



Why should you care about HTTPS and 3 Tips?

Contributed by

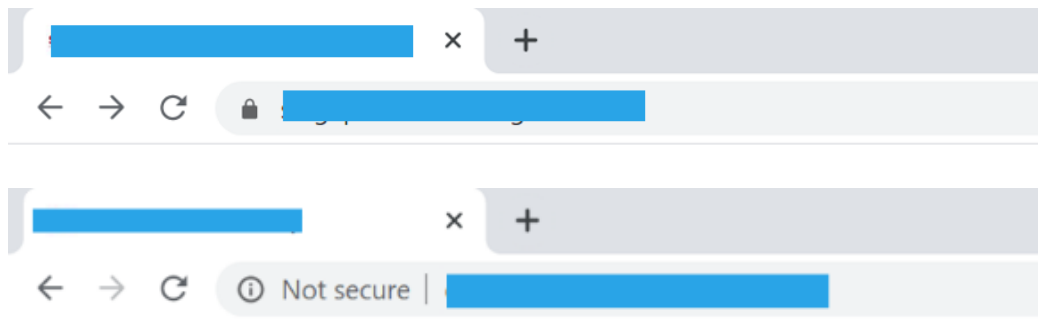
James Tan, MAISP

Co-opted Committee Member, Association of Information Security Professionals (AiSP)

The Internet is a microcosm; it is not all about technology. There are 'pillars' that enable everyone to freely interact, exchange information and transact with one another; privacy, security and trust allow us to do this.

When we click on a link or type "google.com" or any word into our web browser, we are automatically taken to the website. The URL may be shown with an icon (e.g. ) next to it; sometimes not at all, or at times with an alert/warning (e.g. )

Examples,



When you see <https://www.google.com>, you will think it is secure enough. The browser and the website (hosted on the webserver) are using HTTPS¹ to check if they can trust one another. Importantly, the browser is giving you, the user, the visual feedback and mechanism to check (depending on your cybersecurity hygiene habits), whether or not you should trust the website that you are going to transact with.

Privacy is and has been a major concern for both individuals and businesses. Users may not want to be monitored and tracked of their online activities, outside of their control, and businesses are required by legislation to ensure this privacy is taken care. HTTPS ensures privacy via security.

Security is provided by HTTPS using the TLS² protocol to verify and encrypt the information sent between the browser and the website. No one else, except the browser (anyone behind the browser can still see the same conversation) and the website, should view the content in this communication unless they perform a man-in-the-middle or a cryptographic attack to defeat the weak HTTPS implementation³. Besides preventing the potential monitoring of content which could lead to a loss of privacy and confidentiality, the encryption between the two ensures that any 3rd party cannot inject

¹ Hypertext Transfer Protocol Secure

² Transport Layer Security. It succeeds SSL but, we will use this interchangeably in this article.

³ We will not go into other many possible attacks against the OS or the application weaknesses, you could refer to CIS 20 Controls and or OWASP for guidance.

'extra' content, e.g. malware and adware. The HTTPS also provides non-repudiation of the user's requests and the website's content, which adds trust between the two parties.

The following steps illustrate what happens⁴ under the hood when a user visits a *somewhat trustworthy website* hosted on a server in a simplified manner:

1. Browser [HTTPS capabilities, token] → Server.
2. Browser ← [SSL certificate, cipher suite, token] Server.
3. The browser checks the SSL certificate⁵ and extracts the server public key and use it to encrypt a secret⁶.
4. Browser → [secret] Server
5. The server uses its private key to extract the secret. Browser and server use the secret to generate a session key.
6. Browser [session key] → encrypted communication ← [session key] Server

From the business angle, when you use HTTPS on every website, you are demonstrating to your users and markets that you care about their security and privacy when they transact with you, and this is how you can 'earn' their trust.

How can I get started?

Checklist⁷

1. Are you going to implement HTTPS on the website server or put your website behind a CDN, and let the CDN takes care of the HTTPS?
2. How many domains do you need to specify in the certificate so that the Browser can verify against it?
3. Can you use a free CA⁸ signed certificate or a paid CA signed certificate SSL certificate?
 - a. Free: Let's Encrypt.
 - i. Use the script to generate the certificates for you. Once this is set up, the subsequent renewal is automated.
 - b. Paid: Shop around for a reputable reseller.
 - i. Create a private key and generate the certificate signing request (CSR).
 - ii. Submit the CSR to the reseller to let the CA generate and sign the certificates for you.
 - iii. Install the certificates in your web server.
 - iv. Perform the same steps before the certificate expires.

⁴ Interaction between browser and website follows the HTTPS protocol specified in the RFC, e.g. "The Transport Layer Security (TLS) Protocol, Version 1.2".

⁵ Trust, but verify.

⁶ On Step 3, the browser is checking the authenticity of the server's certificate. If the server is using a "self-signed" or an "expired" or "revoked" certificate, the browser will show the user an alert or warning and may also block the individual from interacting with the website.

⁷ Most SSL certificate resellers provide very detailed instructions for the various OS and web servers. You can check with the reseller/vendor of your choice.

⁸ certification authority

Here are our 3 Tips⁹:

Tip 1. HTTPS everything (“leave no stone unturned”)

- a. Open only the HTTPS port on your firewall/server.
- b. HTTPS all external references, e.g. <https://cdnjs.com/libraries/jquery/>
- c. Harden your application/website¹⁰:
 - i. “Secure Cookies”.
 - ii. “HTTP Strict Transport Security”.
 - iii. “OCSP Stapling”.

Tip 2. Do not use a self-signed certificate. Get a CA signed certificate.

- a. Self-signed certificates ‘brainwash’ your users to ignore the risks and bypass the browser restriction. This can be problematic when you train them otherwise during security awareness¹¹.
- b. After generating the private key (to create the CSR and to enable the HTTPS on your Server), protect it¹².
- c. Set up certificate transparency (CT) alerts and monitoring to inform you when a certificate is issued to your domain name. This allows you to verify if you are the one requesting for it or somebody else is.

Tip 3. Check your HTTPS implementation

- a. For publicly accessible websites (and you have nothing to hide), you can use SSL Labs¹³ to run a quick scan. Try to achieve at least an “A”.

⁹ You will need to approach your system administrator, developers and or outsourced vendors.

¹⁰ Refer to OWASP.

¹¹ You can set up your own internal CA and issue your CA signed certificate and then setup every users’ browsers/OS to trust the CA signed certificate; your users will not get a browser alert, and they will not attempt to circumvent it.

¹² You can set up a password to use it and or implement it in an HSM/” keyless” solution (more so when you offload HTTPS termination/function to a load balancer or CDN).

¹³ <https://www.ssllabs.com/ssltest/index.html>

FAQ

Q1. Do I have to go through the whole process of creating the private key, generating the CSR, obtaining the server certificate, and installing it on every website?

A1. It depends. An organization with one domain or website will only need to do this once during the initial implementation and then using the same secured private key (unless it is lost, then you have to recreate another one) regularly before the expiry date.

A2. If every website is on the same FQDN¹⁴, e.g. <https://apps.company.com/trading> and <https://apps.company.com/customer-portal>, you just need to manage one certificate for the domain name company.com.

A3. If you have multiple domains, websites, or applications, you can generate a single SAN/UCC certificate and implement it across the different sites.

Q2. What happens if my certificate expires?

A1. Your customers or users will see a big warning on their browsers and may not be able to transact on your websites. Renew early or build automation into the renewal process.

Q3. Do I have to spend a lot on getting the 'best' TLS certificate from a 'branded' reseller?

A1. No, but do not go for the cheapest just based on the price. Check if the reseller has HTTPS and is reputable. You do not want to lose your personal and credit card information to them.

A2. You can use Let's Encrypt to automatically generate, free, certificates. Free here does not mean 'free meal'; you will need to know how to use and manage it.

Resources:

1. <https://tools.ietf.org/html/rfc5246>
2. <https://ssl-config.mozilla.org/>
3. <https://letsencrypt.org/>
4. <https://howhttps.works/>
5. <https://aboutssl.org/how-to-renew-ssl-certificate/>
6. <http://www.certificate-transparency.org/>
7. <https://www.cisecurity.org/controls/cis-controls-list/>
8. https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html
9. <https://www.ssllabs.com/ssltest/index.html>

This article is specially written for the association partners of AISP. Please contact secretariat@aisp.sg if you have any query.

¹⁴ fully qualified domain name.