleichenbacher, "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1", Springer-Verlag Berlin Heidelberg, 1998, pp 1- 12.

[19] V. Klıma, O. Pokorny, and T. Rosa, "Attacking RSA-Based Sessions in SSL/TLS," in Cryptographic Hardware and Embedded Systems, Springer Berlin Heidelberg, vol. 2779, Sep. 2003, pp 426 – 440.

[20] J.Salowey et al, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, IETF, May 2006.

[21] J. Rizzo and T. Duong, "Practical padding oracle attacks", Proceedings of the 4th USENIX conference on Offensive technologies, vol. 10, 2010, pp 881 – 885.

[22] B. Moller, T. Duong and K. Kotowicz, "This POODLE Bites: Exploiting the SSL 3.0 Fallback", pp 1 – 4.

[23] www.intelsecurity.com/advanced-threat-research/berserk.html as seen on July 2015.
[24] S. Blake-Wilson et al, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, IETF, May 2006.
[25] J. Merkle and M. Lochter, "Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS)", RFC 7027, IETF, October 2013.
[26] T. Izu, M. Takenaka and T. Shimoyama, "Analysis on Bleichenbacher's Forgery Attack", IEEE Second International Conference on Availability, Reliability and Security, 2007, pp 1167 – 1174.
[27] J. Rizzo and T. Duong, "The CRIME attack", ekoparty Security Conference, vol. 8, 2012, pp 1 - 23
[28] cve.mitre.org/cve as seen on July 2015.
[29] J. Hodges and C. Jackson, "HTTP Strict Transport Security (HSTS)", RFC 6797, IETF, November 2012.
[30] E. Rescorla, M. Ray and S. Dispensa, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, IETF, February 2010.
[31] E. Rescorla, "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, IETF, August 2008.
[32] P. Chown, "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, IETF, June 2002.
[33] K. Bhargavan et al, "Implementing TLS with Verified Cryptographic Security", IEEE Symposium on Security and Privacy, 2013, pp 445 – 459.
[34] R. Oppligera, R. Hauserb and D. Basinc, "SSL/TLS session-aware user authentication revisited", Computers & Security, 2008, pp 467-476.

Authors' Profiles

Keerthi Vasan, K received a Bachelor of Technology degree in Information Technology from Tamilnadu College of Engineering under Anna University, Chennai. He is currently pursuing Master of Technology degree in Computer Science and Engineering from National Institute of Technology (NIT) Puducherry, Karaikal, India.

Dr. Arun Raj Kumar, P is working as Assistant Professor in the Computer Science and Engineering Department at National Institute of Technology (NIT) Puducherry, Karaikal. He completed Ph.D. in Computer Science and Engineering at National Institute of Technology (NIT) Tiruchirappalli, India. His research interests include Computer Networks, Network Security, Machine Learning, and Wireless Sensor Networks. He has published 11 papers in reputed and refereed International Journals and IEEE Conferences. He is also invited reviewer for Journals such as International Conferences and International Journal of Network Security (IJNS), etc.

How to cite this paper: Keerthi Vasan K., Arun Raj Kumar P.,"Taxonomy of SSL/TLS Attacks", International Journal of Computer Network and Information Security(IJCNIS), Vol.8, No.2, pp.15-24, 2016.DOI: 10.5815/ijcnis.2016.02.02

