Abertay University

SSL vs. TLS

Introduction to Security - CMP 110
BSc Ethical Hacking Year 1

Jack Clark - 1601798@uad.ac.uk

2016 / 17

Abstract

The paper discusses the ways that secure connections are established between a client and server to protect the client's data during transmission from an attacker. The paper analyses the final version of Secure Socket Layer (SSL), v3.0, and how it establishes a secure connection. However, SSL v3.0 has been found to be insecure, and so Transport Layer Security (TLS) is replacing it. The paper will discuss TLS v1.0 and how it creates a secure connection differently from SSL v3.0 and how the most recent version of TLS, v1.3, builds on previous versions and is set to be used as a successor to both SSL v3.0 and previous versions of TLS, and the differences that all three protocols have between one another.

Contents

1 Introduction

1.1 Background

In this day and age, secure connections are vital due to the vast number of people who transfer private data, such as credit card details, online. The way that these connections are established is just as important, as a flaw in establishing the connection can lead to the data being intercepted and worse, decrypted.

Both SSL and TLS are used nowadays to establish a secure connection, however SSL has had multiple successful attacks launched against it with severe data leaks and so it is less secure than once thought. Due to this, TLS is being developed rapidly to eventually take the place of SSL.

TLS is often referred to as SSL v3.1, however this was due to the first version of TLS being so similar to SSL. This is not the case anymore and any references to TLS throughout will be clearly labelled with the version number, for example TLS v1.0 or TLS v1.3.

1.2 Aim

The aim of this project is to research the similarities and differences between SSL and TLS, including how they secure a connection and network traffic and how the newest version of TLS, v1.3, will further secure connections.

1.3 Objectives

To meet the aim above, it is necessary to meet the following objectives:

1. Understand what SSL and TLS are and how they operate
2. Understand the need for TLS
3. Understand the similarities and differences between SSL and TLS

1.4 Methodology

To meet the above aims and objectives, the following steps must be followed:

1. Research SSL and how it operates
2. Research TLS and how it operates and understand why it was created
3. Consider different versions of TLS
4. Find the differences between SSL and TLS

2 Findings

2.1 SSL

SSL (Secure Socket Layer) has been used for many years to secure connections and it has stood up to the strain that it has been under for most of that time. It does this with the use of Public and Private keys and a Certificate.

A server that is using SSL must have a Certificate attached to it which can be self-signed or signed by a Certificate Authority (CA), who will validate the data in it and add it to a trusted list. A Certificate will hold information about the website and company that it is being assigned to, this can include the name of the company and website that it is authorised for, but the most important data that it contains is a pair of Public and Private keys (SSL.com 2016).

When an SSL connection is to be established, it goes through what's known as an SSL Handshake (*see Figure 1*), which is much like the TCP Handshake (which must be done before the SSL Handshake), in that it verifies that the server and client (generally a web browser) are both connected.

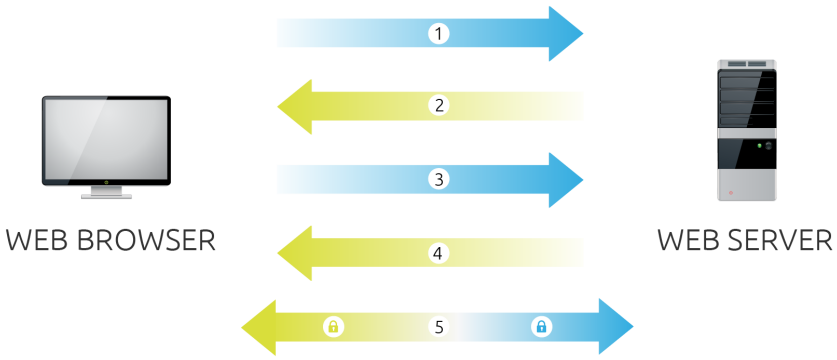

Figure 1: SSL Handshake (Digicert.com 2017)

The steps of an SSL Handshake are as follows (Digicert.com 2017):

1. A TCP Handshake must first occur as the transmission of data is important and using UDP would be impractical as it operates on a connectionless protocol.
2. After the TCP Handshake, the SSL Handshake can commence. It first starts with the client requesting a copy of the server's certificate. This is

important as it contains the server's public key.

3. Once the server receives a request from the client, it will reply with the certificate requested.

   After receiving the certificate, the client will check to ensure that it is valid. This includes checking if:

   (a) the CA that issued the certificate is trusted,

   (b) the certificate is expired or revoked,

   (c) the domain it is registered to is for the domain that it is being used in,

   and should all be valid, a session key (which is used to encrypt all data between the client and server) is generated and encrypted with the public key and then sent to the server.

   However, if any of the checks return false, then the user is warned to let them know that the connection is not private and is using HTTP, or in the case of a certificate being self-signed, the user is warned of this but the connection may still be using HTTPS.

4. The server then decrypts the received session key using its own private key and replies with an acknowledgement that is encrypted with the session key, and now it knows that the connection is secure.

5. After receiving the acknowledgement, the client now knows that it is connected securely to the server and that any data transmitted is encrypted with the session key.

Note that Public and Private Keys are asymmetric, meaning that they are both different, otherwise the connection and protocol as a whole would not be secure. However, the session key that is generated in stage 3 is symmetric, meaning that both the client and the server have the same session key shared between them and it is just encrypted in transit which still makes it secure unless an attacker has control of one of the systems.

2.2 TLS v1.0

Transport Layer Security (TLS) v 1.0 was first developed in 1999 after fears that SSL was less secure and could therefore not be used for sensitive information. But since the POODLE attack on SSL (the attack made use of SSL's encrypt then authenticate sequence, which encrypts traffic before the receiver has been authenticated. During the encryption process, the final byte of a packet is padding, however SSL doesn't specify what must be in the padding therefore attackers can use this to their benefit to decrypt information byte-by-byte, most commonly a cookie) the development of TLS has increased as it aims to take over from SSL and the newest version is set to be released, v1.3, and is currently used in Googles Canary version of Chrome (cloudflare.com 2017).

TLS v1.0 establishes a secure connection in a similar way to SSL and it still makes use of Certificates. It uses a TCP connection to establish an initial connection between client and server and then performs a TLS Handshake (*see Figure 2*).

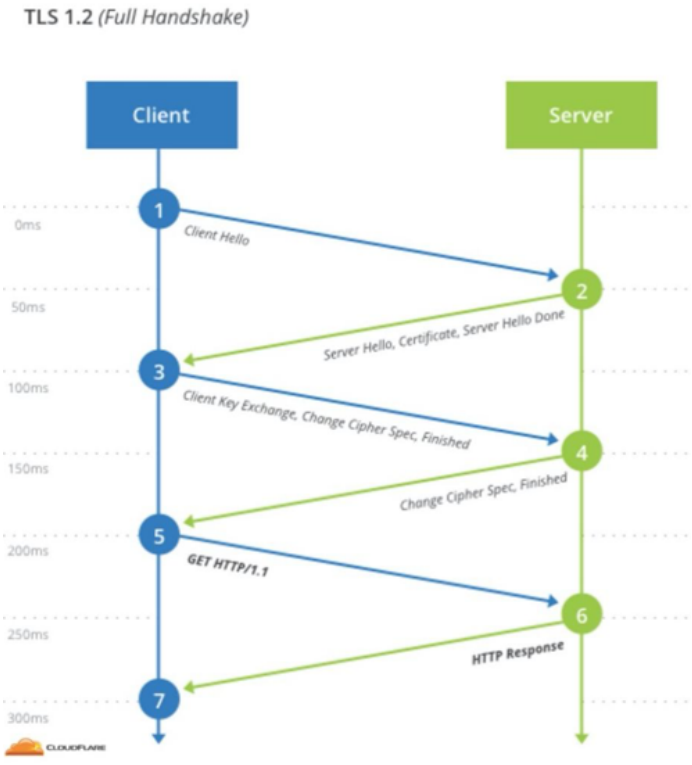

Figure 2: TLS v1.2 Handshake (cloudflare.com 2016a)

The steps of a TLS v1.0 (Figure 2 is the TLS v1.2 handshake, however the handshake has remained the same since v1.0) handshake are (hpbn.co 2017):

1. Just as with the SSL Handshake, a TCP Handshake must be successful before any data relating to TLS Handshake can be sent.

2. Once the TCP Handshake is finished, the client will send information to the server in plaintext, also known as a `CLIENT HELLO`, which will include the version of TLS that the client is running and a list of available ciphersuites that it supports. This being sent in plaintext means that it can be intercepted and read by an attacker, however there is no private or exploitable information transmitted in the `HELLO`.

3. Once the server has received the `HELLO` from the client, it chooses a TLS version and a ciphersuite to use. It then sends this information along with a copy of its Certificate to the client. This information that is sent is called the `SERVER HELLO`.

4. Once the client receives the `SERVER HELLO`, it will check the Certificate to ensure that it is still valid. It checks for the same items as in the SSL Handshake. The client should automatically agree with the TLS version and ciphersuite returned by the server as the server chose from the information given in the `CLIENT HELLO`.

   If the Certificate is valid, the client will initiate the Key Exchange, using either RSA or Diffie-Hellman, which generates a symmetric session key for this session and then sends the session key to the server, encrypted with the Public Key given in the Certificate.

5. Once the server has received the key, it will verify the MAC[1] sent by the client, and if all checks out and is correct then it will send an encrypted `FINISHED` message to the client.

6. After the client has received the encrypted message, it will verify the MAC sent and if it is correct, decrypt the message and if it is as expected (a `FINISHED` message) then the client knows that the connection to the server, and vice versa, is secured with TLS and data transmission can begin.

As can be seen, the SSL and TLS v1.0 handshakes are very different, even though they still use Certificates. However, TLS v1.0 has been found to be vulnerable, as have versions 1.1 and 1.2, so version 1.3 has been created and is set to be officially released this year (and is currently supported by many big companies, for example Cloudflare, Google with their Chrome Canary browser and Firefox with their Nightly browser (cloudflare.com 2017)).

[1]MAC: Message Authentication Code, not to be confused with Media Access Control, is a one-way hash, the keys of which are decided upon by the client and server. The MAC is generated and sent with every message between client and server and is repeatedly checked to ensure the connection integrity.

2.3 TLS v1.3

TLS v1.3 offers vast differences between the previous versions, as well as still being backwards compatible, and has increased the security and speed of the handshake greatly.

2.3.1 Increased Speed (cloudflare.com 2016b)

In previous versions of TLS, the protocol required two round-trips (requests from client to server and back to client) before a page could be requested by the client (*see Figure 2*). However, because of optimization and the removal of the older and less secure encryption methods, it can request a page after the first round-trip (*see Figure 3*).

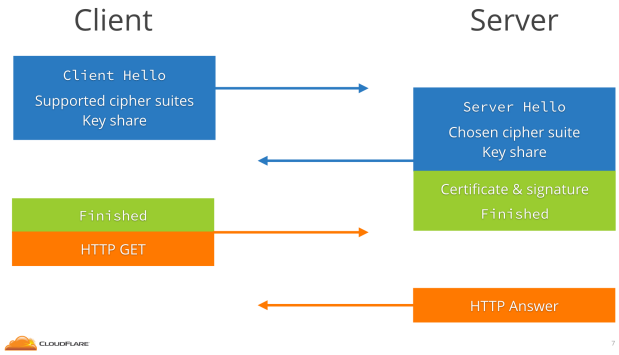

Figure 3: TLS v1.3 Handshake (cloudflare.com 2016b)

As can be seen, in TLS v1.3 the client sends their `HELLO` (including supported ciphersuites) as well as a Key Share that is generated by the client guessing which ciphersuite the server will choose (the equivalent of step 4 of the previous versions of the TLS handshake). This means that when the Server receives this information and decides the ciphersuite it can generate a key to encrypt the traffic with as it already has the clients key, therefore reducing the need for an additional round-trip. The server then replies with its `HELLO`, certificate (encrypted with the generated key), Key Share and a `FINISHED` message to the client. In receiving the data from the server, the client decrypts the server's certificate (by generating the key using both its and the server's key shares) ensures the certificate is valid and checks the `FINISHED` message and if all is as expected it can start its HTTP request knowing that the connection is encrypted, and all done in one round-trip.

However, TLS v1.3 manages to cut down the time to establish a secure connection further for clients that have already connected to the server before by creating a 0-RTT connection (*see Figure 4*).

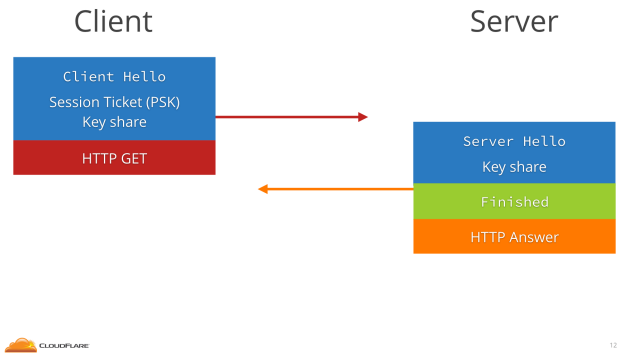

Figure 4: 0-RTT TLS v1.3 Handshake, also known as 0-RTT Resumption (cloudflare.com 2016b)

The first time that a client connects to a server that supports TLS v1.3, both client and server agree on a Pre-Shared Key (PSK) and the server gives the client an encrypted copy of the PSK, also known as a Session Ticket. The Ticket lets the server remember the client.

The next time that the client connects to the server, it sends the Session Ticket along with its `HELLO` and then proceeds to send the HTTP GET request, which is encrypted with the PSK, to the server without any delay. The client also sends a Key Share so that when the server has established the connection, it can use a new key for the request response and the remainder of the connection.

In receiving the data from the client, the server will calculate the PSK from the Session Ticket and then use that PSK to decrypt the traffic from the client. It sends a `FINISHED` message and uses the new key for the HTTP Answer. All of this creates a 0-RTT connection which means that the page can be received in a fraction of the time that even a 1-RTT connections take.

However, 0-RTT doesn't support Forward Secrecy of the Session Ticket key, so if an attacker were to capture the key, they could use it to decrypt the Ticket, then retrieve the PSK from the Ticket and decrypt the initial data that the client sent. Although, an attacker would not be able to decrypt the remainder of the connection as the client and server switch to a new key after the connection has been established.

2.3.2 Increased Security (cloudflare.com 2016b)

TLS v1.3 has removed support for vulnerable encryption methods. These include:

1. RSA Key Transport as it has no support for "Forward Secrecy", the act of ensuring that past keys are not able to be used to decrypt future transmissions by generating a new public key for every connection made, whether it is new or old (cloudlfare.com, 2013). TLS v1.3 now prefers ECDHE (Elliptical Curve Diffie-Hellman Exchange) as it supports Forward Secrecy and a server that uses ECDHE and TLS v1.3 are also what is known as ephemeral, which means that the keys are erased after a connection has ended.

2. SHA-1 hash function after the recent collision where two different PDF files revealed the same hash. Instead SHA-2 is preferred when using TLS v1.3.

There are multiple other methods removed, but the above two were the most commonly used. But with the removal it means that any attacks on previous versions of TLS (which support the encryption methods) will be less likely to work on TLS v1.3. It also means that the TLS protocol is faster to operate and create a handshake.

TLS v1.3 also prevents against downgrade attacks, which an attacker could use to make the TLS version downgrade to a more vulnerable version and then launch an attack that the downgraded protocol is vulnerable to (*see Figure 5*).

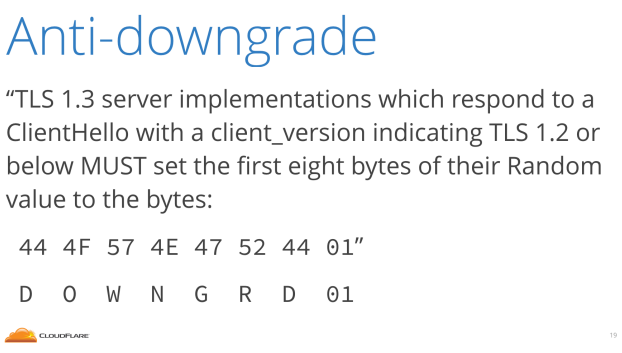

Figure 5: Anti-Downgrade Protection in TLS v1.3 (cloudflare.com 2016b)

TLS v1.3 hides a message in what is known as the Server Random value that equates to `DOWNGRD01`, which a client that is genuinely running a TLS version lower than 1.3 will ignore, but a client that is running TLS v1.3 and has been

requested to downgrade to a lower version will actively search for this message and if found then it knows that it is being tricked into downgrading.

3 Discussion

3.1 Differences Between SSL, TLS v1.0 and TLS v1.3

As mentioned earlier, SSL and TLS are both used to secure a connection between client and server, however they both do this differently and with TLS being more secure and robust. In summary:

SSL v3.0	**TLS v1.2 and Lower**	**TLSv1.3**
Uses encrypt then authenticate when establishing a connection.	All data is encrypted with a session key from the first request to the server.	All data is encrypted with a session key from the first request to the server.
Requires multiple round-trips before page can be requested.	After the first round-trip the page can be requested from the server.	After the first round-trip the page can be requested from the server, with support for 0-RTT connections.
Less secure as still has support for more vulnerable ciphersuites, such as RSA and SHA-1.	More secure than SSL v3.0 however still less secure than TLS v1.3	Removed vulnerable ciphersuites which means attacks on previous versions shouldn't affect v1.3.
Vulnerable to fall-back requests, for example POODLE.	Supports backwards compatibility however is still vulnerable to downgrade attacks.	Supports backwards compatibility but is less likely to be affected by past vulnerabilities.
Still susceptible to MITM attacks.	TLS v1.2 is protected against MITM attacks.	MITM attacks shouldn't work with TLS v1.3 as every request/message sends a MAC which is verified at each end.

As can be seen by the table above, TLS v1.3 is the most secure protocol with the removal of less secure ciphersuites and all data being encrypted between client and server throughout the handshake. This means that TLS v1.3

should be implemented as soon as possible after it is released from its draft state wherever possible to ensure that connections are truly encrypted and private.

4 Conclusion

The issue of secure connections being broken and data being leaked is a very serious and urgent one. However, once TLS v1.3 is out of its draft state and is implemented by servers and the stable release of browsers, it will become very difficult for an attacker to intercept and decrypt data. With the removal of less secure ciphersuites and protection against attacks on SSL and lower versions of TLS, v1.3 appears to be the way forward for HTTPS connections.

References

cloudflare.com (2016a). *Introducing TLS 1.3*. URL: https://blog.cloudflare.com/introducing-tls-1-3/ (visited on 03/30/2017).

— (2016b). *TLS 1.3 Overview and Q and A*. URL: https://blog.cloudflare.com/tls-1-3-overview-and-q-and-a/ (visited on 04/19/2017).

— (2017). *How do I enable TLS 1.3?* URL: https://support.cloudflare.com/hc/en-us/articles/227172348-How-do-I-enable-TLS-1-3 (visited on 04/21/2017).

Digicert.com (2017). *What is SSL and What Are SSL Certificates?* URL: https://www.digicert.com/ssl.htm (visited on 03/29/2017).

hpbn.co (2017). *Transport Layer Security (TLS)*. URL: https://hpbn.co/transport-layer-security-tls/ (visited on 03/30/2017).

SSL.com (2016). *What is SSL?* URL: www.info.ssl.com/article.aspx?id=10241 (visited on 03/29/2017).