HTTPS

Hypertext Transfer Protocol Secure (**HTTPS**) is an extension of the Hypertext Transfer Protocol (HTTP). It uses encryption 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, Secure Sockets Layer (SSL). The protocol is therefore also referred to as **HTTP over TLS**,[3] or **HTTP over SSL**.

The principal motivations for HTTPS are authentication of the accessed website and protection of the privacy and integrity of the exchanged data while it is in transit. It protects against man-in-the-middle attacks, and the bidirectional block cipher encryption of communications between a client and server protects the communications against eavesdropping and tampering.[4][5] The authentication aspect of HTTPS requires a trusted third party to sign server-side digital certificates. This was historically an expensive operation, which meant fully authenticated HTTPS connections were usually found only on secured payment transaction services and other secured corporate information systems on the World Wide Web. In 2016, a campaign by the Electronic Frontier Foundation with the support of web browser developers led to the protocol becoming more prevalent.[6] HTTPS is now 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.

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).

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

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's URL, query parameters, headers, and cookies (which often contain identifying information about the user). However, because 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, and sometimes even the domain name (e.g. www.example.org, but not the rest of the URL) that a user is communicating with, along with the amount of data transferred and the duration of the communication, though not the content of the communication.[4]

Web browsers know how to trust HTTPS websites based on certificate authorities that come pre-installed in their software. Certificate authorities 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 their device, hosting the browser and the method to get the browser itself, is not compromised (i.e. there is no supply chain attack).
- 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 (i.e. the certificate authority is not compromised and there is no mis-issuance of certificates).
- 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 and networks that may be subject to tampering. Insecure networks, such as public Wi-Fi access points, allow anyone on the same local network to packet-sniff and discover sensitive information not protected by HTTPS. Additionally, some free-to-use and paid WLAN networks have been observed tampering with webpages by engaging in packet injection in order to serve their own ads on other websites. This practice can be exploited maliciously in many ways, such as by injecting malware onto webpages and stealing users' private information.[7]

HTTPS is also important for connections over the Tor network, as malicious Tor nodes could otherwise 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 Tor Browser.[8]

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.[9][10] Even though metadata about individual pages that a user visits might not be considered sensitive, when aggregated it can reveal a lot about the user and compromise the user's privacy.[11][12][13]

Deploying HTTPS also allows the use of HTTP/2 and HTTP/3 (and their predecessors SPDY and QUIC), which are new HTTP versions 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.[13][14]

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

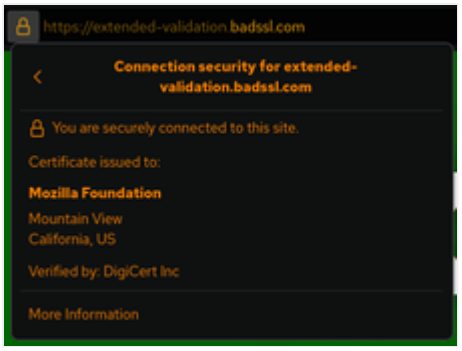
Usage in websites

As of April 2018, 33.2% of Alexa top 1,000,000 websites use HTTPS as default[15] and 70% of page loads (measured by Firefox Telemetry) use HTTPS.[16] As of December 2022, 58.4% of the Internet's 135,422 most popular websites have a secure implementation of HTTPS,[17] However, despite TLS 1.3's release in 2018, adoption has been slow, with many still remaining on the older TLS 1.2 protocol.[18]

Browser integration

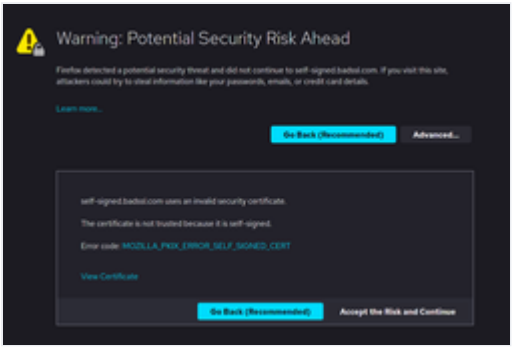
Most browsers display a warning if they receive an invalid certificate. Older browsers, when connecting to a site with an invalid certificate, would present the user with a dialog box asking whether they wanted to continue. Newer browsers display a warning across the entire window. Newer browsers also prominently display the site's security information in the address bar. Extended validation certificates show the legal entity on the certificate information. Most browsers also display a warning to the user when visiting a site that contains a mixture of encrypted and unencrypted content. Additionally, many web filters return a security warning when visiting prohibited websites.

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, an Extended Validation Certificate should identify the legal entity for the certificate.

When accessing a site only with a common certificate, on the address bar of Firefox and other browsers, a "lock" sign appears.



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, Google Chrome, Chromium, and Android, which enables HTTPS by default for hundreds of frequently used

websites.[19][20]

Forcing a web browser to load only HTTPS content has been supported in Firefox starting in version 83.[21] Starting in version 94, Google Chrome is able to "always use secure connections" if toggled in the browser's settings.[22][23]

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 the client and the 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.[24][25] 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.[26]

For HTTPS to be effective, a site must be completely hosted over HTTPS. If some of the site's contents are loaded over HTTP (scripts or images, for example), or if only a certain page that contains sensitive information, such as a log-in page, is loaded over HTTPS while the rest of the site is loaded over plain HTTP, the user will be vulnerable to attacks and surveillance. Additionally, cookies on a site served through HTTPS must have the secure attribute enabled. On a site that has sensitive information on it, the user and the session will get exposed every time that site is accessed with HTTP instead of HTTPS.[13]

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 thus 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 HTTPS implementations that use 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 the use of ordinary HTTP over an encrypted SSL/TLS connection.

HTTPS encrypts all message contents, including the HTTP headers and the request/response data. With the exception of the possible CCA cryptographic attack described in the limitations section below, an attacker should at most be able to discover that a connection is taking place between two parties, along with 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

A number of commercial certificate authorities exist, offering paid-for SSL/TLS certificates of a number of types, including Extended Validation Certificates.

Let's Encrypt, launched in April 2016,[27] provides free and automated service that delivers basic SSL/TLS certificates to websites.[28] 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."[29] The majority of web hosts and cloud providers now 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, which the user loads into their browser. Normally, the certificate contains the name and e-mail address of the authorized user and is automatically checked by the server on each connection to verify the user's identity, potentially without even requiring 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 schemes known to have that property. In 2013, only 30% of Firefox, Opera, and Chromium Browser sessions used it, and nearly 0% of Apple's Safari and Microsoft Internet Explorer sessions.[26] TLS 1.3, published in August 2018, dropped support for ciphers without forward secrecy. As of February 2019, 96.6% of web servers surveyed support some form of forward secrecy, and 52.1% will use forward secrecy with most browsers.[30] As of July 2023, 99.6% of web servers surveyed support some form of forward secrecy, and 75.2% will use forward secrecy with most browsers.[31]

Certificate revocation

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,[32] Opera,[33] and Internet Explorer on Windows Vista[34] 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 (Online Certificate Status Protocol) and the authority responds, telling the browser whether the certificate is still valid or not.[35] The CA may also issue a CRL to tell people that these certificates are revoked. CRLs are no longer required by the CA/Browser forum,[36] nevertheless, they are still widely used by the CAs. Most revocation statuses on the Internet disappear soon after the expiration of the certificates.[37]

Limitations

SSL (Secure Sockets Layer) and TLS (Transport Layer Security) 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.[38] In either case, the level of protection depends on the correctness of the implementation of the 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.[39] 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.[40] In the past, this meant that it was not feasible to use name-based virtual hosting with 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 since Firefox 2, Opera 8, Apple Safari 2.1, Google Chrome 6, and Internet Explorer 7 on Windows Vista.[41][42][43]

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.
- Security is maximal with mutual SSL/TLS, 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 by closing all related client applications.

A sophisticated type of man-in-the-middle attack called SSL stripping was presented at the 2009 Blackhat Conference. 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.[44] This prompted the development of a countermeasure in HTTP called HTTP Strict Transport Security.

HTTPS has been shown to be 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. The researchers found that, despite HTTPS protection in several high-profile, top-of-the-line web applications in healthcare, taxation, investment, and web search, an eavesdropper could infer the illnesses/medications/surgeries of the user, his/her family income, and investment secrets.[45] Although this work demonstrated the 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.[46] Several websites, such as NeverSSL,[47] guarantee that they will always remain accessible by HTTP.[48]

History

Netscape Communications created HTTPS in 1994 for its Netscape Navigator web browser.[49] 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. Google announced in February 2018 that its Chrome browser would mark HTTP sites as "Not Secure" after July 2018.[50] This move was to encourage website owners to implement HTTPS, as an effort to make the World Wide Web more secure.

See also

- Transport Layer Security
- Bullrun (decryption program) – a secret anti-encryption program run by the US National Security Agency
- Computer security
- HSTS
- Opportunistic encryption
- Stunnel

References

1. "Secure your site with HTTPS" (https://support.google.com/webmasters/answer/6073543?hl=en). *Google Support*. Google Inc. Archived (https://web.archive.org/web/20150301023624/https://support.google.com/webmasters/answer/6073543?hl=en) from the original on 1 March 2015. Retrieved 20 October 2018.

2. "What is HTTPS?" (https://web.archive.org/web/20150212105201/https://www.instantssl.com/ssl-certificate-products/https.html). Comodo CA Limited. Archived from the original on 12 February 2015. Retrieved 20 October 2018. "Hyper Text Transfer Protocol Secure (HTTPS) is the secure version of HTTP [...]"
3. "https URI Scheme" (https://datatracker.ietf.org/doc/html/rfc9110#section-4.2.2). *HTTP Semantics* (https://datatracker.ietf.org/doc/html/rfc9110). IETF. June 2022. sec. 4.2.2. doi:10.17487/RFC9110 (https://doi.org/10.17487%2FRFC9110). RFC 9110 (https://datatracker.ietf.org/doc/html/rfc9110).
4. "HTTPS Everywhere FAQ" (https://www.eff.org/https-everywhere/faq). 8 November 2016. Archived (https://web.archive.org/web/20181114011956/https://www.eff.org/https-everywhere/faq/) from the original on 14 November 2018. Retrieved 20 October 2018.
5. "Usage Statistics of Default protocol https for Websites, July 2019" (https://w3techs.com/technologies/details/ce-httpsdefault/all/all). *w3techs.com*. Archived (https://web.archive.org/web/20190801134536/https://w3techs.com/technologies/details/ce-httpsdefault/all/all) from the original on 1 August 2019. Retrieved 20 July 2019.
6. "Encrypting the Web" (https://www.eff.org/encrypt-the-web). *Electronic Frontier Foundation*. Archived (https://web.archive.org/web/20191118094200/https://www.eff.org/encrypt-the-web) from the original on 18 November 2019. Retrieved 19 November 2019.
7. "Hotel Wifi JavaScript Injection" (https://justinsomnia.org/2012/04/hotel-wifi-javascript-injection/). *JustInsomnia*. 3 April 2012. Archived (https://web.archive.org/web/20181118154608/https://justinsomnia.org/2012/04/hotel-wifi-javascript-injection/) from the original on 18 November 2018. Retrieved 20 October 2018.
8. The Tor Project, Inc. "What is Tor Browser?" (https://www.torproject.org/projects/torbrowser.html.en). *TorProject.org*. Archived (https://archive.today/20130717222709/https://www.torproject.org/projects/torbrowser.html.en) from the original on 17 July 2013. Retrieved 30 May 2012.
9. Konigsburg, Eitan; Pant, Rajiv; Kvochko, Elena (13 November 2014). "Embracing HTTPS" (https://open.blogs.nytimes.com/2014/11/13/embracing-https/). *The New York Times*. Archived (https://web.archive.org/web/20190108190000/https://open.blogs.nytimes.com/2014/11/13/embracing-https/) from the original on 8 January 2019. Retrieved 20 October 2018.
10. Gallagher, Kevin (12 September 2014). "Fifteen Months After the NSA Revelations, Why Aren't More News Organizations Using HTTPS?" (https://freedom.press/news-advocacy/fifteen-months-after-the-nsa-revelations-why-arenat-more-news-organizations-using-https/). Freedom of the Press Foundation. Archived (https://web.archive.org/web/20180810204919/https://freedom.press/news-advocacy/fifteen-months-after-the-nsa-revelations-why-arenat-more-news-organizations-using-https/) from the original on 10 August 2018. Retrieved 20 October 2018.
11. "HTTPS as a ranking signal" (https://webmasters.googleblog.com/2014/08/https-as-ranking-signal.html). *Google Webmaster Central Blog*. Google Inc. 6 August 2014. Archived (https://web.archive.org/web/20181017052432/https://webmasters.googleblog.com/2014/08/https-as-ranking-signal.html) from the original on 17 October 2018. Retrieved 20 October 2018. "You can make your site secure with HTTPS (Hypertext Transfer Protocol Secure) [...]"
12. Grigorik, Ilya; Far, Pierre (26 June 2014). "Google I/O 2014 - HTTPS Everywhere" (https://www.youtube.com/watch?v=cBhZ6S0PFCY). Google Developers. Archived (https://web.archive.org/web/20181120144918/https://www.youtube.com/watch?v=cBhZ6S0PFCY) from the original on 20 November 2018. Retrieved 20 October 2018.
13. "How to Deploy HTTPS Correctly" (https://www.eff.org/https-everywhere/deploying-https). 15 November 2010. Archived (https://web.archive.org/web/20181010233702/https://www.eff.org/https-everywhere/deploying-https) from the original on 10 October 2018. Retrieved 20 October 2018.

14. "HTTP Strict Transport Security" (https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security). *Mozilla Developer Network*. Archived (https://web.archive.org/web/20181019171534/https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security) from the original on 19 October 2018. Retrieved 20 October 2018.
15. "HTTPS usage statistics on top 1M websites" (https://statoperator.com/research/https-usage-statistics-on-top-websites/). *StatOperator.com*. Archived (https://web.archive.org/web/20190209055130/https://statoperator.com/research/https-usage-statistics-on-top-websites/) from the original on 9 February 2019. Retrieved 20 October 2018.
16. "Let's Encrypt Stats" (https://letsencrypt.org/stats/). *LetsEncrypt.org*. Archived (https://web.archive.org/web/20181019221028/https://letsencrypt.org/stats/) from the original on 19 October 2018. Retrieved 20 October 2018.
17. "Qualys SSL Labs - SSL Pulse" (https://www.ssllabs.com/ssl-pulse/). *www.ssllabs.com*. 4 December 2022. Archived (https://web.archive.org/web/20221207004823/https://www.ssllabs.com/ssl-pulse/) from the original on 7 December 2022. Retrieved 7 December 2022..
18. "TLS 1.3: Slow adoption of stronger web encryption is empowering the bad guys" (https://www.helpnetsecurity.com/2020/04/06/tls-1-3-adoption/). *Help Net Security*. 6 April 2020. Archived (https://web.archive.org/web/20220524002257/https://www.helpnetsecurity.com/2020/04/06/tls-1-3-adoption/) from the original on 24 May 2022. Retrieved 23 May 2022.
19. Eckersley, Peter (17 June 2010). "Encrypt the Web with the HTTPS Everywhere Firefox Extension" (https://www.eff.org/deeplinks/2010/06/encrypt-web-https-everywhere-firefox-extension). *EFF blog*. Archived (https://web.archive.org/web/20181125102636/https://www.eff.org/deeplinks/2010/06/encrypt-web-https-everywhere-firefox-extension) from the original on 25 November 2018. Retrieved 20 October 2018.
20. "HTTPS Everywhere" (https://www.eff.org/https-everywhere). *EFF projects*. 7 October 2011. Archived (https://web.archive.org/web/20110605022218/https://www.eff.org/https-everywhere) from the original on 5 June 2011. Retrieved 20 October 2018.
21. "HTTPS-Only Mode in Firefox" (https://support.mozilla.org/en-US/kb/https-only-prefs). Archived (https://web.archive.org/web/20211112222245/https://support.mozilla.org/en-US/kb/https-only-prefs) from the original on 12 November 2021. Retrieved 12 November 2021.
22. "Manage Chrome safety and security - Android - Google Chrome Help" (https://support.google.com/chrome/answer/10468685?hl=en&co=GENIE.Platform=Android). *support.google.com*. Archived (https://web.archive.org/web/20220307190622/https://support.google.com/chrome/answer/10468685?hl=en&co=GENIE.Platform=Android) from the original on 7 March 2022. Retrieved 7 March 2022.
23. "Hands on Chrome's HTTPS-First Mode" (https://techdows.com/2021/07/hands-on-chromes-https-first-mode.html). *Techdows*. 19 July 2021. Archived (https://web.archive.org/web/20220307190617/https://techdows.com/2021/07/hands-on-chromes-https-first-mode.html) from the original on 7 March 2022. Retrieved 7 March 2022.
24. Singel, Ryan (24 March 2010). "Law Enforcement Appliance Subverts SSL" (https://www.wired.com/2010/03/packet-forensics/). *Wired*. Archived (https://web.archive.org/web/20190117142906/https://www.wired.com/2010/03/packet-forensics/) from the original on 17 January 2019. Retrieved 20 October 2018.
25. Schoen, Seth (24 March 2010). "New Research Suggests That Governments May Fake SSL Certificates" (https://www.eff.org/deeplinks/2010/03/researchers-reveal-likelihood-governments-fake-ssl). *EFF*. Archived (https://web.archive.org/web/20160104234608/https://www.eff.org/deeplinks/2010/03/researchers-reveal-likelihood-governments-fake-ssl) from the original on 4 January 2016. Retrieved 20 October 2018.

26. Duncan, Robert (25 June 2013). "SSL: Intercepted today, decrypted tomorrow" (https://news.netcraft.com/archives/2013/06/25/ssl-intercepted-today-decrypted-tomorrow.html). *Netcraft*. Archived (https://web.archive.org/web/20181006021916/https://news.netcraft.com/archives/2013/06/25/ssl-intercepted-today-decrypted-tomorrow.html) from the original on 6 October 2018. Retrieved 20 October 2018.
27. Cimpanu, Catalin (12 April 2016). "Let's Encrypt Launched Today, Currently Protects 3.8 Million Domains" (https://news.softpedia.com/news/let-s-encrypt-launched-today-currently-protects-3-8-million-domains-502857.shtml). Softpedia News. Archived (https://web.archive.org/web/20190209055129/https://news.softpedia.com/news/let-s-encrypt-launched-today-currently-protects-3-8-million-domains-502857.shtml) from the original on 9 February 2019. Retrieved 20 October 2018.
28. Kerner, Sean Michael (18 November 2014). "Let's Encrypt Effort Aims to Improve Internet Security" (http://www.eweek.com/security/let-s-encrypt-effort-aims-to-improve-internet-security). *eWeek.com*. Quinstreet Enterprise. Archived (https://web.archive.org/web/20230402154948/https://www.eweek.com/security/let-s-encrypt-effort-aims-to-improve-internet-security/) from the original on 2 April 2023. Retrieved 20 October 2018.
29. Eckersley, Peter (18 November 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. Archived (https://web.archive.org/web/20181118160126/https://www.eff.org/deeplinks/2014/11/certificate-authority-encrypt-entire-web) from the original on 18 November 2018. Retrieved 20 October 2018.
30. Qualys SSL Labs. "SSL Pulse" (https://web.archive.org/web/20190215213454/https://www.ssllabs.com/ssl-pulse/). Archived from the original (https://www.ssllabs.com/ssl-pulse/) (3 February 2019) on 15 February 2019. Retrieved 25 February 2019.
31. "Qualys SSL Labs - SSL Pulse" (https://www.ssllabs.com/ssl-pulse/). *www.ssllabs.com*. Retrieved 4 September 2023.
32. "Mozilla Firefox Privacy Policy" (https://www.mozilla.org/en-US/privacy/). Mozilla Foundation. 27 April 2009. Archived (https://web.archive.org/web/20181018063732/https://www.mozilla.org/en-US/privacy/) from the original on 18 October 2018. Retrieved 20 October 2018.
33. "Opera 8 launched on FTP" (https://news.softpedia.com/news/Opera-8-launched-on-FTP-1330.shtml). Softpedia. 19 April 2005. Archived (https://web.archive.org/web/20190209055128/https://news.softpedia.com/news/Opera-8-launched-on-FTP-1330.shtml) from the original on 9 February 2019. Retrieved 20 October 2018.
34. Lawrence, Eric (31 January 2006). "HTTPS Security Improvements in Internet Explorer 7" (https://docs.microsoft.com/en-us/previous-versions/aa980989(v=msdn.10)). *Microsoft Docs*. Archived (https://web.archive.org/web/20211024181937/https://docs.microsoft.com/en-us/previous-versions/aa980989(v=msdn.10)) from the original on 24 October 2021. Retrieved 24 October 2021.
35. Myers, Michael; Ankney, Rich; Malpani, Ambarish; Galperin, Slava; Adams, Carlisle (20 June 1999). "Online Certificate Status Protocol – OCSP" (https://tools.ietf.org/html/rfc2560). Internet Engineering Task Force. doi:10.17487/RFC2560 (https://doi.org/10.17487%2FRFC2560). Archived (https://web.archive.org/web/20110825095059/http://tools.ietf.org/html/rfc2560) from the original on 25 August 2011. Retrieved 20 October 2018.
36. "Baseline Requirements" (https://cabforum.org/baseline-requirements-documents/). CAB Forum. 4 September 2013. Archived (https://web.archive.org/web/20141020234802/https://cabforum.org/baseline-requirements-documents/) from the original on 20 October 2014. Retrieved 1 November 2021.

37. Korzhitskii, N.; Carlsson, N. (30 March 2021). "Revocation Statuses on the Internet". *Passive and Active Measurement*. Lecture Notes in Computer Science. Vol. 12671. pp. 175–191. arXiv:2102.04288 (https://arxiv.org/abs/2102.04288). doi:10.1007/978-3-030-72582-2_11 (https://doi.org/10.1007%2F978-3-030-72582-2_11). ISBN 978-3-030-72581-5.
38. "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*. Archived (https://web.archive.org/web/20190209055127/https://support.google.com/chrome/a/answer/6080885?hl=en) from the original on 9 February 2019. Retrieved 20 October 2018.
39. 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). Archived (https://web.archive.org/web/20180620001518/https://www.exploit-db.com/docs/english/13026-the-pirate-bay-un-ssl.pdf) (PDF) from the original on 20 June 2018. Retrieved 20 October 2018.
40. "SSL/TLS Strong Encryption: FAQ" (https://httpd.apache.org/docs/2.0/ssl/ssl_faq.html#vhosts). *apache.org*. Archived (https://web.archive.org/web/20181019105423/http://httpd.apache.org/docs/2.0/ssl/ssl_faq.html#vhosts) from the original on 19 October 2018. Retrieved 20 October 2018.
41. Lawrence, Eric (22 October 2005). "Upcoming HTTPS Improvements in Internet Explorer 7 Beta 2" (https://blogs.msdn.microsoft.com/ie/2005/10/22/upcoming-https-improvements-in-internet-explorer-7-beta-2/). Microsoft. Archived (https://web.archive.org/web/20180920113838/https://blogs.msdn.microsoft.com/ie/2005/10/22/upcoming-https-improvements-in-internet-explorer-7-beta-2/) from the original on 20 September 2018. Retrieved 20 October 2018.
42. "Server Name Indication (SNI)" (https://blog.ebrahim.org/2006/02/21/server-name-indication-sni/). *inside aebrahim's head*. 21 February 2006. Archived (https://web.archive.org/web/20180810173628/https://blog.ebrahim.org/2006/02/21/server-name-indication-sni/) from the original on 10 August 2018. Retrieved 20 October 2018.
43. Pierre, Julien (19 December 2001). "Browser support for TLS server name indication" (https://bugzilla.mozilla.org/show_bug.cgi?id=116169). *Bugzilla*. Mozilla Foundation. Archived (https://web.archive.org/web/20181008070112/https://bugzilla.mozilla.org/show_bug.cgi?id=116169) from the original on 8 October 2018. Retrieved 20 October 2018.
44. "sslstrip 0.9" (https://moxie.org/software/sslstrip/index.html). Archived (https://web.archive.org/web/20180620042059/https://moxie.org/software/sslstrip/index.html) from the original on 20 June 2018. Retrieved 20 October 2018.
45. Shuo Chen; Rui Wang; XiaoFeng Wang; Kehuan Zhang (20 May 2010). "Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow" (https://www.microsoft.com/en-us/research/publication/side-channel-leaks-in-web-applications-a-reality-today-a-challenge-tomorrow/). *Microsoft Research*. IEEE Symposium on Security & Privacy 2010. Archived (https://web.archive.org/web/20180722120329/https://www.microsoft.com/en-us/research/publication/side-channel-leaks-in-web-applications-a-reality-today-a-challenge-tomorrow/) from the original on 22 July 2018. Retrieved 20 October 2018.
46. Guaay, Matthew (21 September 2017). "How to Force a Public Wi-Fi Network Login Page to Open" (https://zapier.com/blog/open-wifi-login-page/). Archived (https://web.archive.org/web/20180810143254/https://zapier.com/blog/open-wifi-login-page/) from the original on 10 August 2018. Retrieved 20 October 2018.
47. "NeverSSL" (http://neverssl.com).
48. "NeverSSL" (http://neverssl.com/). Archived (https://web.archive.org/web/20180901224536/http://neverssl.com/) from the original on 1 September 2018. Retrieved 20 October 2018.
49. Walls, Colin (2005). *Embedded Software: The Works* (https://books.google.com/books?id=FLvsis4_QhEC&pg=PA344). Newnes. p. 344. ISBN 0-7506-7954-9. Archived (https://web.archive.org/web/20190209055124/https://books.google.com/books?id=FLvsis4_QhEC&pg=PA344) from the original on 9 February 2019. Retrieved 20 October 2018.

50. "A secure web is here to stay" (https://blog.chromium.org/2018/02/a-secure-web-is-here-to-stay.html). *Chromium Blog*. Archived (https://web.archive.org/web/20190424215132/https://blog.chromium.org/2018/02/a-secure-web-is-here-to-stay.html) from the original on 24 April 2019. Retrieved 22 April 2019.

External links

- RFC 8446 (https://datatracker.ietf.org/doc/html/rfc8446): The Transport Layer Security (TLS) Protocol Version 1.3
