HTTPS

What is Web Security?

The web is not safe for every user, Daily we hear about websites becoming unavailable because of denial of service attacks, or displaying changed information on their pages.  
These articles are essential for understanding web security basics.

Those articles summarize the most common attacks and explain the countermeasures every web application should implement.

Essentially, these articles provide you with the knowledge needed to create better and safer web applications.

Different Sorts of Web security:

[HTTPS](https://dev.to/ahmedatefae/web-security-knowledge-you-must-understand-it-part-i-https-tls-ssl-cors-csp-298l#https)

[TLS](https://dev.to/ahmedatefae/web-security-knowledge-you-must-understand-it-part-i-https-tls-ssl-cors-csp-298l#tls)

[SSL](https://dev.to/ahmedatefae/web-security-knowledge-you-must-understand-it-part-i-https-tls-ssl-cors-csp-298l#ssl)

[CORS](https://dev.to/ahmedatefae/web-security-knowledge-you-must-understand-it-part-i-https-tls-ssl-cors-csp-298l#cors)

[CSP](https://dev.to/ahmedatefae/web-security-knowledge-you-must-understand-it-part-i-https-tls-ssl-cors-csp-298l#csp)

OWASP (Security Risks) >> I will explain it in part II of this series

Hashing Algorithms >> I will explain it in part III of this series

MD5

SHA

scrypt

bcrypt

HTTPS

It is the secure version of HTTP, the full name of this protocol is Hypertext transfer protocol secure, which is the primary used to send data between a web browser and a website.

No one else can access the data, Because It uses TLS protocol to encrypt communications, I will explain it at the coming topic.

How HTTPS work?

It uses encryption communication protocol, named Transport Layer Security (TLS), was known as Secure Sockets Layer (SSL).

this encryption uses two keys, one names public key and the other one names private key.

**public key**: this shared between the browser and website.

**private key**: this key used to decrypt information encrypted by the public key and it not shared out of the server.

TLS

it is the most protocol in use and designed to facilitate privacy, data security for communications over the internet, the use case of TLS is encrypting the communication between application and servers, emails, messaging voice by use (VoIP).

How TLS work?

Any application or website to use the TLS it must have to install the TLS certification (also known as an “SSL certification”) on the base server, by issued to the person or the organization that own the Domain to install it on the base server.

It contains very important information about the owner, private and public keys to use in decrypt and encrypt the communication.

this process named TLS handshake 🤝 its steps :

determined the version of the TLS will use during the session.

Authenticate the identity of the server by using the TLS certificate.

Generate the Session key for use during the session after the handshake process ended.

This topic needed a farther explanation for this I will make an article about it, and I will add a link to it here

SSL

Secure Sockets Layer (SSL), an encryption-based internet security protocol, it was founded for the ensuring of the integrity and privacy of the connections of the internet by the Netscape company at 1995, nowadays it names [TLS](https://dev.to/ahmedatefae/web-security-knowledge-you-must-understand-it-part-i-https-tls-ssl-cors-csp-298l#tls).

How SSL works?

it like the new TLS based on the concept of the handshake [TLS](https://dev.to/ahmedatefae/web-security-knowledge-you-must-understand-it-part-i-https-tls-ssl-cors-csp-298l#tls).

What is the difference between TLS and SSL?

SSL is the older version of the TLS, the name changed after the Internet Engineering Task Force (IETF) be the owner of the SSL development after Netscape, some developer nowadays uses the SSL and TLS to referring for the same thing.

CORS

Cross-Origin Resource Sharing (CORS) is a mechanism that uses HTTP headers to specify which outer origin have access to the local assets and how can access it that is mean we can make a white list for the allowed Cross-Origins that has access to our assets.

the server must have a way to handle the outer requests and this what we will discuss now.

How CORS works?

when the site makes a get request to get resource from the out server, the browser adds a header that contains the origin like the example Origin: http://www.example.com.

The server receives the preflight request and searches in its white-list for Access-Control-Allow-Origins about the giving origin and sends to the browser option call, then the browser will determine if the actual request is safe to send or not, example Access-Control-Allow-Origin: http://www.example.com or this header Access-Control-Allow-Origin: \* will allow any request to take the resource.

if the server specifies the Methods it will compare the request method with its example Access-Control-Allow-Methods: PUT, DELETE.

CSP

Content Security Policy is more security layer that helps in detect and mitigate different sort of militias attacks like (Cross-Site Scripting (XSS), data injection attacks, ClickJacking, ETC...).

**Cross-Site Scripting (XSS)**: it a vulnerability that allows the hacker to inject a militias code in the base website and it is for making the client execute it to take sensitive data like cookies, session’s info and site-specific information, That happens because web app does not use enough validation or encoding, The user’s browser cannot detect the malicious script is untrustworthy.

**data injection attacks:** is a malicious code injected in the network which fetched all the information from the database to the attacker and the number one type of it is the SQL injection.

**Click Jacking:** or “UI redress attack” is when an attacker tricks a user into clicking on a button or link on another page that uses multiple transparent or opaque layers when he intended to click on the top-level.

How CSP works?

it use Directives concept that’s every Directive have to specify where resources can load from, preventing browsers from loading data from any other locations.

**Most used Directives are:**

**default-src:** the default policy for loading (JavaScript, images, CSS, AJAX requests, ETC...) example default-src ‘self’ cdn.example.com;

**img-src:** defines sources for images example img-src ‘self’ img.example.com;

**style-src:** defines sources for CSS files example style-src ‘self’ css.example.com;

**script-src:** defines sources for JavaScript files example script-src ‘self’ js.example.com;

**connect-src:** defines valid targets for to XMLHttpRequest (AJAX), WebSockets or EventSource, If it makes any connections to a host, that’s not allowed here, the browser will respond with a 400 error example connect-src ‘self’;

**Multi-label directives defines:** default-src ‘none’; script-src ‘self’; connect-src ‘self’; img-src ‘self’; style-src ‘self’;