HTTPS

Hypertext Transfer Protocol Secure (**HTTPS**) is an extension of the Hypertext Transfer Protocol (HTTP) for secure communication over a computer network, and is widely used on the Internet.[1][2] In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS), or formerly, its predecessor, Secure Sockets Layer (SSL). The protocol is therefore also often referred to as **HTTP over TLS**,[3] or **HTTP over SSL**.

The principal motivation for HTTPS is authentication of the accessed website and protection of the privacy and integrity of the exchanged data while in transit. It protects against man-in-the-middle attacks. The bidirectional encryption of communications between a client and server protects against eavesdropping and tampering of the communication.[4] In practice, this provides a reasonable assurance that one is communicating without interference by attackers with the website that one intended to communicate with, as opposed to an impostor.

Historically, HTTPS connections were primarily used for payment transactions on the World Wide Web, e-mail and for sensitive transactions in corporate information systems. Since 2018[5], HTTPS is used more often by web users than the original non-secure HTTP, primarily to protect page authenticity on all types of websites; secure accounts; and keep user communications, identity, and web browsing private.

Contents

Overview

The Uniform Resource Identifier (URI) scheme *HTTPS* has identical usage syntax to the HTTP scheme. However, HTTPS signals the browser to use an added encryption layer of SSL/TLS to protect the traffic. SSL/TLS is especially suited for HTTP, since it can provide some protection even if only one side of the communication is authenticated. This is the case with HTTP transactions over the Internet, where typically only the server is authenticated (by the client examining the server's certificate).

HTTPS creates a secure channel over an insecure network. This ensures reasonable protection from eavesdroppers and man-in-the-middle attacks, provided that adequate cipher suites are used and that the server certificate is verified and trusted.

Because HTTPS piggybacks HTTP entirely on top of TLS, the entirety of the underlying HTTP protocol can be encrypted. This includes the request URL (which particular web page was requested), query parameters, headers, and cookies (which often contain identity information about the user). However, because host (website) addresses and port numbers are necessarily part of the underlying TCP/IP protocols, HTTPS cannot protect their disclosure. In practice this means that even on a correctly configured web server, eavesdroppers can infer the IP address and port number of the web server (sometimes even the domain name e.g. www.example.org, but not the rest of the URL) that one is communicating with, as well as the amount (data transferred) and duration (length of session) of the communication, though not the content of the communication.[4]

URL beginning with the HTTPS scheme and the WWW domain name label

Web browsers know how to trust HTTPS websites based on certificate authorities that come pre-installed in their software. Certificate authorities (such as Let's Encrypt, Digicert, Comodo, GoDaddy and GlobalSign) are in this way being trusted by web browser creators to provide valid certificates. Therefore, a user should trust an HTTPS connection to a website if and only if all of the following are true:

- The user trusts that the browser software correctly implements HTTPS with correctly pre-installed certificate authorities.
- The user trusts the certificate authority to vouch only for legitimate websites.
- The website provides a valid certificate, which means it was signed by a trusted authority
- The certificate correctly identifies the website (e.g., when the browser visits "https://example.com", the received certificate is properly for "example.com" and not some other entity).
- The user trusts that the protocol's encryption layer (SSL/TLS) is sufficiently secure against eavesdroppers.

HTTPS is especially important over insecure networks (such as public Wi-Fi access points), as anyone on the same local network can packet-sniff and discover sensitive information not protected by HTTPS. Additionally, many free to use and paid WLAN networks engage in packet injection in order to serve their own ads on webpages. However, this can be exploited maliciously in many ways, such as injecting malware onto webpages and stealing users' private information.[6]

HTTPS is also very important for connections over the Tor anonymity network, as malicious Tor nodes can damage or alter the contents passing through them in an insecure fashion and inject malware into the connection. This is one reason why the Electronic Frontier Foundation and the Tor project started the development of HTTPS Everywhere,[4] which is included in the Tor Browser Bundle.[7]

As more information is revealed about global mass surveillance and criminals stealing personal information, the use of HTTPS security on all websites is becoming increasingly important regardless of the type of Internet connection being used.[8][9] While metadata about individual pages that a user visits is not sensitive, when combined, they can reveal a lot about the user and compromise the user's privacy.[10][11][12]

Deploying HTTPS also allows the use of HTTP/2 (or its predecessor, the now-deprecated protocol SPDY), that are new generations of HTTP, designed to reduce page load times, size and latency.

It is recommended to use HTTP Strict Transport Security (HSTS) with HTTPS to protect users from man-in-the-middle attacks, especially SSL stripping.[12][13]

HTTPS should not be confused with the little-used Secure HTTP (S-HTTP) specified in RFC 2660.

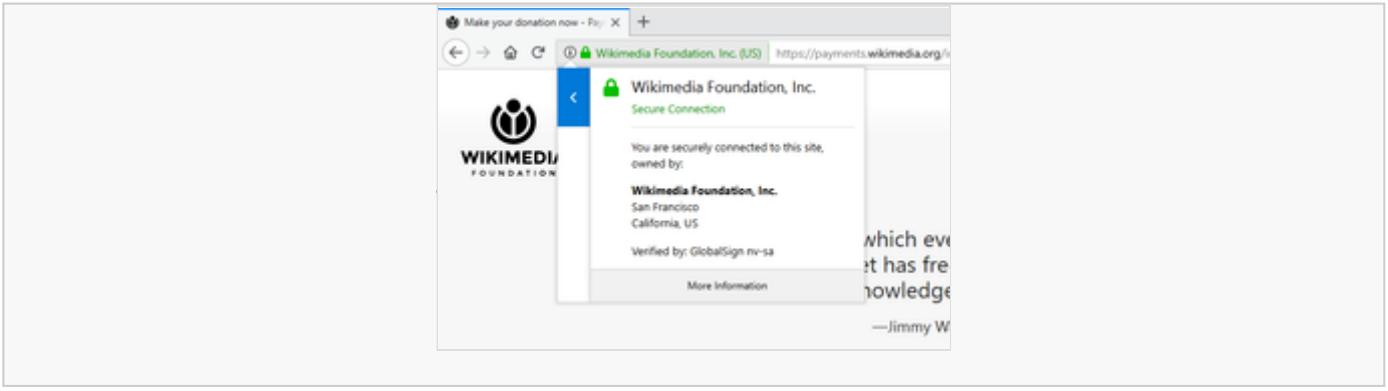
Usage in websites

As of April 2018, 33.2% of Alexa top 1,000,000 websites use HTTPS as default,[14] 57.1% of the Internet's 137,971 most popular websites have a secure implementation of HTTPS,[15] and 70% of page loads (measured by Firefox Telemetry) use HTTPS.[16]

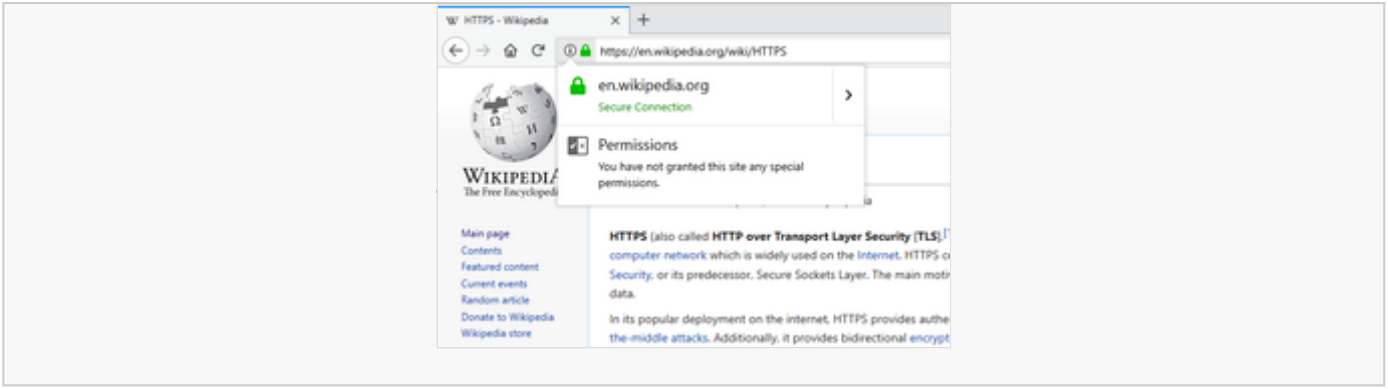
Browser integration

Most browsers display a warning if they receive an invalid certificate. Older browsers, when connecting to a site with an invalid certificate, would present the user with a dialog box asking whether they wanted to continue. Newer browsers display a warning across the entire window. Newer browsers also prominently display the site's security information in the address bar. Extended validation certificatesturn the address bar green in newer browsers. Most browsers also display a warning to the user when visiting a site that contains a mixture of encrypted and unencrypted content.

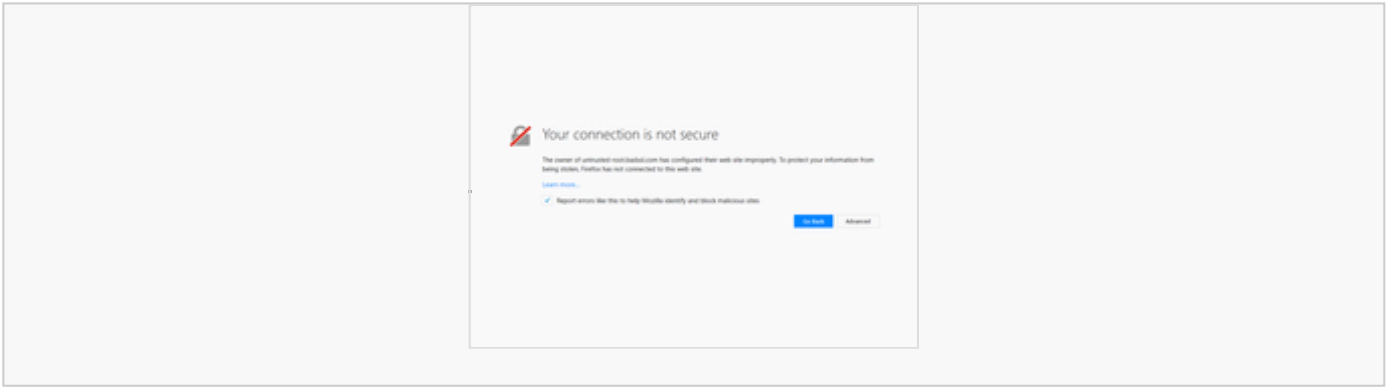
Comparison between different kinds of SSL/TLS certificates
(Using Firefox as an example)



Many web browsers, including Firefox (shown here), use the address bar to tell the user that their connection is secure, often by coloring the background.



When accessing a site only with a common certificate, the address bar of Firefox turns green. For some other browsers, a "lock" sign may appear.



Most web browsers alert the user when visiting sites that have invalid security certificates.

The Electronic Frontier Foundation, opining that "In an ideal world, every web request could be defaulted to HTTPS", has provided an add-on called HTTPS Everywhere for Mozilla Firefox that enables HTTPS by default for hundreds of frequently used websites. A beta version of this plugin is also available for Google Chrome and Chromium.[17][18]

Security

The security of HTTPS is that of the underlying TLS, which typically uses long-term public and private keys to generate a short-term session key, which is then used to encrypt the data flow between client and server. X.509 certificates are used to authenticate the server (and sometimes the client as well). As a consequence, certificate authorities and public key certificates are necessary to verify the relation between the certificate and its owner, as well as to generate, sign, and administer the validity of certificates. While this can be more beneficial than verifying the identities via a web of trust, the 2013 mass surveillance disclosures drew attention to certificate authorities as a potential weak point allowing man-in-the-middle attacks.[19][20] An important property in this context is forward secrecy, which ensures that encrypted communications recorded in the past cannot be retrieved and decrypted should long-term secret keys or passwords be compromised in the future. Not all web servers provide forward secrecy.[21]

A site must be completely hosted over HTTPS, without having part of its contents loaded over HTTP—for example, having scripts loaded insecurely—or the user will be vulnerable to some attacks and surveillance. Also having only a certain page that contains sensitive information (such as a log-in page) of a website loaded over HTTPS, while having the rest of the website loaded over plain HTTP, will expose the user to attacks. On a site that has sensitive information somewhere on it, every time that site is accessed with HTTP instead of HTTPS, the user and the session will get exposed. Similarly, cookies on a site served through HTTPS have to have the secure attribute enabled.[12]

Technical

Difference from HTTP

HTTPS URLs begin with "https://" and use port 443 by default, whereas HTTP URLs begin with "http://" and use port 80 by default.

HTTP is not encrypted and is vulnerable to man-in-the-middle and eavesdropping attacks, which can let attackers gain access to website accounts and sensitive information, and modify webpages to inject malware or advertisements. HTTPS is designed to withstand such attacks and is considered secure against them (with the exception of older, deprecated versions of SSL).

Network layers

HTTP operates at the highest layer of the TCP/IP model, the Application layer; as does the TLS security protocol (operating as a lower sublayer of the same layer), which encrypts an HTTP message prior to transmission and decrypts a message upon arrival. Strictly speaking, HTTPS is not a separate protocol, but refers to use of ordinary HTTP over an encrypted SSL/TLS connection.

Everything in the HTTPS message is encrypted, including the headers, and the request/response load. With the exception of the possible CCA cryptographic attack described in the limitations section below, the attacker can only know that a connection is taking place between the two parties and their domain names and IP addresses.

Server setup

To prepare a web server to accept HTTPS connections, the administrator must create a public key certificate for the web server. This certificate must be signed by a trusted certificate authority for the web browser to accept it without warning. The authority certifies that the certificate holder is the operator of the web server that presents it. Web browsers are generally distributed with a list of signing certificates of major certificate authorities so that they can verify certificates signed by them.

Acquiring certificates

Let's Encrypt, launched in April 2016,[22] provides free and automated SSL/TLS certificates to websites.[23] According to the Electronic Frontier Foundation, "Let's Encrypt" will make switching from HTTP to HTTPS "as easy as issuing one command, or clicking one button."[24]. The majority of web hosts and cloud providers already leverage Let's Encrypt, providing free certificates to their customers.

Use as access control

The system can also be used for client authentication in order to limit access to a web server to authorized users. To do this, the site administrator typically creates a certificate for each user, a certificate that is loaded into their browser. Normally, that contains the name and e-mail address of the authorized user and is automatically checked by the server on each reconnect to verify the user's identity, potentially without even entering a password.

In case of compromised secret (private) key

An important property in this context is perfect forward secrecy (PFS). Possessing one of the long-term asymmetric secret keys used to establish an HTTPS session should not make it easier to derive the short-term session key to then decrypt the conversation, even at a later time. Diffie–Hellman key exchange (DHE) and Elliptic curve Diffie–Hellman key exchange (ECDHE) are in 2013 the only ones known to have that property. Only 30% of Firefox, Opera, and Chromium Browser sessions use it, and nearly 0% of Apple's Safari and Microsoft Internet Explorersessions.[21] Among the larger internet providers, only Google supports PFS since 2011 (State of September 2013).

A certificate may be revoked before it expires, for example because the secrecy of the private key has been compromised. Newer versions of popular browsers such as Firefox,[25] Opera,[26] and Internet Explorer on Windows Vista[27] implement the Online Certificate Status Protocol (OCSP) to verify that this is not the case. The browser sends the certificate's serial number to the certificate authority or its delegate via OCSP and the authority responds, telling the browser whether the certificate is still valid.[28]

Limitations

SSL and TLS encryption can be configured in two modes: *simple* and *mutual.* In simple mode, authentication is only performed by the server. The mutual version requires the user to install a personal client certificate in the web browser for user authentication.[29] In either case, the level of protection depends on the correctness of the implementation of software and the cryptographic algorithms in use.

SSL/TLS does not prevent the indexing of the site by a web crawler, and in some cases the URI of the encrypted resource can be inferred by knowing only the intercepted request/response size.[30] This allows an attacker to have access to the plaintext (the publicly available static content), and the encrypted text (the encrypted version of the static content), permitting a cryptographic attack.

Because TLS operates at a protocol level below that of HTTP, and has no knowledge of the higher-level protocols, TLS servers can only strictly present one certificate for a particular address and port combination.[31] In the past, this meant that it was not feasible to use name-based virtual hostingwith HTTPS. A solution called Server Name Indication(SNI) exists, which sends the hostname to the server before encrypting the connection, although many old browsers do not support this extension. Support for SNI is available sinc Firefox 2, Opera 8, Safari 2.1, Google Chrome 6, andInternet Explorer 7on Windows Vista.[32][33][34]

From an architectural point of view:

- An SSL/TLS connection is managed by the first front machine that initiates the TLS connection. If, for any reasons (routing, traffic optimization, etc.), this front machine is not the application server and it has to decipher data, solutions have to be found to propagate user authentication information or certificate to the application server which needs to know who is going to be connected.
- For SSL/TLS with mutual authentication, the SSL/TLS session is managed by the first server that initiates the connection. In situations where encryption has to be propagated along chained servers, session timeOut management becomes extremely tricky to implement.

1. With mutual SSL/TLS, security is maximal, but on the client-side, there is no way to properly end the SSL/TLS connection and disconnect the user except by waiting for the server session to expire or closing all related client applications.

A sophisticated type of man-in-the-middle attack called SSL stripping was presented at the Blackhat Conference 2009. This type of attack defeats the security provided by HTTPS by changing the `https:` link into an `http:` link, taking advantage of the fact that few Internet users actually type "https" into their browser interface: they get to a secure site by clicking on a link, and thus are fooled into thinking that they are using HTTPS when in fact they are using HTTP. The attacker then communicates in clear with the client.[35] This prompted the development of a countermeasure in HTTP called HTTP Strict Transport Security.

HTTPS has been shown vulnerable to a range of traffic analysis attacks. Traffic analysis attacks are a type of side-channel attack that relies on variations in the timing and size of traffic in order to infer properties about the encrypted traffic itself. Traffic analysis is possible because SSL/TLS encryption changes the contents of traffic, but has minimal impact on the size and timing of traffic. In May 2010, a research paper by researchers from Microsoft Research and Indiana University discovered that detailed sensitive user data can be inferred from side channels such as packet sizes. More specifically, the researchers found that an eavesdropper can infer the illnesses/medications/surgeries of the user, his/her family income and investment secrets, despite HTTPS protection in several high-profile, top-of-the-line web applications in healthcare, taxation, investment and web search.[36] Although this work demonstrated vulnerability of HTTPS to traffic analysis, the approach presented by the authors required manual analysis and focused specifically on web applications protected by HTTPS.

The fact that most modern websites, including Google, Yahoo!, and Amazon, use HTTPS causes problems for many users trying to access public Wi-Fi hot spots, because a Wi-Fi hot spot login page fails to load if the user tries to open an HTTPS resource.[37][38] Several websites, such as neverssl.com or nonhttps.com, guarantee that they will always remain accessible by HTTP.

History

Netscape Communications created HTTPS in 1994 for its Netscape Navigator web browser.[39] Originally, HTTPS was used with the SSL protocol. As SSL evolved into Transport Layer Security (TLS), HTTPS was formally specified by RFC 2818 in May 2000.

See also

- Bullrun (decryption program) – a secret anti-encryption program run by the U.S. National Security Agency
- Computer security
- curl-loader
- Diameter protocol
- HTTPsec
- Moxie Marlinspike
- Opportunistic encryption
- Stunnel

References

1. "Secure your site with HTTPS" (https://support.google.com/webmasters/answer/6073543?hl=en). *Google Support*. Google, Inc. Retrieved February 27, 2015.
2. "What is HTTPS?" (https://www.instantssl.com/ssl-certificate-products/https.html). Comodo CA Limited. Retrieved February 27, 2015. "Hyper Text Transfer Protocol Secure (HTTPS) is the secure version of HTTP [...]"
3. Network Working Group (May 2000). "HTTP Over TLS" (https://tools.ietf.org/html/rfc2818). The Internet Engineering Task Force. Retrieved February 27, 2015.
4. "HTTPS Everywhere FAQ" (https://www.eff.org/https-everywhere/faq). Retrieved 3 May 2012.
5. "A secure web is here to stay" (https://security.googleblog.com/2018/02/a-secure-web-is-here-to-stay.html). *Google Online Security Blog*. Retrieved 2018-07-13.
6. "Hotel Wifi JavaScript Injection" (https://justinsomnia.org/2012/04/hotel-wifi-javascript-injection/). Retrieved 24 July 2012.

7. The Tor Project, Inc. "Tor" (https://www.torproject.org/projects/torbrowser.html.en). *torproject.org*.
8. Konigsburg, Eitan; Pant, Rajiv; Kvochko, Elena (November 13, 2014). "Embracing HTTPS" (https://open.blogs.nytimes.com/2014/11/13/embracing-https/) *The New York Times*. Retrieved February 27, 2015.
9. Gallagher, Kevin (September 12, 2014). "Fifteen Months After the NSA Revelations, Why Aren't More News Organizations Using HTTPS?" (https://freedom.press/blog/2014/09/after-nsa-revelations-why-arent-more-news-organizations-using-https) Freedom of the Press Foundation. Retrieved February 27, 2015.
10. "HTTPS as a ranking signal" (https://googlewebmastercentral.blogspot.com/2014/08/https-as-ranking-signal.html) *Google Webmaster Central Blog* Google Inc. August 6, 2014 Retrieved February 27, 2015. "You can make your site secure with HTTPS (Hypertext Transfer Protocol Secure) [...]"
11. Grigorik, Ilya; Far, Pierre (June 26, 2014). "Google I/O 2014 - HTTPS Everywhere" (https://www.youtube.com/watch?v=cBhZ6S0PFCY). Google Developers Retrieved February 27, 2015.
12. "How to Deploy HTTPS Correctly" (https://www.eff.org/https-everywhere/deploying-https) Retrieved 13 June 2012.
13. "HTTP Strict Transport Security" (https://developer.mozilla.org/en/Security/HTTP_Strict_Transport_Security) *Mozilla Developer Network*
14. "HTTPS usage statistics on top websites" (https://statoperator.com/research/https-usage-statistics-on-top-websites/) *statoperator.com*. Retrieved 25 April 2018.
15. "Qualys SSL Labs - SSL Pulse" (https://www.ssllabs.com/ssl-pulse/) *www.ssllabs.com*. 3 April 2018. Retrieved 25 April 2018.
16. "Let's Encrypt Stats -Let's Encrypt - Free SSL/TLS Certificates" (https://letsencrypt.org/stats/) *letsencrypt.org*. Retrieved 25 April 2018.
17. Peter Eckersley: Encrypt the Web with the HTTPS Everywhere Firefox Extension (https://www.eff.org/deeplinks/2010/06/encrypt-web-https-everywhere-firefox-extension) EFF blog, 17 June 2010
18. HTTPS Everywhere (https://www.eff.org/https-everywhere) EFF projects
19. Law Enforcement Appliance Subverts SSL (https://www.wired.com/threatlevel/2010/03/packet-forensics/), Wired, 3 April 2010.
20. New Research Suggests That Governments May Fake SSL Certificates (https://www.eff.org/deeplinks/2010/03/researchers-reveal-likelihood-governments-fake-ssl) EFF, 2010-03-24.
21. SSL: Intercepted today, decrypted tomorrow (https://news.netcraft.com/archives/2013/06/25/ssl-intercepted-today-decrypted-tomorrow.html), Netcraft, 25 June 2013.
22. Catalin Cimpanu. "Let's Encrypt Launched Today, Currently Protects 3.8 Million Domains" (http://news.softpedia.com/news/let-s-encrypt-launched-today-currently-protects-3-8-million-domains-502857.shtml) Softpedia News. Retrieved April 12, 2016.
23. Kerner, Sean Michael (November 18, 2014). "Let's Encrypt Effort Aims to Improve Internet Security" (http://www.eweek.com/security/lets-encrypt-effort-aims-to-improve-internet-security.html). *eWeek.com*. Quinstreet Enterprise Retrieved February 27, 2015.
24. Eckersley, Peter (November 18, 2014). "Launching in 2015: A Certificate Authority to Encrypt the Entire Web" (https://www.eff.org/deeplinks/2014/11/certificate-authority-encrypt-entire-web) Electronic Frontier Foundation Retrieved February 27, 2015.
25. "Mozilla Firefox Privacy Policy" (https://www.mozilla.org/en-US/privacy/) Mozilla Foundation. 27 April 2009. Retrieved 20 May 2018.
26. "Opera 8 launched on FTP" (http://news.softpedia.com/news/Opera-8-launched-on-FTP-1330.shtml) Softpedia. 19 April 2005. Retrieved 13 May 2009.
27. Lawrence, Eric (31 January 2006). "HTTPS Security Improvements in Internet Explorer 7" (https://msdn.microsoft.com/en-us/library/bb250503.aspx) MSDN. Retrieved 13 May 2009.
28. Myers, M; Ankney, R; Malpani, A; Galperin, S; Adams, C (June 1999). "Online Certificate Status Protocol – OCSP" (https://tools.ietf.org/html/rfc2560) Internet Engineering Task Force. Retrieved 13 May 2009.
29. "Manage client certificates on Chrome devices – Chrome for business and education Help" (https://support.google.com/chrome/a/answer/6080885?hl=en) *support.google.com*. Retrieved 13 October 2016.
30. Pusep, Stanislaw (31 July 2008). "The Pirate Bay un-SSL" (https://www.exploit-db.com/docs/english/13026-the-pirate-bay-un-ssl.pdf) (PDF). Retrieved 20 May 2018.
31. "SSL/TLS Strong Encryption: FAQ" (https://httpd.apache.org/docs/2.0/ssl/ssl_faq.html#vhosts) *apache.org*.

32. Lawrence, Eric (22 October 2005). "Upcoming HTTPS Improvements in Internet Explorer 7 Beta 2" (https://blogs.msdn.com/ie/archive/2005/10/22/483795.aspx). Microsoft. Retrieved 12 May 2009.
33. "Server Name Indication (SNI)" (http://blog.ebrahim.org/2006/02/21/server-name-indication-sni/). *inside aebrahim's head*.
34. Pierre, Julien (19 December 2001). "Browser support for TLS server name indication" (https://bugzilla.mozilla.org/show_bug.cgi?id=116169). *Bugzilla*. Mozilla Foundation. Retrieved 15 December 2010.
35. "sslstrip" (http://www.thoughtcrime.org/software/sslstrip/index.html). Retrieved 26 November 2011.
36. Shuo Chen; Rui Wang; XiaoFeng Wang; Kehuan Zhang (May 2010). "Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow" (https://research.microsoft.com/pubs/119060/WebAppSideChannel-final.pdf) (PDF). IEEE Symposium on Security & Privacy 2010.
37. "How to Force a Public Wi-Fi Network Login Page to Open" (https://zapier.com/blog/open-wifi-login-page/). Retrieved 30 December 2017.
38. "iOS: What To Do When Your Public Wi-Fi Won't Connect All the Way" (https://mashtips.com/solve-wifi-login-page-not-loading-issue-ios/). Retrieved 30 December 2017.
39. Walls, Colin (2005). *Embedded software* (https://books.google.com/books?id=FLvsis4_QhEC&pg=PA344). Newnes. p. 344. ISBN 0-7506-7954-9.

External links

- RFC 2818: HTTP Over TLS
- RFC 5246: The Transport Layer Security Protocol 1.2
- RFC 6101: The Secure Sockets Layer (SSL) Protocol Version 3.0