*Attacking SSL/TLS Implementations*

Shawn Philip Babu  
Information Systems Security  
Concordia UniversityMontreal, Canada  
shawn0111babu@gmail.com  
  
  
Sai Sandeep Vendra  
Information Systems Security  
Concordia UniversityMontreal, Canada  
sandeepvendra1404@gmail.com line 1: 3rd Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID  
  
line 1: 4th Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 5th Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 6th Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

*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](https://www.websecurity.digicert.com/ssl-certificate?inid=infoctr_buylink_sslhome) from DigiCert you are actually buying the most up to date TLS certificates with the option of [ECC, RSA or DSA encryption](https://www.websecurity.digicert.com/security-topics/how-ssl-works). 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.

# Introduction

JEMY & EKASMEET

(Times new roman 10)

# Limitation of SSL/TLS Implementations

SHAWN & HARSHITHA

(Times new roman 10)

# How SSL/TLS secure data ?

VENKY

(Times new roman 10)

# 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.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |

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.

TABLE

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.

# Vulnerabilities in SSL/TLS-Protocol based

VANI & SAUMYA

(Times new roman 10)

# 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.

# Concluding Thoughts

##### References

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. K. Elissa, “Title of paper if known,” unpublished.
5. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
6. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
7. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 198