Evolution of SSL/TLS Indicators and Warnings in Web Browsers

Lydia Kraus[1], Martin Ukrop[1][0000−0001−8110−8926], Vashek Matyas[1], and Tobias Fiebig[2][0000−0002−0163−5134]

[1] Masaryk University, Brno, Czech Republic
{lydia.kraus,mukrop}@mail.muni.cz, matyas@fi.muni.cz
[2] TU Delft, Delft, Netherlands
T.Fiebig@tudelft.nl

Abstract. The creation of the World Wide Web (WWW) in the early 1990's finally made the Internet accessible to a wider part of the population[3]. With this increase in users, security became more important. To address confidentiality and integrity requirements on the web, Netscape—by then a major web browser vendor—presented the Secure Socket Layer (SSL), later versions of which were renamed to Transport Layer Security (TLS). In turn, this necessitated the introduction of both security indicators in browsers to inform users about the TLS connection state and also of warnings to inform users about potential errors in the TLS connection to a website. Looking at the evolution of indicators and warnings, we find that the qualitative data on security indicators and warnings, i.e., screen shots of different browsers over time is inconsistent. Hence, in this paper we outline our methodology for collecting a comprehensive data set of web browser security indicators and warnings, which will enable researchers to better understand how security indicators and TLS warnings in web browsers evolved over time.

1 Introduction

The emergence of the World Wide Web (WWW) and the development of the first browser in the beginning of the 90's made the Internet accessible to the wider public. Today, web browsers are the major interface for users to engage with the Internet. However, availability of Internet access to the wide public also led to the average end-user being a common target of attacks. The common way on the web to ensure confidentiality, integrity, and ideally even identity is TLS (formerly SSL). In the case of web browsers, security indicators and warnings communicate TLS information to end-users, so these can take necessary action, e.g., not visit a site. However, communicating the current TLS security state and implications of TLS errors to end-users in the web browser user interface (UI), enabling them to make the right decisions and take the right actions is a still ongoing challenge.

While usability and user experience are drivers of UI design in consumer-oriented products, they are only one among several influencing factors in the context of security protocol related UIs. Security protocols evolve over time, in a complex process

[3] This is the accepted version of our SPW 2019 paper. The authenticated version is available online at Springer via https://doi.org/10.1007/978-3-030-57043-9_25

that is not only driven by market demands (including usability needs), but also by the interests of diverse actors (e.g., industry, academia, standardization bodies, etc.) [2] and changing security requirements [2, 16].

For web browsers, this has natural implications for their user interface design, specifically for how they inform users about errors and the state of TLS-enabled (and plain text) connections. To better understand today's security and usability challenges, it is important to understand how these indicators and warnings evolved over time and how research, industry, and the practical reality of the Internet influenced this process.

However, until now, there is no consistent historical data set of browsers' ways of reporting SSL/TLS states via indicators and warnings. In practice, the qualitative data, i.e., screen shots of different browsers over time, is inconsistent. Figures found in the literature are temporarily and spatially limited, and online resources are not sufficiently uniform for a consistent comparison. To build a systematic overview of web browsers' communication of TLS states and error conditions in the UI, we need a data set that contains the evolution of TLS security indicators and warnings in web browsers over time. Hence, in this work we make the following contributions towards an open data set of browser UIs—currently focusing only on desktop browsers—in the context of TLS security:

- As the literature does not provide sufficient detail and consistent information on the evolution of browser security indicators and warnings, we propose to build an open data set to enable an in-depth analysis of the evolution.
- To set the ground for us and others to contribute to and to use such a data set, we outline the requirements for a browser TLS UI data set in terms of browsers, browser versions, and TLS errors and states.
- We describe the expected practical challenges in collecting this data in terms of data quality and comparability, including the visual representation of collected screen shots.

The remainder of this paper is structured as follows: In Section 2, we outline the technical background on TLS. In Section 3, we then revisit web browsers, and TLS information display and errors over time. We continue by providing data set requirements, outlining a data collection procedure and discussing its limitations and challenges in Section 4. Here, we address and discuss challenges like the impact of screen resolution and size on the visibility of indicators, and the limited availability of older software versions. Finally, we summarize our results and conclude in Section 5.

2 History and Functionality of SSL/TLS

2.1 History of SSL/TLS

SSL (Secure Socket Layer) 1.0 was first introduced by Netscape Communications in 1994, followed by versions 2.0 in 1995 and 3.0 in 1996 [43]. The protocol was further developed within the IETF, where its successor protocol was renamed to TLS (Transport Layer Security) [11]. Since then, newer versions regularly appear, with TLS 1.1 in 2006 [12], TLS 1.2 in 2008 [13], and TLS 1.3 in 2018 [31]. While the development of new versions was initially driven by features, especially TLS

1.3 introduces new security mechanics and defenses against recently discovered attacks [31, 35] and the discontinuation of by-now insecure algorithms and key-sizes [34]. In addition, existing attacks against TLS [35] led to the deprecation of SSL 2.0 [44] and SSL 3.0 [4] by the IETF. Similarly, TLS 1.0 is no longer an option for systems handling credit card data [29].

High-Level Functionality. In general, a TLS session follows four major steps while being established [11–13, 31]:

1. **Session initiation:** Client and server exchange their capabilities (supported ciphers, etc.)
2. **(Mutual) authentication:** The server authenticates to the client by default. Optionally, the client may also authenticate to the server. Usually, this is done by using the local trust store found on most devices.
3. **Key establishment:** Server and client establish a shared secret, usually using public key cryptography.
4. **Symmetrically encrypted session:** Finally, the established key is used for a session using a symmetric cipher.

2.2 X.509 Certificates, PKI and CAs

An essential part of TLS is the authentication of at least the server by the client. For this purpose, TLS relies on a PKI system where Certificate Authorities (CAs) issue X.509 certificates to parties operating a host with a specific DNS [25] name. Certificates are usually signed by intermediate CAs. Clients, e.g., web browsers then have a vendor-supplied (local/OS-wide) trust-store that contains the root certificates of CAs used to sign those intermediate certificates. Therefore, validating a certificate is also called validating the trust chain.

The process of a CA assuring that an entity is authorized to receive a certificate for a name (has authority over the name) is called validation. Traditionally, there are Domain Validation (DV), Organization Validation (OV), and Extended Validation (EV) [10]. DV usually just requires the certificate requesting party to demonstrate authority over a proxy, e.g., being able to host content on the website a domain points to, or being able to access a mail address associated with a domain [5]. OV requires the issuing CA to verify an organization, while EV requires the CA to follow an extensive guide to verify the requesting entity [10]. Hence, in theory, EV certificates should significantly reduce phishing attacks [21].

2.3 Deployment on the Web

While SSL and TLS adoption has been traditionally slow, recent changes to the PKI ecosystem have led to a surge of TLS enabled websites. The introduction of Let's Encrypt lead to a significant rise in websites using TLS [24] as it removed traditional—mostly cost—barriers to certificate deployment [1]. Similarly, various large browser vendors [19] and the payment card industry [29] increased requirements for TLS.

3 SSL/TLS in Web Browsers in the Present and Past

3.1 Web Browsers

Web browsers are the standard tool to browse the web. As of December 2018, there are at least 30 browsers from different vendors and organizations, with an even richer history of different versions [40]. Among those browsers are desktop and mobile versions of which some are bound to specific operating systems or devices. Mobile browsers became popular with the wider spread of mobile computing devices at the end of the last decade. As the scope of this work are desktop browsers, we use 'browser' and 'desktop browser' interchangeably in the remainder of this work.

From the mid 1990's up to the early years of the new millennium, the browser market used to be dominated first by Netscape Navigator and later by Microsoft Internet Explorer (IE) [26]. In 2004, Mozilla Firefox (FF) was officially released [28] and soon received constant growth, leading to a decreased market share of IE [40]. Soon after the release of the Chrome browser by Google in 2008, the browser landscape changed again significantly with Chrome becoming the most popular browser, see Table 1.

While the set of popular browsers did not change in the last decade, there was a significant change in popularity, with both FF and IE—as well as Edge as its successor—having lost popularity, while Chrome now accounts for nearly $\frac{3}{4}$ of the market, see Table 1. As of December 2018, the most popular browsers are Google's Chrome, followed by Mozilla's Firefox, Microsoft's Internet Explorer, Apple's Safari, Microsoft's Edge, and Opera's Opera.

Browser	**Jan. 2009**	**Dec. 2018**
Internet Explorer (IE)	65.41%	5.31%
Edge	N/A	4.03%
Firefox (FF)	27.30%	10.15%
Chrome	1.38%	71.29%
Safari	2.57%	5.09%
Opera	2.92%	2.18%

Table 1. Market share of most popular web browsers in Jan. 2009 and Dec. 2018 according to statcounter.com [40]

3.2 Web Browser TLS User Interface Standards

In 2010, the World Wide Web Consortium (W3C) has published user interface guidelines for the web security context [32]. These guidelines describe the TLS information that should be communicated to end-users in web user agents (such as web browsers) and provides recommendations on how this information has to be presented.

The guidelines differentiate between the *primary user interface* that is the part of the user interfaces that is directly accessible by the user, and the *secondary interface* that is not directly accessible, e.g., extended settings after clicking an 'options' button. The guidelines require a security indicator, either in the primary or in the secondary user interface in a consistent position, which can not be obscured by web content. They also require the presence of identity information, either in the primary or the secondary user interface. Again, the position must be consistent and the information must only be derived from the validated certificate. In addition, the guidelines explicitly require warnings for self-signed, untrusted root, expired, and revoked certificate errors, or if "TLS negotiation otherwise fails" [32]. While

the guidelines provide a high-level set of requirements and definitions, the actual appearance of indicators and warnings is subject to the decisions of individual browser vendors.

3.3 Web Browser SSL/TLS User Interfaces in Practice

Traditionally, SSL/TLS information has been displayed by means of security indicators and warnings in the *primary UI*. Indicators are passive security signals that inform the user of the TLS security state, including TLS errors. They change their visual appearance based on TLS states (including error states). Major TLS errors are further accompanied by warnings in the UI. Warnings are active security signals: in contrast to indicators, they require users to actively take an action.

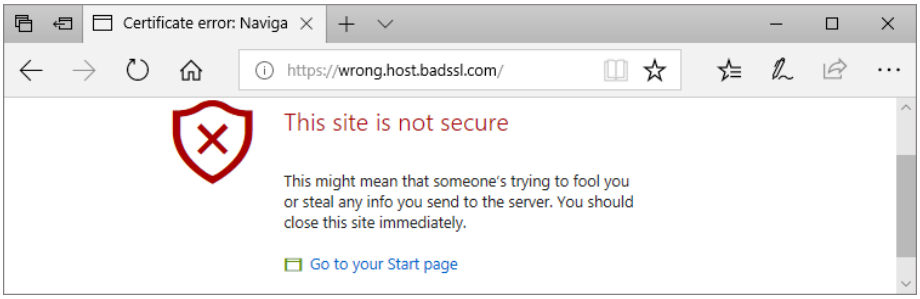
Indicator and warning UI states. Current publications on browser security indicators and warnings distinguish five UI states [8, 15]: HTTP (i.e., no HTTPS), HTTPS (including DV and OV certificates), HTTPS with EV certificate, HTTPS with minor errors, and HTTPS with major errors. In addition, Felt et al. also consider non-TLS related issues, e.g., if a website is blacklisted for hosting malware [15]. While these definitions provide an overview of the different UI states that indicators and warnings can take in concurrent browsers, the question is whether all browsers translate the (technical) TLS and error states equally to the respective UI states.

When looking at a fine-grained overview of (technical) TLS errors, as, for instance, provided on badssl.com [22], there seems to be an agreement on how browsers translate the majority of TLS states and errors to the UI. For instance, standard server-side certificate issues (self-signed, expired, unknown issuer, domain name mismatch) result in major errors and subsequent warnings in Edge 42, Chrome 71, and Firefox 64. However, as of December 2018, for some errors there are differences in how browsers translate them to the UI states. For instance, a revoked certificate (a major error) is displayed with a warning in Edge 42 and Chrome 71 (though without the option to add an exception), but Firefox 64 does not allow the connection and shows a "Secure connection failed" page.

There have been also differences in translating the error states to the UI in the past. For instance, whereas Firefox had a dedicated indicator for HTTPS with minor errors, most other browsers treated this state like HTTP [15].

Indicator position. The position of indicators differed significantly among the different versions of the same browser and among different browsers in the past. For instance, Internet Explorer showed the indicator on the lower right of the browser, within the status bar (IE 5-6), whereas later versions (IE 7-9) showed it on the right side of the URL bar [7]. Similarly, Firefox has seen indicator position changes over time within the primary UI [7]. In contrast, today's major browsers, see Table 1, all show the security indicator on the left side of the URL bar [8].

Visual appearance of indicators. Indicators have mostly used symbols (e.g., padlocks, shields, or triangles), color (e.g., grey, yellow, green, red) and text strings (such as "not secure") to indicate different TLS and error states. For instance, a closed padlock has traditionally signaled in many – and even in early browsers such as Netscape Navigator [38] – a valid SSL/TLS connection (HTTPS): the data is



(a) Warning displayed due to an unmatching host.

(b) Website with an extended validation certificate.

Fig. 1. Display and location of a) website with an error, and b) website with an extended validation certificate in Microsoft Edge 41.

encrypted in transit and the certificate chain validated correctly. Padlocks were used to indicate *HTTPS with major errors* in earlier Chrome versions, signaling the more risky state with red color or a crossed-out padlock [15]. Nowadays, they have been replaced with a red triangle that includes an exclamation mark (Chrome 71). Not all browsers have a dedicated indicator for major TLS errors (additionally to the warning): as of December 2018, the Firefox 64 indicator for *HTTPS with major errors* features a grey circle with an 'i' inside. Besides symbols, some browsers (e.g., IE 7) further used color-highlighted URL bars to distinguish different TLS and error states [21]. The URL bar also often holds the EV certificate state. An example of contemporary indicator placement is shown in Figure 1.

Visual appearance of warnings. In case of *HTTPS with major errors*, browsers show a warning. Initially, as for example in Firefox 2 [41], warnings were presented as a pop-up dialog featuring a heading ("Website Certified by an Unknown Authority") and an additional description. Users received interaction options, for example, accepting the certificate permanently, temporaryly (default), or not connecting to the website. Among other things, issues with this warning were that its appearance was similar to other uncritical dialogs and that it contained technical jargon [41].

Nowadays, the warnings in the five major browsers cover the full content window. All of them contain a heading, explanatory text, different interaction options (e.g.

proceed or go back to previous page), and an image. This image may be a red shield with an exclamation mark (Chrome 70, Edge 42), a red-crossed padlock (Firefox 63), or just a padlock (Safari) [39, 30].

3.4 Examples for UI Evolution

The premise for our work is that indicators and warnings evolved over time and that research, industry, and the practical reality of the Internet influenced each other in this process. In this subsection, we will briefly go over examples for such use cases, which can be further investigated using our proposed data set.

HTTP considered insecure. HTTP works only with the plain-text communication, and is thus insecure by default. As such, researchers suggested indicating this [15]. While in the past many browsers did not feature a dedicated HTTP indicator [15], recent developments induced a shift in how the HTTP connection state is communicated to users. Starting from Chrome 56 and Firefox 51, the so far neutral HTTP indicator got accompanied with either text ("not secure", Chrome) or a symbol (Firefox), signaling an insecure state for HTTP pages which contain password and credit card fields [33, 46]. At the same time, the Chrome team announced future plans to further transit the HTTP state towards being shown as insecure by using the same indicator for this state as for the *HTTPS with major error* state [42]. The changes were motivated by the increase in HTTPS deployment in the Internet and the aim to further promote the use of TLS [33, 46]. They can thus be seen as an example of how the practical reality of the Internet (see also Section 2.3) influences indicator design.

Development of EV certificates. The increase in phishing attacks led browser developers to collaborate on improving phishing defenses, among other things through means of EV certificates [17, 37]. Eventually, the CA/Browser Forum, a standardization body of mainly certificate authorities and browser vendors, was founded [47] and issued later on the EV certificate guidelines in 2007 [9]. In the meantime, the Internet Explorer team was the first to come up with a proposal for indicating the *HTTPS with EV* state in their browser UI [17] and other browser vendors later on followed in designing a dedicated indicator [27]. However, as of 2009, usable security researchers criticized that indicators still differed considerably among browsers [36]. As of December 2018, the visual appearances of the HTTPS with EV state has harmonized: Major browsers show website identity information next to the padlock (by showing the company name and sometimes the company location) for EV certificates.

The introduction of EV certificates is an interesting example that shows how developments in the ecosystem led to observable changes in the user interface and usability improvements of the UI over time. However, some researchers claim that the end-users do not understand the different assurances provided by EV certificates as opposed to OV certificates. This leads to varying adoption of the extended verification since the extra burden may have only marginal benefit [20].

Treatment of mixed content. Mixed content occurs if parts of a web page are loaded via HTTPS while others are still loaded via HTTP. This used to trigger a

warning pop-up dialog in IE 7 and IE 8 [6], necessitating a user interaction. IE 9 then started to automatically block insecure content [23], and changed indicator state by removing the padlock. Firefox traditionally used the padlock plus an additional indicator (a shield) to notify for mixed content. In 2015, they decided to remove the shield and to add additional cues to the padlock (e.g. a crossed padlock or a triangle with exclamation mark) [45]. Chrome used to have an *HTTPS with minor error* indicator for mixed content, but decided in 2015 to use the HTTP indicator for this state instead [18]. As the treatment of mixed content does not seem to be harmonized between browsers, it may be considered as an example of an ongoing indicator evolution in the UI.

4 Browser's Security Indicator Open-Data Set

In the previous section, we provided examples of how research, industry, and the practical reality of the Internet influence the evolution of indicators and warnings. To work towards a better understanding of how browser vendors adopt security technology from each other, and to correlate this with the browser version history and the general time-line of developments in the ecosystem, we need a qualitative data set of web browser UIs for TLS handling. Hence, we propose an open data repository of primary browser UI screen shots, and describe the requirements and challenges of creating such a repository. Thus, the focus of this paper is on the methodological framework.

4.1 Requirements

Security warnings and indicators coverage. To get a complete picture of the developments around browsers' security indicators and warnings, a data set should test for all issues listed in Table 2. Please note that depending on specific research questions, this list might have to be extended or may be confined. For example, for research focusing on the evolution of indicator and warning usability it may be sufficient to look at the indicator and warning UI states, as outlined in Section 3.2. For research focusing on differences between browsers in the translation of TLS and error states to the UI, the full list should be considered.

Certificate Validation	Wrong Host
	Selfsigned
	Revoked
	Expired
	HSTS Violation
	No Common Name
	No Subject
	Untrusted Root
	No Certificate Transparency
	SNI Support
Content and Connection Type	HTTP Only
	HTTP Text Area
	HTTP Password
	HTTP Credit Card (+Autofill)
	Mixed Content
	Mixed Script
	HTTP Favicon
Protocols and Ciphers in Certificates, Chain, and Connection	Key Exchange Algorithms
	Digest
	Ciphers and Cipher Modes
	TLS Versions
	Attack Indicators (e.g. DH small subgroup)

Table 2. Overview of relevant error scenarios, the content issues follow badssl.com [22].

In addition, non-security related work might also benefit from our data set. As such, we strive to keep an open mind towards additional cases that can be added to the data set. This includes cipher choices and TLS versions and various content issues, as well as straight-forward validation issues. Due to the ongoing nature of cipher discontinuation, we suggest to test against all ciphers from all supported SSL/TLS versions of the currently tested browser version (see below). Screen shots will only be recorded for combinations that divert from the happy flow (i.e. a website successfully validating). If an SSL/TLS version is no longer supported by a browser, it is sufficient to test against any non-supported SSL version to record a screen shot of browser behavior in this case.

Furthermore, as data collection is an ongoing process for the future, so far unsupported technology should be tested as soon as it becomes available in the first browser. Examples include DNSSEC validation of the target name zone [3], and TLSA record validation [14]. While both are available via plugins already, the focus of this work is on out-of-the-box UIs.

In addition, other relevant TLS mechanics, like SNI, might be worth evaluating in this context, if the data set is collected anyway. Finally, the happy flow should be captured for DV, OV, and EV certificates, including any warnings that might occur.[4] Depending on the time-line, the happy flow will require a dedicated setup with settings that might be considered insecure at a later point in time, e.g., SSLv3 support. Note that we do not make a distinction between major and minor errors, as this notion changes over time. For example, while a missing Certificate Transparency log entry may be acceptable for certificates issued before 2016, it may be a critical error for a certificate issued by Symantec in 2018 [19].

Comparable visual presentation. Creating screen shots in a manner that they are comparable is difficult. Furthermore, browsers may change their behavior if the window size decreases. For example, Microsoft Edge as in Figure 1 removes the EV indicator if the window becomes too small. In addition, DPI (Dots Per Inch) settings may further influence the visibility or attention towards security indicators. Similarly, OS decorations may distract from indicators. The most common desktop screen resolutions and DPI settings, as well as aspect ratios (4:3/16:10/16:9) underwent changes over time as well.

Hence, we suggest to evaluate the 4 most common combinations of resolution and screen size per year. To cover the potential impact of OS decorations, we suggest to create screen shots of a default-full-screen browser, including decorations and other desktop elements like taskbars etc.

Browser coverage. To build a data set that is as complete as possible, we suggest to evaluate all browsers, starting from 1994, focusing on all browsers that have been among the five most popular for any year since 1995, support SSL/TLS and have a graphical user interface. If a browser is discontinued, e.g., like Netscape Navigator, testing should presume up until the year the latest available version was released.

[4] For example, early versions of Internet Explore displayed a warning when a *non* plaintext website was visited.

Platform coverage. The underlying operating system may influence the appearance of browsers and how security indicators are recognized. We acknowledge that the large number of available platforms is an even more difficult issue than the issue of which browsers to test. To make this project feasible, we suggest the following selection:

- For the Windows platform, the most recent version as well as the most used one for each year.
- For the Linux platform Ubuntu (before then Debian) using the default desktop environment of default installation with a desktop environment. For the distribution we use the most recent stable (Debian) or Long Term Support (Ubuntu) version for the corresponding year.

We acknowledge that evaluating Safari Mac OS X (and Mac OS 9 before then) might yield interesting results. However, due the strong hardware dependence of Apple software, and the strict update regime making earlier versions hardly available (see below), we will leave Apple products out of scope for now.

Time scale. The time scale with which we can evaluate browsers is heavily dependent on the availability of verified browser versions we can test. Hence, our time resolution will follow the software versions to which we can obtain access.

4.2 Challenges

Obtaining correct software versions. Obtaining old software versions is a challenge. There is no consistent repository for outdated browser versions. While there usually are archive mirrors for, e.g., Linux, additional research is necessary to first compile such a software repository.

Creating a reasonable server-side test environment. With the addition of a timescale, we will also have to provide a time-adjusted server-side. A 2018 TLS implementation is usually no longer interoperable with browsers using SSL implementation from the pre-TLS time. As with the client side software selection, this is an issue, especially as we will have to also simulate our own CA for this purpose. As this infrastructure can be created using open-source components, we will be able to utilize archive repositories.

Running outdated systems. In addition to getting access to the right software in terms of operating systems and web browsers, we also have to get access to hardware supported by this software. While virtualization may be feasible for many newer software versions, there will be limitations for Mac OS X and early Windows versions. In addition, the (automatic) update functionality of—especially more modern—software may prevent us from running older software versions.

Limitations, Open Data and Crowdsourcing. Building a *complete* data set certainly exceeds the capabilities of an individual research group. Therefore, we will focus on initially producing a data set that is complete in terms of error types (denied connection, major (with interaction), minor (without interaction), HTTP, HTTPS, HTTPS with EV [15, 8]) for the most commonly used browsers today and ten years ago, see Table 1. We plan to integrate our screen shots in an open data

setup, where researchers (and possibly crowd-sourcers) can assist us in collecting further data to build a complete list.

5 Conclusion

In this paper, we present our methodology for collecting an open data set of web browser UIs related to SSL/TLS over time. Rooted in a literature study and timeline of SSL and TLS development, we outline the basic requirements for such a data set and describe the necessary test cases. Furthermore, we identify several challenges, including the comparability of web browser screen shots due to changing conventions of screen resolution and size combinations, issues in operating outdated software, and hardware dependence of outdated software. We plan to collect the data in an automated and replicable manner, keeping it open for research collaboration.

Acknowledgements

We would like to thank Petr Švenda, Matúš Nemec, Marek Sýs and Adam Janovský for their comments during the paper writing and the participants of the 2019 Security Protocols Workshop for the lively discussion and the useful hints for our research and the paper. Furthermore, we would like to acknowledge the help of Richard Pánek and Filip Gontko with the collection of TLS warning screen shots.

This work has been partly funded by the European Union's Horizon 2020 research and innovation programme under grant agreements No. 830929 (CyberSec4Europe), and No. 825225 (Safe-DEED). The content herein reflects only the authors' view, and not that of the involved funding bodies.

References

1. Aertsen, M., Korczyński, M., Moura, G., Tajalizadehkhoob, S., van den Berg, J.: No domain left behind: Is Let's Encrypt democratizing encryption? In: Proceedings of the Applied Networking Research Workshop. pp. 48–54. ACM (2017)
2. Anderson, R., Baqer, K.: Reconciling multiple objectives – politics or markets? In: Cambridge International Workshop on Security Protocols. pp. 144–156. Springer (2017)
3. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS Security Introduction and Requirements. RFC 4033, IETF (Mar 2005), http://tools.ietf.org/rfc/rfc4033.txt
4. Barnes, R., Thomson, M., Pironti, A., Langley, A.: Deprecating Secure Sockets Layer Version 3.0. RFC 7568, IETF (Jun 2015), http://tools.ietf.org/rfc/rfc7568.txt
5. Borgolte, K., Fiebig, T., Hao, S., Kruegel, C., Vigna, G.: Cloud strife: Mitigating the security risks of domain-validated certificates. In: Proceedings of 2018 Internet Society Symposium on Network and Distributed System Security (NDSS). The Internet Society (2018)
6. BrentgMS: Mixed content and Internet Explorer 8.0 (2009), https://blogs.msdn.microsoft.com/askie/2009/05/14/mixed-content-and-internet-explorer-8-0/
7. Burzstein, E.: Evolution of the https lock icon (infographic) (2011), https://elie.net/blog/security/evolution-of-the-https-lock-icon-infographic
8. CA Security Council: Browser UI security indicators (2017), https://casecurity.org/browser-ui-security-indicators/

9. CA/Browser Forum: Guidelines for the issuance and management of extended validation certificates (2007), https://cabforum.org/wp-content/uploads/EV_Certificate_Guidelines.pdf
10. Delignat-Lavaud, A., Abadi, M., Birrell, A., Mironov, I., Wobber, T., Xie, Y.: Web PKI: Closing the gap between guidelines and practices. In: Proceedings of the 2014 Internet Society Symposium on Network and Distributed System Security (NDSS). The Internet Society (2014)
11. Dierks, T., Allen, C.: The TLS Protocol Version 1.0. RFC 2246, IETF (Jan 1999), http://tools.ietf.org/rfc/rfc2246.txt
12. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, IETF (Apr 2006), http://tools.ietf.org/rfc/rfc4346.txt
13. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, IETF (Aug 2008), http://tools.ietf.org/rfc/rfc5246.txt
14. Dukhovni, V., Hardaker, W.: The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance. RFC 7671, IETF (Oct 2015), http://tools.ietf.org/rfc/rfc7671.txt
15. Felt, A.P., Reeder, R.W., Ainslie, A., Harris, H., Walker, M., Thompson, C., Acer, M.E., Morant, E., Consolvo, S.: Rethinking connection security indicators. In: Proceedings of the 2016 Symposium On Usable Privacy and Security (SOUPS). pp. 1–14. USENIX Association (2016)
16. Fiebig, T., Lichtblau, F., Streibelt, F., Krüger, T., Lexis, P., Bush, R., Feldmann, A.: Learning from the past: Designing secure network protocols. In: Cybersecurity Best Practices, pp. 585–613. Springer (2018)
17. Franco, R.: Better website identification and extended validation certificates in IE7 and other browsers (2005), https://blogs.msdn.microsoft.com/ie/2005/11/21/better-website-identification-and-extended-validation-certificates-in-ie7-and-other-browsers/
18. Garron, L., Palmer, C.: Simplifying the page security icon in Chrome (2015), https://security.googleblog.com/2015/10/simplifying-page-security-icon-in-chrome.html
19. Gustafsson, J., Overier, G., Arlitt, M., Carlsson, N.: A first look at the CT landscape: Certificate transparency logs in practice. In: Proceedings of the 2017 International Conference on Passive and Active Network Measurement. pp. 87–99. Springer (2017)
20. Hunt, T.: Extended validation certificates are dead (2018), https://www.troyhunt.com/extended-validation-certificates-are-dead/
21. Jackson, C., Simon, D.R., Tan, D.S., Barth, A.: An evaluation of extended validation and picture-in-picture phishing attacks. In: International Conference on Financial Cryptography and Data Security. pp. 281–293. Springer (2007)
22. King, A., Garron, L., Thompson, C.: Memorable site for testing clients against bad SSL configs (2018), https://badssl.com
23. Lawrence, E.: Mixed content and Internet Explorer 8.0 (2011), https://blogs.msdn.microsoft.com/ie/2011/06/23/internet-explorer-9-security-part-4-protecting-consumers-from-malicious-mixed-content/
24. Manousis, A., Ragsdale, R., Draffin, B., Agrawal, A., Sekar, V.: Shedding light on the adoption of Let's Encrypt. Computing Research Repository **abs/1611.00469** (2016), http://arxiv.org/abs/1611.00469
25. Mockapetris, P.: Domain names - concepts and facilities. RFC 1034, IETF (Nov 1987), http://tools.ietf.org/rfc/rfc1034.txt
26. Naughton, John: Netscape: The web browser that came back to haunt microsoft (2015), https://www.theguardian.com/global/2015/mar/22/web-browser-came-back-haunt-microsoft

27. Nightingale, J.: Will Firefox have a green bar? (2007), http://blog.johnath.com/2007/06/04/will-firefox-have-a-green-bar/
28. Orgera, Scott: The history of Mozilla's Firefox web browser (2018), https://www.lifewire.com/the-history-of-firefox-446233
29. PCI Security standards council: Payment card industry data security standards. Tech. rep. (2018), v3.2.1
30. Reeder, R.W., Felt, A.P., Consolvo, S., Malkin, N., Thompson, C., Egelman, S.: An experience sampling study of user reactions to browser warnings in the field. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. p. 512. ACM (2018)
31. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, IETF (Aug 2018), http://tools.ietf.org/rfc/rfc8446.txt
32. Roessler, T., Saldhana, A.: Web security context: User interface guidelines. W3C recommendation, W3C (2010), https://www.w3.org/TR/wsc-ui/
33. Schechter, E.: Moving towards a more secure web (2016), https://security.googleblog.com/2016/09/moving-towards-more-secure-web.html
34. Sheffer, Y., Holz, R., Saint-Andre, P.: Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 7525, IETF (May 2015), http://tools.ietf.org/rfc/rfc7525.txt
35. Sheffer, Y., Holz, R., Saint-Andre, P.: Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS). RFC 7457, IETF (Feb 2015), http://tools.ietf.org/rfc/rfc7457.txt
36. Sobey, J., Van Oorschot, P.C., Patrick, A.S.: Browser interfaces and EV-SSL certificates: Confusion, inconsistencies and HCI challenges. Tech. Rep. TR-09-02, Carleton University School of Computer Science, Canada (2009)
37. Staikos, G.: Web browser developers work together on security (2005), https://dot.kde.org/2005/11/22/web-browser-developers-work-together-security
38. Stallings, W.: SSL: Foundation for web security. The Internet Protocol Journal **1**(1), 20–29 (1998)
39. Stark, E., Sleevi, R., Muminović, R., O'Brien, D., Messeri, E., Felt, A.P., McMillion, B., Tabriz, P.: Does certificate transparency break the web? Measuring adoption and error rate. In: Proceedings of the 2019 IEEE Symposium on Security and Privacy (S&P) (to appear) (2019)
40. Statcounter GlobalStats: Browser market share worldwide (2018), http://gs.statcounter.com/browser-market-share/desktop/worldwide
41. Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., Cranor, L.F.: Crying wolf: An empirical study of SSL warning effectiveness. In: Proceedings of the 2009 USENIX Security Symposium. pp. 399–416. USENIX Association (2009)
42. The Chromium projects: Marking HTTP as non-secure (2016), https://www.chromium.org/Home/chromium-security/marking-http-as-non-secure
43. Thomas, S.A.: SSL and TLS Essentials: Securing the Web. John Wiley & Sons, Inc., New York, NY, USA (2000)
44. Turner, S., Polk, T.: Prohibiting Secure Sockets Layer (SSL) Version 2.0. RFC 6176, IETF (Mar 2011), http://tools.ietf.org/rfc/rfc6176.txt
45. Vyas, T.: Updated Firefox security indicators (2015), https://blog.mozilla.org/security/2015/11/03/updated-firefox-security-indicators-2/
46. Vyas, T., Dolanjski, P.: Communicating the dangers of non-secure HTTP (2017), https://blog.mozilla.org/security/2017/01/20/communicating-the-dangers-of-non-secure-http/

47. Yiu, K.: Improving SSL: Extended validation (EV) SSL certificates coming in January (2006), https://blogs.msdn.microsoft.com/ie/2006/11/07/improving-ssl-extended-validation-ev-ssl-certificates-coming-in-january/