Learning about TLS

Without TLS and with TLS

Without TLS means data is transmitted openly across a network, where anyone can intercept a view sensitive information like login credentials, credit card details, or personal messages, while "With TLS" indicates that data is encrypted during transmission, protecting it from eavesdropping by malicious actors when communicating between two systems, effectively essentially, "Without TLS" is like sending a postcard with your address visible, while "with TLS" is like sending a sealed, encrypted letter.

Key scenarios where TLS is crucial:

Online banking:

When logging into our online bank account, TLS ensures that our login credentials are not visible to anyone monitoring the network traffic.

E-Commerce shopping:

When entering credit card information on an online store, TLS protects the data from being intercepted during transmission.

Email Communication:

Sending sensitive information on an online store, TLS-encrypted connection prevents unauthorized access to the message content.

Secure logins into websites:

When logging into any website that requires authentication, TLS protects our username and password from being intercepted.

Example of "wihtout TLS" scenarios (insecure):

* + **Sending a plain text password over a public wi-fi network**: Anyone on the network could easily read our password.
  + **Accessing a website that doesn’t use HTTPS**: Our browsing activity and personal information could be visible to third parties.
  + **Sharing sensitive documents via an unencrypted file transfer protocol**: The documents could be accessed by unauthorized individuals during transfer.

Key points about TLS:

**Encryption**: TLS uses encryption algorithms to scramble data before it is transmitted, making it unreadable to anyone without decryption key.

**Authentication**: TLS verified the identity of the server using digital certificates, ensuring we are communicating with the intended recipient.

**Handshake process**: Before transmission begins, secure connection is established through a handshake process where the client and server negotiate encryption parameters.

Before and After TLS

"Before TLS" signifies a communication scenario where data is transmitted over a network completely unencrypted, meaning anyone can intercept and read the information

While

"After TLS" indicated that data is encrypted using the Transport Layer security protocol, protecting it from eavesdropping and ensuring privacy during transmission;

Essentially, the key difference is level of security, with TLS providing robust encryption for sensitive data once implemented.

Before TLS (UNSECURED)

Data visibility:

Any party on network can easily access and view the transmitted data, including usernames, passwords, credit cards details etc.

Vulnerability to Man-in-the-middle attacks:

Hackers can intercept communication between two parties and modify the data without detection.

No authentication:

There is no way to verify the identity of the server or client, making it susceptible to phishing attempts.

After TLS (Secured):

Encryption:

Data is encrypted using a cryptographic key, making it unreadable to anyone who does not have the decryption key.

Authentication:

TLS uses digital certificates to verify the identity of the server, ensuring users are connecting to the intended party.

Integrity protection:

Data integrity mechanism can be implemented to detect any modification made to the data during transmission.

**Example Scenarios Before TLS**:

Sending login credentials to a website over a plain HTTP connection, where anyone could see the password as it is transmitted.

**Example Scenario After TLS**:

Using HTTPS, where the login credentials are encrypted before being sent to the website, preventing unauthorized access to the data.

What was used before TLS?

TLS evolved from a previous encryption protocol called Secure Socket Layer (SSL), which was developed by Netscape.

Why TLS must be used during and after user authentication occurs?

TLS authentication facilitates encryption and verifies the legitimacy of the server (server authentication) and, optionally, the client (client authentication), thus ensuring data integrity and confidentiality.

TLS handshake occur after a TCP connection has been opened via TCP handshake.

Does TLS runs over TC or UDP?

TLS runs over a reliable transport (TCP), which means that we must first complete the TCP three-way handshake, which takes one full roundtrip.

What is the reason for TLS handshake timeout?

Protocol mismatch: A TLS handshake failures occurs when the client and the server don’t mutually support TLS version, e.g., the browser supports TLS 1.0 or TLS 1.1 while server supports TLS 1.3. in this case, the user should upgrade their browser to work with the latest TLS version.

Does API use TLS?

SSL/TLS (Secure socket layer/ Transport layer security) secures APIs by encrypting data transmitted between client and servers It ensures data confidentiality, integrity and authenticity.

How to implement TLS over TCP?

Lets walk through the steps at a high level:

* + TCP connection. Our client (browser or application) will initiate a TCP connection with a server.
  + SSL/ TLS handshake. The SSL/TLS handshake takes place once a TCP connection is established.
  + Client hello.
  + Certificate verification.
  + Client Key exchange.
  + Finished/ Application data.

What is port 53?

Port 53 is used for both TCP and UDP communication. For example, when a user types a URL into their web browser, the browser first sends a DNS to DNS server to translate the domain name into an IP address. The response from the server is sent back to the browser on the same port. - port 53.

What is port 389?

LDAP TCP:

LDAP TCP and UDP port 389 is used for Directory, Replication, User and computer authentication, Group policy, trusts.

Is DHCP UDP or TCP?

The DHCP employs a connectionless service model, using user datagram protocol (UDP). It is implemented with 2 UDP port numbers for its operations which are the same as for the bootstrap protocol (BOOTP). The server listens on UDP port number67, and the client listens on UDP port number 68.

What port is DHCP?

DHCP uses UDP port 67. on the server side and UDP port 68 on the client side.

What port is HTTPS?

By default, HTTPS connections use TCP port 443. HTTP, the unsecure protocol uses port 80.

What is a SSH port?

Port 22 is associated with the SSH(Secure Shell) protocol, which is used to securely connect to a remote device and issue commands just like we would on our device. This default port does have its vulnerabilities though: port 22 a popular target for brute force attacks and unauthorized access attempts.

What port is Telnet?

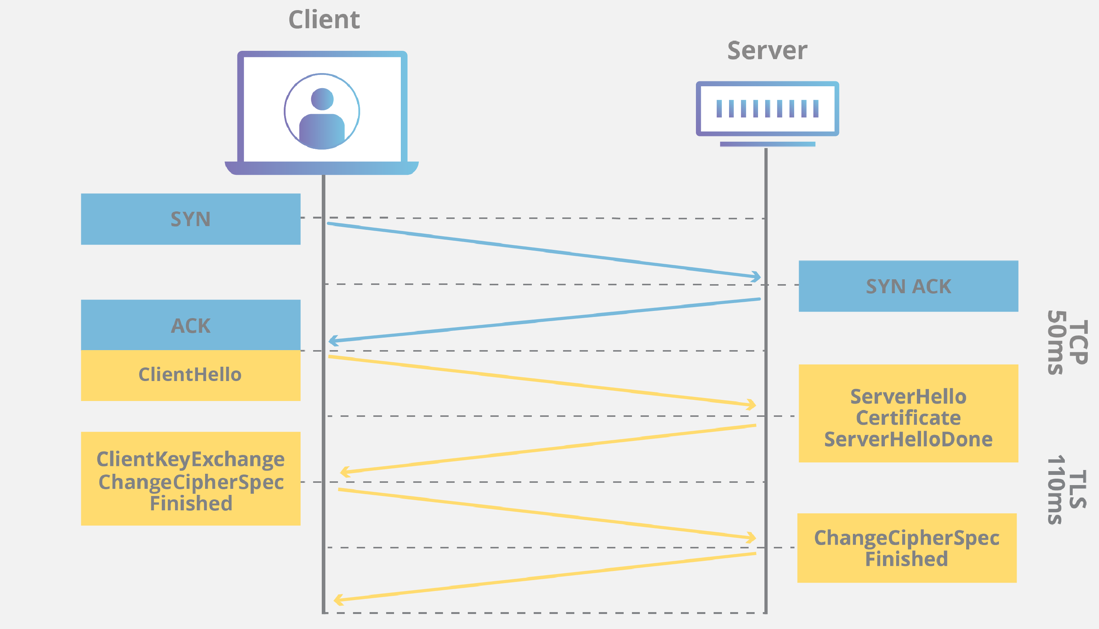
* + The telnet protocol is a client-server protocol, base don a reliable connection-oriented transport. This protocol is used to establish a connection Transmission control protocol (TCP) port number 23 or 2323, where a telnet server application is listening.

Learning Objectives

After reading this article you will be able to:

* + Learn what a TLS handshake is
  + Understand what a TLS handshake accomplishes
  + Explain the steps in a TLS handshake
  + Explore different types of TLS handshakes

What is a TLS handshake?



[TLS](https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/) is an encryption and authentication protocol designed to secure Internet communications. A TLS handshake is the process that kicks off a communication session that uses TLS. During a TLS handshake, the two communicating sides exchange messages to acknowledge each other, verify each other, establish the cryptographic algorithms they will use, and agree on session keys. TLS handshakes are a foundational part of [how HTTPS works](https://www.cloudflare.com/learning/ssl/what-is-https/).

TLS vs. SSL handshakes

[SSL, or Secure Sockets Layer](https://www.cloudflare.com/learning/ssl/what-is-ssl/), was the original security protocol developed for [HTTP](https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/). SSL was replaced by TLS, or Transport Layer Security, some time ago. SSL handshakes are now called TLS handshakes, although the "SSL" name is still in wide use.

Whitepaper

Maximize the power of TLS

[Get the report](https://www.cloudflare.com/lp/maximize-tls/)

Guide

The Zero Trust guide to securing aplication access

[Read the guide](https://www.cloudflare.com/lp/guide-to-zero-trust-access/)

When does a TLS handshake occur?

A TLS handshake takes place whenever a user navigates to a website over HTTPS and the browser first begins to query the website's [origin server](https://www.cloudflare.com/learning/cdn/glossary/origin-server/). A TLS handshake also happens whenever any other communications use HTTPS, including [API calls](https://www.cloudflare.com/learning/security/api/what-is-api-call/) and [DNS over HTTPS](https://www.cloudflare.com/learning/dns/dns-over-tls/) queries.

TLS handshakes occur after a [TCP](https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/) connection has been opened via a TCP handshake.

Secure SSL

Free SSL included with any Cloudflare plan

[Start for free](https://www.cloudflare.com/plans/)

What happens during a TLS handshake?

During the course of a TLS handshake, the client and server together will do the following:

* + Specify which version of TLS (TLS 1.0, 1.2, 1.3, etc.) they will use
  + Decide on which cipher suites (see below) they will use
  + Authenticate the identity of the server via the server’s public key and the SSL certificate authority’s digital signature
  + Generate session keys in order to use symmetric encryption after the handshake is complete

What are the steps of a TLS handshake?

TLS handshakes are a series of datagrams, or messages, exchanged by a client and a server. A TLS handshake involves multiple steps, as the client and server exchange the information necessary for completing the handshake and making further conversation possible.

The exact steps within a TLS handshake will vary depending upon the kind of key exchange algorithm used and the cipher suites supported by both sides. The RSA key exchange algorithm, while now considered not secure, was used in versions of TLS before 1.3. It goes roughly as follows:

* + **The 'client hello' message:** The client initiates the handshake by sending a "hello" message to the server. The message will include which TLS version the client supports, the cipher suites supported, and a string of random bytes known as the "client random."
  + **The 'server hello' message:** In reply to the client hello message, the server sends a message containing the server's [**SSL certificate**](https://www.cloudflare.com/learning/ssl/what-is-an-ssl-certificate/), the server's chosen cipher suite, and the "server random," another random string of bytes that's generated by the server.
  + **Authentication:** The client verifies the server's SSL certificate with the certificate authority that issued it. This confirms that the server is who it says it is, and that the client is interacting with the actual owner of the domain.
  + **The premaster secret:** The client sends one more random string of bytes, the "premaster secret." The premaster secret is encrypted with the public key and can only be decrypted with the private key by the server. (The client gets the [**public key**](https://www.cloudflare.com/learning/ssl/how-does-public-key-encryption-work/) from the server's SSL certificate.)
  + **Private key used:** The server decrypts the premaster secret.
  + **Session keys created:** Both client and server generate session keys from the client random, the server random, and the premaster secret. They should arrive at the same results.
  + **Client is ready:** The client sends a "finished" message that is encrypted with a session key.
  + **Server is ready:** The server sends a "finished" message encrypted with a session key.
  + **Secure symmetric encryption achieved:** The handshake is completed, and communication continues using the session keys.

All TLS handshakes make use of asymmetric cryptography (the public and private key), but not all will use the private key in the process of generating session keys. For instance, an ephemeral Diffie-Hellman handshake proceeds as follows:

* + **Client hello:** The client sends a client hello message with the protocol version, the client random, and a list of cipher suites.
  + **Server hello:** The server replies with its SSL certificate, its selected cipher suite, and the server random. In contrast to the RSA handshake described above, in this message the server also includes the following (step 3):
  + **Server's digital signature:** The server computes a digital signature of all the messages up to this point.
  + **Digital signature confirmed:** The client verifies the server's digital signature, confirming that the server is who it says it is.
  + **Client DH parameter:** The client sends its DH parameter to the server.
  + **Client and server calculate the premaster secret:** Instead of the client generating the premaster secret and sending it to the server, as in an RSA handshake, the client and server use the DH parameters they exchanged to calculate a matching premaster secret separately.
  + **Session keys created:** Now, the client and server calculate session keys from the premaster secret, client random, and server random, just like in an RSA handshake.
  + **Client is ready:** Same as an RSA handshake.
  + **Server is ready**
  + **Secure symmetric encryption achieved**

\*DH parameter: DH stands for Diffie-Hellman. The Diffie-Hellman algorithm uses exponential calculations to arrive at the same premaster secret. The server and client each provide a parameter for the calculation, and when combined they result in a different calculation on each side, with results that are equal.

To read more about the contrast between ephemeral Diffie-Hellman handshakes and other kinds of handshakes, and how they achieve forward secrecy, see [What is Keyless SSL?](https://www.cloudflare.com/learning/ssl/keyless-ssl/)

What is different about a handshake in TLS 1.3?

TLS 1.3 does not support RSA, nor other cipher suites and parameters that are vulnerable to attack. It also shortens the TLS handshake, making a TLS 1.3 handshake both faster and more secure.

The basic steps of a TLS 1.3 handshake are:

* + **Client hello:** The client sends a client hello message with the protocol version, the client random, and a list of cipher suites. Because support for insecure cipher suites has been removed from TLS 1.3, the number of possible cipher suites is vastly reduced. The client hello also includes the parameters that will be used for calculating the premaster secret. Essentially, the client is assuming that it knows the server’s preferred key exchange method (which, due to the simplified list of cipher suites, it probably does). This cuts down the overall length of the handshake — one of the important differences between TLS 1.3 handshakes and TLS 1.0, 1.1, and 1.2 handshakes.
  + **Server generates master secret:** At this point, the server has received the client random and the client's parameters and cipher suites. It already has the server random, since it can generate that on its own. Therefore, the server can create the master secret.
  + **Server hello and "Finished":** The server hello includes the server’s certificate, digital signature, server random, and chosen cipher suite. Because it already has the master secret, it also sends a "Finished" message.
  + **Final steps and client "Finished":** Client verifies signature and certificate, generates master secret, and sends "Finished" message.
  + **Secure symmetric encryption achieved**

0-RTT mode for session resumption

TLS 1.3 also supports an even faster version of the TLS handshake that does not require any [round trips](https://www.cloudflare.com/learning/cdn/glossary/round-trip-time-rtt/), or back-and-forth communication between client and server, at all. If the client and the server have connected to each other before (as in, if the user has visited the website before), they can each derive another shared secret from the first session, called the "resumption main secret." The server also sends the client something called a session ticket during this first session. The client can use this shared secret to send encrypted data to the server on its first message of the next session, along with that session ticket. And TLS resumes between client and server.

What is a cipher suite?

A cipher suite is a set of algorithms for use in establishing a secure communications connection. There are a number of cipher suites in wide use, and an essential part of the TLS handshake is agreeing upon which cipher suite will be used for that handshake.

To learn more about TLS/SSL, see [How does SSL work?](https://www.cloudflare.com/learning/ssl/how-does-ssl-work/).

<https://www.cloudflare.com/en-gb/learning/ssl/what-is-https/>