Attacking SSL/TLS Implementations

Jemy Amrutia
Information Systems Security
Concordia University
Montreal, Canada
jemyamrutia@mail.com

Vani Kaithi
Information Systems Systems
Concordia University
Montreal, Canada
vanikaithic@gmail.com

Sarat Sai Bammidi
Information Systems Securty
Concordia University
Montreal, Canada.
sai.sarat2@gmail.com

Sai Venkata Lakshmi Soumya Challa
Information Systems Security
Concordia University
Montreal, Canada.
soumyasaic@gmail.com

Ekasmeet Gahir
Information Systems Security
Concordia University
Montreal, Canada.
ekasmeet97@gmail.com

Sai Sandeep Vendra
Information Systems Security
Concordia University
Montreal, Canada.
Sandeepvendra1404@gmail.com

Mannam Venkatesh
Information Systems Security
Concordia University
Montreal, Canada
venkateshmannam22@gmail.com

Sai Harshitha Velamuri
Information Systems Security
Concordia University
Montreal, Canada.
harshithalalitha98@gmail.com

Pranathi Akula
Information Systems Security
Concordia University
Montreal, Canada
akula.pranathi98@gmail.com

Shawn Philip Babu
Information Systems Security
Concordia University
Montreal, Canada.
shawn0111babu@gmail.com

***Abstract*—SSL stands for Secure Sockets Layer and, it's the standard technology for keeping an internet connection secure and safeguarding any sensitive data that is being sent between two systems, preventing criminals from reading and modifying any information transferred, including potential personal details. TLS (Transport Layer Security) is just an updated, more secure, version of SSL. We still refer to security certificates as SSL because it is a more commonly used term, but when you are buying SSL from DigiCert you are actually buying the most up to date TLS certificates with the option of ECC, RSA or DSA encryption. However, there are undeniable differences between the libraries that implement SSL/TLS protocol and vulnerabilities in these libraries. Hence, the two main questions asked are: what's the difference between TLS vs SSL? And is it something we need to worry about? In this report, we summarize some of the limitations by considering implementations of each along with review of past protocol-based and software-based vulnerabilities.**

I. INTRODUCTION

The Secure Sockets Layer (SSL) protocol is a security technology that offers online privacy. The protocol enables secure communication between client and server applications. Clients can choose to authenticate themselves, while servers are always authenticated. There is a big demand right now for SSL certificates. The encryption landscape has undergone a substantial transformation since Google launched its "HTTPS Everywhere" campaign. Because not all servers provide web interfaces for managing SSL, OpenSSL can be the only choice for some systems to import and setup your certificate. OpenSSL is quite helpful if we want to speed up the entire process or don't have access to an online administration panel. With just a few OpenSSL lines, it is possible to generate the Certificate Signing Request and the private key, combine files, review the certificate's specifics, and address any potential issues. An outdated cryptographic technique that encrypts network interactions between two computer applications is called Secure Sockets Layer, or SSL. Transmission Layer Security) certificates are what we mean when we talk about SSL certificates. OpenSSL is a cryptographic tool that uses command lines to control SSL/TLS certificate creation, installation, and identification.

A web server (i.e., a website) that is SSL-secured is attempted to be contacted by a browser or server. The browser/server asks for the web server's identity. The browser or server is then sent a copy of the web server's SSL certificate. This check determines if a browser or server trusts the SSL certificate. If so, a message alerts the web server. The web server sends back a digitally signed acknowledgement to start an SSL encrypted

session. Eventually, encrypted data is exchanged between the web server and browser/server. Website owners must get an SSL certificate from a certificate authority and install it on the web server (often a web host can handle this process). An impartial third party that may attest to the legitimacy of the website owner is known as a certificate authority. They maintain records of the certifications they issue. Online payments conducted via a credit card or another way are included in the SSL apps. Intranet-based traffic includes, but is not limited to, internal networks, file sharing, extranets, and database access. Webmail servers include Exchange, Office Communications Server, and Outlook Web Access. The connection between an email client like Microsoft Outlook and a mail server like Microsoft Exchange.

SSL Benefits and Drawbacks: -

Benefits: -

1.Improving Data Security: -

Stronger data security is one of the major advantages of SSL certificates for websites. They protect the data in transit while securing connections between the browser and server thanks to cryptography. Even if there is a data breach, the attacker won't be able to fully decrypt and comprehend all of the data because of how highly and intricately encrypted it is.

2.Identity Verification and Authentication: -

Data that is transmitted via the internet is moved between different organisations with a significant potential of ending up in the hands of intruders or other unauthorised third parties. The fact that SSL confirms and authenticates the parties' identities is another important benefit.

3. Avoid a Variety of Attacks: -

The ability of SSL certificates to protect websites and users from a variety of threats is another significant advantage. As it encrypts all data in transit, it helps prevent eavesdropping, impersonation, data theft, identity theft, and Man-in-the-Middle attacks.

Drawbacks: -

1.Cost: -

Free SSL certificates are offered, however they do not provide the necessary level of security and encryption. One must pick a CA who can offer the highest levels of protection while adhering to industry standards and rigorous compliance standards. This CA must have the necessary infrastructure, knowledge, and reputation. There is a price for this. Your fees rise if you have many domains and sub-domains.

2. Expiry : -

For continuous security, expired SSL certificates must have their expiration date monitored and renewed. In the absence of sight, the process of monitoring and renewal may be more difficult. Reputable suppliers like Entrust from Indusface offer cutting-edge Certificate Management Systems (CMS) to administer SSL easily and effectively in order to address this SSL drawback.

3.SSL-Related Vulnerabilities: -

Although SSL has many advantages for the website, it also has several security flaws. There are numerous vulnerabilities in the older SSL/TLS protocols, including POODLE, BEAST, Heartbleed, CRIME, and TLS 1.0 or TLS 1.1. Browsers label websites with certificates that use these antiquated protocols as unsafe.

Why the developers decided on new TLS implementation instead of already using open SSL library?

First off, developers can require particular features or components that are absent from the libraries that are currently available. Creating a custom TLS implementation from scratch allows developers to create a solution that is tailored to their needs. Second, developers can be concerned about the safety or calibre of the current libraries. Developers can have more control over the security and quality of the codebase with a new TLS implementation, which may be essential for applications that need security. Finally, developers may elect to employ a new TLS implementation as a teaching tool or as a means of better understanding the underlying technology. The process of developing a new implementation can help developers learn more about how TLS works and progress their careers.

TLS OVER SSL

TLS offers a more secure way to handle authentication and message exchange. TLS uses the more secure Key-Hashing for Message Authentication Code (HMAC) to ensure that a record cannot be changed during transmission over an open network like the Internet, whereas SSL uses keyed message authentication.TLS specifies the Improved Pseudorandom Function (PRF), which generates key information with the HMAC using two hash methods. By prohibiting data from being altered if only one algorithm is compromised, two algorithms strengthen security. As long as the second algorithm is not compromised, the data is secure.

TLS uses PRF and HMAC values in the message to provide a more secure authentication technique than SSL, which both send messages to each node to verify that the exchanged communications were not tampered with. the TLS protocol specifies the sort of certificate that must be sent between nodes in order to guarantee more consistency . Moreover,
TLS offers more detailed alerts concerning session issues and keeps track of when specific alarms are given.

They were identical for a while to make the switch from SSL to TLS easier. The only differences between SSL v3 and TLS 1.0 were a few minor protocol features. This made it simple to keep using SSL, as we had for so many years, rather than encouraging people to use the new TLS language. Some

individuals began to use the names interchangeably. The harm had already been done when TLS 1.1 (and later) replaced SSL as the only reliable protocols when it was phased out (deprecated).

What is the difference between HTTP and HTTPS?

HTTPS stands for "secure" and has a S in it. HTTP over SSL/TLS is just HTTPS. Traffic to and from a website with an HTTPS address is authenticated and encrypted using the SSL/TLS protocol, and that website has a valid SSL certificate that was issued by a certificate authority. Several web browsers have started to flag HTTP pages as "not secure" or "unsafe," in an effort to persuade users to switch to the more secure HTTPS protocol. Hence, HTTPS has become crucial for establishing user trust in addition to being necessary for keeping consumers safe and user data secure.

What is TLS

An encryption system called Transport Layer Security was created to provide end-to-end security for web-based communications. To prevent manipulation and eavesdropping, the Internet Engineering Task Force (IETF) adopted TLS as the default protocol.

Users and online apps frequently run into many potential security issues when accessing the internet. They consist of verifying the other party's identification, data manipulation, and third-party surveillance. TLS uses cryptographic methods to protect users while they browse online, help ensure the integrity of the data being exchanged, and authenticate the client or server in a connection.

TLS is often associated with secure web browsing, which guards against hackers and eavesdroppers during online transactions. The padlock icon at the top left corner of the web browser denotes secure browsing sessions. TLS is also utilised by programmes like email, file transfers, and audio and video conferencing. The vast majority of protocols, including HTTP, SMTP, FTP, XMPP, and many others, are also interoperable with TLS. Users should be aware that only data transported over the internet is intended to be secured by TLS; data on end systems is not.

How does TLS work?

To assist provide a secure connection between two or more interacting programmes, assure device interoperability, and run reasonably efficiently, TLS security is designed to use encryption on both the client and server sides.

The first step in client-server communication is choosing whether to use TLS protocols or not. There are numerous methods the client might specify a TLS connection. The client might, for instance, utilise a port that supports the encryption types used in TLS connections. Making a protocol-specific request to transition to a TLS connection is another possible approach.

The TLS protocol definition moves via two layers, the TLS handshake protocol and the TLS record protocol, once the client and server have decided to interact over TLS. Combining symmetric and asymmetric cryptography is a feature of TLS protocols. Asymmetric cryptography produces key pairs, one public (shared by the sender and receiver) and one private, as opposed to symmetric cryptography, which generates keys known to both the sender and recipient.

The TLS handshake protocol defines the standards necessary to exchange an application "message." Depending on the supported cypher suites and the key exchange mechanism in use, a TLS handshake involves a series of exchanges between the client and server that can go something like this:

- A client initiates a connection by sending a "client hello" message that includes a list of supported cypher suites (a group of encryption techniques needed to create a secure connection) and a client random, or random sequence of bytes.
- The chosen cypher suite, the chosen TLS protocol version (1.0, 1.2, etc.), and a random sequence of bytes (referred to as the "server random") are all included in the server's "server hello" message.
- For authentication, the server transmits the client its SSL certificate. When the server requests it, the client can also transmit a certificate for authentication in addition to authenticating the server by checking the SSL certificate.
- The "premaster secret," which is a second string of random bytes, is sent by the client. The premaster secret is first encrypted by the client using asymmetric cryptography to create a public key from the server's security certificate. Only the server has the private key necessary to decrypt the premaster secret.
- With the help of the private key, the server decrypts the premaster secret.
- Client and server both produce session keys using the premaster secret, client random, and server random.
- A "completed" message that has been encrypted using the session key is sent by the client.
- The server replies with an encrypted "done" message using the session key.
- Secure symmetric encryption has been accomplished by the client and server, concluding the handshake and allowing communication to proceed using the pre-agreed session keys.

The TLS record protocol uses symmetric cryptography to create distinct session keys for each connection after the decryption technique is decided upon during the handshake stage. These keys allow for ongoing communication throughout the session. The record protocol also adds a message authentication code based on hashing to any data being delivered (HMAC).

Users should anticipate using some computer power in the process because TLS's encryption methods are complicated.

TLS, however, also has internal mechanisms in place to avoid material lags. The performance and load times of online applications shouldn't be significantly impacted by TLS protocols, and most enterprises shouldn't see an increase in computing expenditures.

TLS Implementation

1. TLS STACK:
A TLS (Transport Layer Security) stack is a particular way to implement the TLS protocol, which enables secure internet connection. A TLS stack is often made up of several software layers that cooperate to offer secure communication between two endpoints.
A cryptographic library that offers cryptographic operations like encryption, decryption, hashing, and key exchange is a component of a TLS stack at the lowest level. The negotiation and initialization of the TLS session are taken care of by a protocol implementation layer. For secure communication, there is a record layer that offers message framing and fragmentation. The interface between the TLS stack and the application employing secure communication is provided by the application layer, in the end.

2. Cipher Suites:
A collection of cryptographic methods used in the TLS (Transport Layer Security) protocol are referred to as cypher suites.

● Algorithm for Key Exchange: This algorithm is employed to safely transfer cryptographic keys between the two endpoints. example: RSA and Diffie-Hellman (DH).
● Data transferred between the two endpoints is encrypted using an encryption technique, such as Advanced Encryption Standard (AES) or Triple Data Encryption Standard (3DES).
● The integrity of the data transported between the two endpoints is guaranteed by the Message Authentication Code (MAC) Algorithm.
● For instance, Hash-based Message Authentication Code (HMAC)
● The hashing method is used to make sure the data being sent between the two endpoints is legitimate. for example, SHA-256 and SHA-384.

3. TLS Configuration:
The parameters and settings used to create a secure connection between two endpoints using the TLS protocol are referred to as TLS (Transport Layer Security) configuration.
The TLS version that will be applied to the connection. TLS 1.2 and TLS 1.3 are versions that are frequently used. To authenticate endpoints and create trust between them, TLS uses digital certificates. Resuming previously started sessions is supported by TLS, which can boost efficiency and cut down on overhead. There are many other parameters that are part of the TLS setup, such as protocol-level settings, certificate revocation settings, and other security-related settings.

Purpose of TLS

Most websites now use TLS encryption as normal procedure to safeguard web apps from data manipulation and eavesdropping. The SSL/TLS protocols were created in response to the growing security risks and the requirement for encryption on both the client and server sides.

TLS is in place to support the protection of user security and privacy. Without TLS, sensitive data being exchanged online, including login credentials, personal data, and credit card numbers, is susceptible to theft. Moreover, emails, browsing patterns, and direct message interactions could all be observed by unidentified third parties.

TLS helps defend online applications against data breaches and distributed denial-of-service (DDoS) assaults in addition to protecting the personal information of individual users. For businesses of all sizes, data breaches and DDoS assaults can prove to be extremely expensive and harm consumer trust irreparably. It is simple to increase security and contribute to the protection of both user and corporate privacy by making sure web browsers are using TLS.

TLS support is now standard in the majority of browsers. Google Chrome, for instance, regularly warns people away from websites that are not HTTPS. User awareness of website security and the need to look for secure data transfer protocols is growing as a result. Organizations and people can contribute to ensuring a minimal level of shared protection for web-based activities by insisting on the required use of TLS in all web-based communications.

That being said, TLS protocols have been breached in the past ten years, including BEAST in 2011, CRIME in 2012, BREACH in 2013, and Heartbleed in 2014. Yet since then, TLS protocols have undergone significant updates aimed at removing problematic code and adding enhancements in security, performance, and privacy. The handshake step is accelerated by the most recent TLS version (1.3), which also speeds up the encryption process. Also, it gets rid of outdated algorithms that led to security holes in earlier iterations.

II. Limitation of SSL/TLS Implementations

Transport Layer Security (TLS) and Secure Sockets Layer (SSL) are cryptographic protocols designed to provide secure communication over the internet. While these protocols offer several benefits, they are not without limitations.

TLS STACK

TLS (Transport Layer Security) is a protocol used to provide secure communication over the internet.However, middleboxes can interfere with the normal operation of TLS by modifying or blocking TLS traffic.

TLS stack middlebox interference occurs when a network middlebox, such as a firewall or a proxy, interferes with the TLS handshake process between a client and server. The TLS handshake is the initial step in establishing a secure connection between two endpoints, where the client and server agree on a shared encryption key and negotiate the encryption protocol and cipher suite to be used for the secure communication.

Middleboxes can interfere with the TLS handshake by modifying the TLS packets, blocking certain TLS versions or cipher suites, or terminating TLS connections and creating new ones. For example, a middlebox might terminate a TLS connection and create a new one using a different cipher suite or key length, which can result in a downgrade attack or a vulnerability in the encryption.

To minimize the impact of TLS stack middlebox interference, TLS protocol designers and implementers can follow best practices, such as using the latest TLS versions and cipher suites, avoiding weak or deprecated encryption protocols, and using certificate pinning to prevent man-in-the-middle attacks. Network administrators can also configure middleboxes to allow TLS traffic that is needed for specific applications and protocols, and monitor the network for any signs of TLS stack middlebox interference.

LEGACY SYSTEMS

Some legacy systems may only support older versions of TLS/SSL, such as SSLv3 or TLS 1.0, which are now considered insecure and vulnerable to attacks. Newer versions of TLS/SSL, such as TLS 1.2 or TLS 1.3, provide stronger encryption and better security, but may not be supported by legacy systems.

Legacy systems may only support a limited number of cipher suites, which can restrict the strength of the encryption and the key exchange mechanisms available for secure communication.

Legacy systems may not support modern security features, such as certificate pinning, forward secrecy, or perfect forward secrecy, which can increase the security of the TLS/SSL connection.

Legacy systems may not have the processing power or memory required to support the latest TLS/SSL protocols and cipher suites, which can result in slower performance or connection failures.

CIPHER SUITES

TLS (Transport Layer Security) and its predecessor SSL (Secure Sockets Layer), cipher suites are sets of encryption algorithms, key exchange algorithms, and message authentication codes (MACs) used to secure communication between a client and server. When a TLS/SSL connection is established, the client and server negotiate a cipher suite to use for the secure communication.

The Encryption algorithm is used to encrypt the data being transmitted between the client and server. Common encryption algorithms used in TLS/SSL include Advanced Encryption Standard (AES), Triple Data Encryption Standard (3DES), and RC4.

The Key exchange algorithm is used to establish a shared secret key between the client and server, which is used for the encryption and decryption of data. Common key exchange algorithms used in TLS/SSL include RSA, Diffie-Hellman (DH), and Elliptic Curve Cryptography (ECC).

Additionally, Message authentication code (MAC): This code is used to verify the integrity of the data being transmitted between the client and server. Common MAC algorithms used in TLS/SSL include HMAC-SHA256 and HMAC-SHA384.

Some examples of cipher suites commonly used in TLS/SSL include:

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256: This cipher suite uses elliptic curve cryptography for key exchange, RSA for authentication, AES-128 for encryption, and SHA-256 for message authentication.

TLS_RSA_WITH_AES_256_CBC_SHA: This cipher suite uses RSA for key exchange and authentication, AES-256 for encryption, and SHA-1 for message authentication.

TLS_DHE_RSA_WITH_AES_128_CBC_SHA: This cipher suite uses Diffie-Hellman for key exchange, RSA for authentication, AES-128 for encryption, and SHA-1 for message authentication.

Older TLS/SSL versions may not support newer, more secure cipher suites. In addition, some older systems may not support certain cipher suites, which can limit the ability to establish a secure connection. Some cipher suites, particularly those that use more complex encryption algorithms or key exchange algorithms, can have a performance impact on the connection. This can result in slower response times or increased latency.

To address these limitations, it's important to use modern and secure cipher suites that are compatible with both the client and server, avoid weak or deprecated cipher suites, and monitor the network for any signs of malicious activity. Also, implementing additional security measures such as certificate pinning, strict transport security (HSTS), or perfect forward secrecy (PFS) can provide additional protection against attacks and improve the security of TLS/SSL communication.

Some of the more limitations of TLS and SSL implementations include:

Outdated cryptographic algorithms: SSL and early versions of TLS use cryptographic algorithms that are now considered

weak and vulnerable to attacks. For example, SSLv3 and TLS 1.0 use the RC4 encryption algorithm, which is known to be susceptible to attacks.

Certificate revocation: The process of revoking a certificate when it is compromised or no longer valid is not always reliable. Certificate revocation is often not checked by clients, which can result in communication with a compromised server.

Key management: TLS and SSL rely on public-key cryptography, which requires proper key management. If the private key used for encryption is compromised, it can lead to a breach of the entire system. Similarly, if the key exchange process is not secure, it can lead to a man-in-the-middle attack.

Compatibility issues: TLS and SSL implementations may not be compatible with all web browsers and servers. This can result in interoperability issues and may limit the ability of users to access secure websites.

Performance: Implementing SSL and TLS can have an impact on the performance of web applications. This is because the encryption and decryption process can increase the processing time required for each request.

Implementation flaws: Implementation flaws in SSL/TLS libraries and applications can lead to security vulnerabilities. For example, the OpenSSL library, which is used by many applications, has had several critical vulnerabilities in the past.

False sense of security: SSL/TLS can create a false sense of security, leading to users being less cautious about their online behavior. Users may assume that their communication is secure when it is not, leading to potential security breaches.

Interception: SSL/TLS communication can be intercepted by attackers using man-in-the-middle attacks. This can allow attackers to steal sensitive information or modify the communication without detection.

Limited protection against some types of attacks: SSL/TLS provides protection against eavesdropping and tampering attacks, but it does not provide protection against attacks such as DNS spoofing and phishing.

Limited Cryptographic Strength: The cryptographic strength of SSL and TLS implementations can also be a limitation. For example, the use of weak key lengths or inadequate key exchange algorithms can make encrypted communication vulnerable to attacks.

Difficulty in Configuration: Configuring SSL and TLS properly can be challenging, and misconfigurations can lead to vulnerabilities. For example, disabling certain encryption ciphers or enabling weak ciphers can expose encrypted communication to attacks.

Performance Overhead: SSL and TLS encryption can introduce a performance overhead, which can be significant in high-traffic environments. This overhead can impact the user experience and can also make SSL and TLS impractical for certain applications.

Limited trust model: SSL and TLS rely on a centralized trust model, where certificates issued by trusted Certificate Authorities (CAs) are used to verify the identity of servers. However, this model is not perfect, and there have been instances where CAs have issued fraudulent certificates, which can compromise the security of encrypted connections.

Limited protection against endpoint compromise: TLS and SSL only protect the communication channel between the client and the server. If either endpoint is compromised, the communication can be intercepted or tampered with.

Lack of perfect forward secrecy: Perfect forward secrecy (PFS) is a property that ensures that even if an attacker obtains the private key of a server, they cannot decrypt past communications. While some TLS and SSL implementations support PFS, it is not always enabled by default, and not all cipher suites support it.

Limited protection against side-channel attacks: Side-channel attacks exploit information leaked through the implementation of a cryptographic protocol rather than attacking the protocol itself. While TLS and SSL implementations are designed to be resistant to side-channel attacks, they are not immune to them.

Cost: Implementing SSL and TLS can be costly for organizations, especially for those that require high levels of security. This is because they may need to purchase digital certificates and hardware security modules to ensure proper key management.

In conclusion, while TLS and SSL provide a secure means of communication over the internet, their implementations are not without limitations. Organizations should be aware of these limitations and take appropriate measures to mitigate the risks associated with them.

III. HOW SSL/TLS SECURE DATA ?

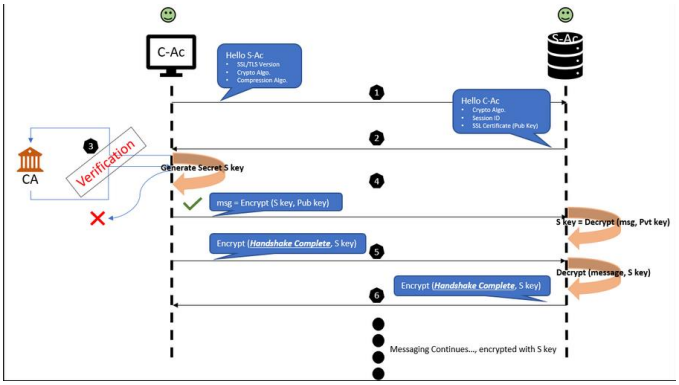
When it comes to transferring data via the internet in today's digital era, security is a huge problem. TLS (Transport Layer Security) and SSL (Secure Sockets Layer) are protocols used to encrypt communication between two devices over the internet. They use encryption, authentication, and integrity checking techniques to offer a safe channel for data transfer. In this report, we will look at how TLS and SSL operate together to protect data.

Encryption: Encryption is a major function of TLS and SSL. The process of converting plain text into ciphertext, which is

unreadable without the correct key, is known as encryption. TLS and SSL encrypt data using symmetric encryption, which means that the sender and receiver share a secret key that is used to encrypt and decode the data.

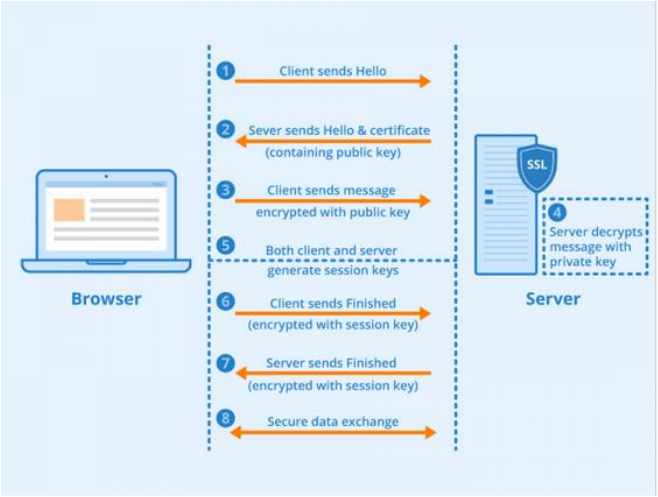
Authentication: TLS and SSL also enable authentication to verify that data is only delivered between the parties intended. To authenticate the persons participating in the connection, TLS and SSL require digital certificates. Digital certificates are issued by trusted certificate authority and contain information about the parties' identities.

Integrity: Another critical component of TLS and SSL is integrity. The integrity of the data assures that it has not been tampered with during transmission. TLS and SSL implement message digests to ensure the data's integrity. A hash function, which is a mathematical function that accepts an input and returns a fixed-length output, is used to generate message digests. Because the hash function's output is unique for each input, even little changes in the input data result in a completely new hash value. The message digest is delivered with the data, and the receiver may use the same hash algorithm to ensure that the data is not tampered with.



TLS Handshake: The TLS handshake is the process by which two devices establish a secure connection. The TLS handshake involves the following steps:

1. The client initiates the TLS handshake by submitting a request to connect to the server.
2. The server responds with its digital certificate, which provides the public key of the server.
3. The client validates the digital certificate issued by the server to confirm that it is legitimate and issued by a trusted certificate authority.
4. The client generates a random symmetric key and encrypts it with the public key of the server.
5. The server uses its private key to decode the symmetric key.
6. All throughout the session, both the client and the server utilize the symmetric key to encrypt and decode data.



Certificate Validation: Certificate validation is the process of verifying the authenticity and validity of digital certificates used for secure communication. A digital certificate is issued by a trusted Certificate Authority (CA) and contains information about the entity using it. Certificate validation ensures that the certificate is not compromised and is valid. It prevents unauthorized access to communication and ensures the privacy and security of sensitive data.

Key Exchange: Key exchange is the process of securely exchanging keys between the sender and receiver to ensure that the data transmitted between them is encrypted and cannot be intercepted. The key exchange process involves generating a unique symmetric key that is used to encrypt and decrypt data during the session. The key exchange process ensures that only the sender and receiver have access to the symmetric key, and it prevents unauthorized access to the communication.

Data Encryption: Data encryption is the process of transforming plain text into encrypted text, which cannot be deciphered without the right key. Even if an attacker intercepts the data, they won't be able to read it because of the encryption procedure. Data encryption preserves the secrecy and privacy of sensitive information and is a key component of secure communication.

Session Termination: The process of terminating a secure communication connection is known as session termination. The data transferred during the communication session is inaccessible after the communication session ends because the symmetric key used for encryption and decryption is destroyed.

IV. Comparisions of Different versions of SSL/TLS

TLS (Transport Layer Security) and OpenSSL are both important cryptographic protocols used to secure communication on the internet. OpenSSL is a widely used open-source implementation of SSL/TLS protocols. Here is a comparison of different versions of TLS and OpenSSL:

TLS 1.0 - This version of TLS is now considered insecure due to several vulnerabilities, including POODLE and BEAST attacks. OpenSSL 1.0.1g or later supports TLS 1.0.

TLS 1.1 - This version of TLS addresses some of the vulnerabilities present in TLS 1.0, but it is also considered insecure due to certain vulnerabilities such as Lucky13. OpenSSL 1.0.1g or later supports TLS 1.1.

TLS 1.2 - This version of TLS is currently the most widely used and is considered secure. It has improved security features compared to TLS 1.1, such as stronger cipher suites, and support for authenticated encryption with associated data (AEAD). OpenSSL 1.0.1g or later supports TLS 1.2.

TLS 1.3 - This version of TLS is the latest and most secure version of TLS. It provides better security and performance compared to TLS 1.2, including faster handshakes and improved forward secrecy. OpenSSL 1.1.1 or later supports TLS 1.3.

OpenSSL also has different versions, and the latest version is OpenSSL 3.0.0.

OpenSSL 1.0.2 - This version is no longer supported and has reached its end of life. It supports up to TLS 1.2.

OpenSSL 1.1.0 - This version introduced support for TLS 1.3 and Elliptic Curve Cryptography (ECC).

OpenSSL 1.1.1 - This version introduced several improvements, including support for TLS 1.3, ChaCha20-Poly1305 cipher suites, and Ed25519 and Ed448 elliptic curves.

Some of the major differences are listed below based on the characteristics

Cipher suites
SSL protocol offers support for Fortezza cipher suite. TLS does not offer support. TLS follows a better standardization process that makes defining of new cipher suites easier like RC4, Triple DES, AES, IDEA, etc.

Alert messages
SSL has the "No certificate" alert message. TLS protocol removes the alert message and replaces it with several other alert messages.

Record Protocol
SSL uses Message Authentication Code (MAC) after encrypting each message while TLS on the other hand uses HMAC — a hash-based message authentication code after each message encryption.

Handshake process
In SSL, the hash calculation also comprises the master secret and pad while in TLS, the hashes are calculated over handshake message.

Message Authentication
SSL message authentication adjoins the key details and application data in ad-hoc way while TLS version relies on HMAC Hash-based Message Authentication Code.

TABLE

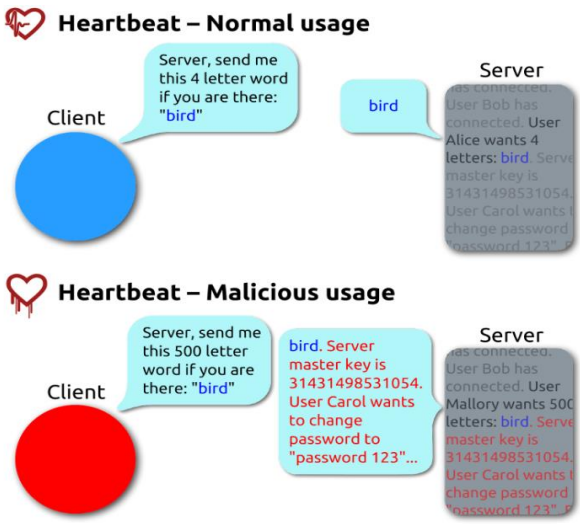
Version	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	OpenSSL 0.9.8	OpenSSL 1.0.1	OpenSSL 1.1.1
Release Date	1999	2006	2008	2018	2005	2012	2018
Security	Weak	Medium	Strong	Very Strong	Weak	Medium	Strong
Cipher Suites	Limited	Expanded	Expanded	Expanded	Limited	Expanded	Expanded
Handshake	Slow	Faster	Faster	Fastest	Slow	Faster	Fastest
Certificate Handling	Limited	Expanded	Expanded	Expanded	Limited	Expanded	Expanded
Support	Widely supported	Widely supported	Widely supported	Limited support	Widely supported	Limited support	Widely supported

V. VULNERABILITIES IN SSL/TLS-PROTOCOL BASED

Heartbleed attack:

Heartbleed is a security vulnerability that was discovered in April 2014 and has been widely regarded as one of the most significant security breaches in recent years. It affected the popular OpenSSL cryptographic software library, which is used to secure communications over the Internet, including email, messaging, and online transactions.

Working:



The Heartbleed attack exploits a flaw in the OpenSSL implementation of the Transport Layer Security (TLS) heartbeat extension. This extension is used to check if a connection between a client and a server is still active. The server sends a small message, known as a heartbeat message, to the client, which responds with the same message to indicate that the connection is still active.

The vulnerability in the OpenSSL implementation allowed an attacker to send a malformed heartbeat message to a server running a vulnerable version of OpenSSL. This malformed message could trick the server into sending back a larger response than intended, which could include sensitive information from the server's memory.

The attacker could then retrieve this sensitive information, such as passwords, credit card numbers, and other data, from the server's memory. This attack could be repeated multiple times, allowing the attacker to gather large amounts of sensitive information.

Mitigation:

1. Applying security patches
2. Changing passwords

POODLE attack:

POODLE, which stands for **Padding Oracle On Downgraded Legacy Encryption**, is a security vulnerability that was discovered in October 2014. It is a flaw in the design of SSL 3.0, an older version of the SSL/TLS encryption protocol that is used to secure Internet communications.

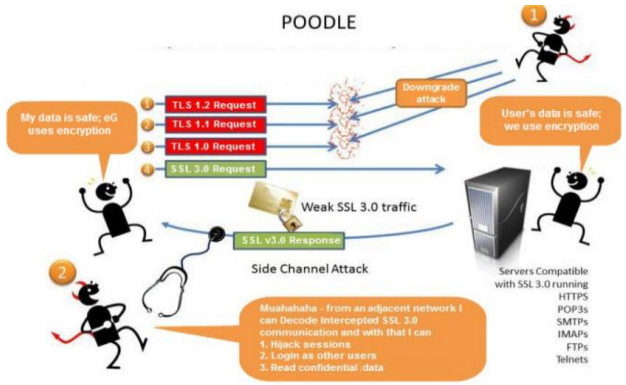
Working:

The POODLE attack exploits a flaw in the design of SSL 3.0, an older version of the SSL/TLS encryption protocol. In SSL 3.0, a block cipher mode known as CBC (cipher block chaining) is used to encrypt data. CBC encryption requires padding the plain-text to a multiple of the block size before encryption. The padding is then removed after decryption.

The POODLE attack works by exploiting a vulnerability in the way that SSL 3.0 handles padding. An attacker intercepts an encrypted message that is being transmitted between a client and a server using SSL 3.0. The attacker then modifies the padding bytes in the message to try to decrypt the encrypted data one byte at a time.

The attacker sends the modified message to the server, which attempts to decrypt it. If the padding is invalid, the server will send an error message back to the attacker. However, the error message contains information that the attacker can use to determine the decrypted value of one byte of the encrypted data.

By repeating this process multiple times, the attacker can gradually decrypt the entire message. This can allow the attacker to gain access to sensitive information, such as login credentials and personal data.



Mitigation:

1) Disable SSL 3.0
2) Use TLS_FALLBACK_SCSV
3) Implement Perfect Forward Secrecy (PFS)
4) Keep systems and software up-to-date

CRIME(CVE-2012-4929):

Compression Ratio Info-leak Made Easy (CRIME) is a security exploit against secret web cookies over connections using the HTTPS and SPDY protocols that also use data compression. It enables an attacker to perform session hijacking on an authenticated web session, enabling the start of additional attacks, when used to retrieve the content of private authentication cookies. Although CRIME is a client-side attack, the client can be protected by the server by asking it not to use feature combinations that can be exploited. The drawback of CRIME is Deflate compression. This alert is issued if the server accepts Deflate compression.

Working:

Cybercriminals can take advantage of a flaw in the SSL/TLS protocol and the SPDY protocol's compression method to carry out a CRIME attack by decrypting the HTTPS cookies that a website has established. This can then compel a user to view a malicious website while the attack is being carried out by forcing their browser to forward HTTPS requests to that site. Afterward, the attackers control the path for new requests. Cybercriminals have access to information about the client browser's sent ciphertext size. They can then observe how the size of the compressed request payload, which includes the hidden cookie sent by the browser and the harmful content injection, varies. When the compressed content gets smaller, it means that some of the confidential content they want access to have probably been partially mirrored by the injected content. The value of the user's session cookie may be ascertained by observing the fluctuation in length, compression ratio, or varying content.

FREAK(CVE-2015-0204):

The term "FREAK vulnerability" describes a flaw in the Secure Sockets Layer (SSL)/Transport Layer Security (TLS) protocols brought on using "export-grade" encryption. The term is an acronym for "Factoring RSA Export Keys."

Working:

The FREAK vulnerability enables hackers to intercept HTTPS communications between clients and susceptible websites in order to obtain a website's secret key. As a result, they are now able to decode HTTPS connection logon cookies, passwords, credit card data, and other susceptible data.

Because the client is required to use a "export-grade" key or 512-bit export RSA key, which is much simpler to trace and crack than current encryption standards, private communications are essentially in danger.

How exactly does this work? An attacker can ask for ‘export RSA’ instead of the standard RSA cipher suites through the client’s Hello message. The server then answers with a 512-bit-long export cipher key instead of today’s high-security keys. The response is signed with its long-term key.

The website client takes in the weak ‘export-grade’ key, allowing the Man-in-the-Middle attacker to get the RSA decryption key and use the ‘pre-master secret’ to gain access to the TLS’ master secret’, which is employed for symmetric encryption of messages in the connection. Afterward, the attacker can inject malicious code into the plaintext file — the essence of command injection risks.

DROWN(CVE-2016-0800):

HTTPS and other services that depend on SSL and TLS, two crucial cryptographic protocols for Internet security, are vulnerable to the severe flaw known as DROWN. With the help of these protocols, anyone with access to the Internet can use email, instant messaging, and web browsing without worrying about a third party being able to read the communication.

DROWN allows attackers to break the encryption and read or steal sensitive communications, including passwords, credit card numbers, trade secrets, or financial data.

Working:

The DROWN attack goes through several stages. First, the attacker must observe and record sessions between the server and the client that use any version of SSL or TLS. For DROWN to work, these sessions must also use RSA cipher suites. Eventually, one of these recorded sessions will be decrypted. At the second stage of the attack, the attacker has captured the usual client/server handshake. They then create multiple connections to the server using the cross-protocol vulnerability. They establish SSLv2 connections to the server, and since the server allows these connections, it is open to exposure. These connections are modified handshake messages that target the RSA ciphertext because unpadded RSA, as used in SSLv2 can be changed. At the second stage of the attack, the attacker has captured the usual client/server handshake. They then create multiple connections to the server using the cross-protocol vulnerability. They establish SSLv2 connections to the server, and since the server allows these connections, it is open to exposure. These connections are modified handshake messages that target the RSA ciphertext because unpadded RSA, as used in SSLv2 can be changed.

VI. VULNERABILITIES IN SSL/TLS-SOFTWARE BASED

SWEET32 (BIRTHDAY) ATTACK

The Sweet32 birthday attack works by exploiting the fact that 64-bit block ciphers, such as Triple DES and Blowfish, have a limited number of possible block values. An attacker must keep track of a sizable volume of encrypted communication using the same key in order to conduct the Sweet32 attack. For instance, this traffic may be HTTPS transmission between a server and a web browser. Depending on the quantity of network traffic, it may take a while for the attacker to capture at least 2^30 blocks of encrypted data. The attacker can start looking for two plaintext blocks that are encrypted to the same ciphertext block after they have gathered enough encrypted data. The attacker expedites this search by applying the birthday paradox. According to the birthday paradox, there is a 50% probability that two persons in a group of 23 will have the same birthdate. There is a 99% probability that 57 people will share a birthdate. The Sweet32 assault follows the same rules. Given a 64-bit block size, there is a significant likelihood of discovering two plaintext blocks that share the same ciphertext block after encrypting around 232 blocks. By XORing known plaintext and the ciphertext, the attacker can then utilize this information to decipher portions of the plaintext.

Vulnerabilities -

1. The 64-bit block ciphers Triple DES and Blowfish are vulnerable to a collision attack because there are only a finite amount of potential block values. The Sweet32 birthday attack takes use of this vulnerability. After an attacker has obtained enough encrypted traffic using the same key, generally at least 230 blocks of data, the attack becomes practical.
2. CBC (Cipher Block Chaining) mode is a frequently used encryption mechanism that is susceptible to several attacks. In the instance of Sweet32, the flaw is that the same ciphertext is generated when successive blocks are encrypted with the same key, which might leak information about the plaintext.
3. Use of long-lived encryption: In some circumstances, 3DES with a 64-bit block size and CBC mode is utilized for a long time, which raises the possibility that an attacker can gather enough ciphertext to carry out a successful attack.

Mitigation:

By employing good key management procedures, such as utilizing multiple keys for various sessions or encrypting just limited quantities of data with the same key, the Sweet32 attack can also be lessened. An organization may become open to assault if certain procedures are not followed.

Limiting the quantity of data that is encrypted with a single key is an additional strategy for preventing the Sweet32 attack. This can be done through key rotation or rekeying, which encrypts data using a new key after a certain quantity of data has been encrypted using the previous key. This strategy makes it more difficult for an attacker to obtain sufficient encrypted data to start the attack.

BREACH ATTACK

The security flaw known as Browser Reconnaissance and Exfiltration through Adaptive Compression of Hypertext (BREACH) targets online applications that employ HTTP compression. It is an attack that may be used to steal sensitive information that is being communicated between a web server and a user's browser, such as passwords, credit card numbers, or other personal information. The victim's browser could have had malicious JavaScript injected into it by the attacker. This may be accomplished via a number of techniques, including a phishing email, an XSS vulnerability, or a malicious advertising. Once the JavaScript has been injected, the attacker may see the communication between the victim's browser and a weak web application. The attacker then issues a string of requests to the web application that have been carefully constructed and each include a small variant of the data that the attacker is attempting to steal. The attacker may be attempting to steal the victim's session ID or other private data, for instance. The attacker estimates the size of the compressed response data after the web application has compressed the response data.

Vulnerabilities-

HTTP Compression: The vulnerability stems from the fact that HTTP compression algorithms recognise repetitive patterns in data and replace them with references to prior occurrences. If sensitive data is transferred via HTTP and compressed using these techniques, an attacker can deduce the original content by analysing the size of the compressed data.

BREACH attacks can also leverage weaknesses in online applications, allowing attackers to inject JavaScript into sites to monitor the number of compressed answers. Attackers can implant malicious scripts that monitor the compressed traffic if the web application is subject to cross-site scripting (XSS) assaults, for example.

Another weakness is that BREACH attacks can be effective when data is transferred via HTTP rather than HTTPS. When data is transferred through HTTPS, it is encrypted, making it considerably more difficult for attackers to deduce the plaintext.

BREACH attacks can also take advantage of human behaviour to steal sensitive data. An attacker, for example, may send a phishing email with a link to a vulnerable web application, then monitor traffic between the user's browser and the web application to steal sensitive information.

Mitigation:

The Browser Reconnaissance and Exfiltration through Adaptive Compression of Hypertext (BREACH) attack can be mitigated in a variety of ways. Among the most successful measures are:

Removing HTTP Compression: The simplest technique to protect against BREACH attacks is to stop HTTP compression entirely. This is accomplished by altering the web server's configuration settings to disable compression. While this increases the quantity of data transferred over the network, it also makes it much more difficult for attackers to deduce the plaintext.

Employing Encryption: Utilizing encryption is one of the most effective strategies to reduce BREACH attacks. Using HTTPS to encrypt data communicated over the network makes it far more difficult for attackers to deduce the plaintext of the data. It is suggested to use strong encryption techniques such as TLS 1.2 or above.

Employing a Web Application Firewall: Web application firewalls can also aid in the prevention of BREACH attacks. These systems are intended to monitor online traffic and stop harmful requests. Several web application firewalls incorporate rules for detecting and preventing BREACH attacks.

SSL STRIPPING ATTACK

SSL stripping is a type of man-in-the-middle (MitM) attack in which an attacker intercepts a victim's encrypted connection to a website or online application and converts it to an unencrypted, plain-text connection. This allows the attacker to intercept and view any sensitive information passed between the victim's browser and the website, such as login credentials, personal information, and financial information.

The attack takes advantage of the fact that many websites and online apps employ Hypertext Transfer Protocol Secure (HTTPS) to encrypt data delivered over the network. The attacker intercepts the victim's encrypted connection and redirects it to an unencrypted HTTP connection. The attacker then redirects the victim's queries to the website, which answers with unencrypted material. The attacker can then intercept and change the content before sending it back to the victim's browser. Because the connection is now unencrypted, the attacker can intercept and read any sensitive information that the victim transmits, including login credentials, personal information, and financial details.

Vulnerabilities –

1. SSL stripping attacks rely on an insecure initial connection between the client and the server, which is vulnerable to interception and manipulation. If the first connection is encrypted via HTTPS, the SSL stripping attack will fail.
2. User trust: SSL stripping attacks frequently rely on the user's willingness to click through warnings and alarms concerning a website's SSL certificate. SSL stripping may be avoided if the user is diligent and takes the time to study and verify SSL certificate data.
3. HTTP Strict Transport Security (HSTS): HTTP Strict Transport Security (HSTS) is a security feature that allows a website to require HTTPS connections and prevent SSL stripping. If HSTS is enabled on a website, the browser will immediately upgrade any HTTP request to HTTPS, rendering the SSL stripping attack useless.
4. SSL stripping attacks may leverage browser vulnerabilities such as obsolete software, extensions, or plugins that may be controlled to intercept or change SSL communications.

Mitigation:

Install HTTPS: Employing HTTPS for all website traffic guarantees that SSL/TLS encryption is utilised to safeguard all client-server interactions. This makes it more difficult for attackers to switch to an unencrypted connection.

Use HTTP Strict Transport Security (HSTS): HSTS encourages browsers to only connect to a website through HTTPS and prevents them from connecting via unencrypted HTTP. This makes it more difficult for an attacker to switch the connection from encrypted to unencrypted.

Use certificate pinning: Certificate pinning allows a website to define which SSL/TLS certificate authorities are trusted and prohibits any other certificate authority from being used. This makes it more difficult for an attacker to execute man-in-the-middle attacks with a forged certificate.

Employ a VPN: A virtual private network (VPN) encrypts all communication between the client and the VPN server, making a man-in-the-middle assault more difficult. This can assist against SSL stripping attacks, which are common while utilizing public Wi-Fi networks.

VII. Concluding Thoughts

In conclusion, SSL/TLS is a widely used protocol for secure communication over the internet. While the protocol offers strong security guarantees, attacks on SSL/TLS have been reported in the past, and it is important to remain vigilant against new vulnerabilities. Common attacks on SSL/TLS include man-in-the-middle attacks, BEAST attacks, Heartbleed, and POODLE. To mitigate these attacks, it is important to keep SSL/TLS implementations up to date, use strong encryption algorithms and key lengths, and implement proper certificate management practices. Additionally, monitoring SSL/TLS traffic for suspicious activity and being prepared to respond to security incidents is crucial.

To mitigate these attacks, it is important to implement proper security measures such as using strong encryption algorithms, key lengths, and certificate management practices. It is also essential to keep SSL/TLS implementations up to date and monitor traffic for suspicious activity. Overall, maintaining vigilance and taking proactive security measures can help protect against attacks on SSL/TLS.

References

[1] https://www.us-cert.gov/ncas/alerts/TA14-098A

[2] https://security.stackexchange.com/questions/19911/crime-how-to-beat-the-beast-successor

[3] https://blog.qualys.com/ssllabs/2011/10/17/mitigating-the-beast-attack-on-tls

[4] https://www.cvedetails.com/

[5] https://www.acunetix.com/blog/articles/tls-vulnerabilities-attacks-final-part/

[6] https://drownattack.com/

[7] https://www.smashingmagazine.com/2015/03/security-concerns-overcome-ssl-tls/

[8] https://www.acunetix.com/blog/articles/tls-vulnerabilities-attacks-final-part/

[9] https://ieeexplore.ieee.org/abstract/document/7877537

[10] https://pdfs.semanticscholar.org/f370/4c1c560ebc2972af454394d54a9469407854.pdf

[11] https://www.cloudflare.com/learning/ssl/why-use-tls-1.3/

[12] https://heimdalsecurity.com/blog/what-is-transport-layer-security/

[13] https://www.sciencedirect.com/science/article/pii/S2210832714000039