Here's My Cert, So Trust Me, Maybe? Understanding TLS Errors on the Web

Devdatta Akhawe
University of California, Berkeley
devdatta@cs.berkeley.edu

Johanna Amann
International Computer Science Institute, Berkeley
johanna@icir.org

Matthias Vallentin
University of California, Berkeley
vallentin@cs.berkeley.edu

Robin Sommer
International Computer Science Institute, Berkeley
robin@icir.org

ABSTRACT

When browsers report TLS errors, they cannot distinguish between attacks and harmless server misconfigurations; hence they leave it to the user to decide whether continuing is safe. However, actual attacks remain rare. As a result, users quickly become used to "false positives" that deplete their attention span, making it unlikely that they will pay sufficient scrutiny when a real attack comes along. Consequently, browser vendors should aim to minimize the number of low-risk warnings they report. To guide that process, we perform a large-scale measurement study of common TLS warnings. Using a set of passive network monitors located at different sites, we identify the prevalence of warnings for a total population of about 300,000 users over a nine-month period. We identify low-risk scenarios that consume a large chunk of the user attention budget and make concrete recommendations to browser vendors that will help maintain user attention in high-risk situations. We study the impact on end users with a data set much larger in scale than the data sets used in previous TLS measurement studies. A key novelty of our approach involves the use of internal browser code instead of generic TLS libraries for analysis, providing more accurate and representative results.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public Key Cryptosystems

Keywords

TLS; Warnings; Usability;

1. INTRODUCTION

The Transport Layer Security (TLS) protocol provides secure channels between browsers and web servers, making it fundamental to user security and privacy on the web. As a critical step, TLS[1] enables clients to verify a server's identity by validating its public-key certificate against a set of trusted root authorities. If that validation fails browsers cannot distinguish between actual attacks and benign errors (such as a server misconfiguration). Instead, browsers display a warning and ask the user to decide whether continuing is safe.

Unfortunately, multiple studies have shown the proclivity of users to click through browser warnings, thus negating any security provided by TLS [22, 43, 44]. One reason is the prevalence of warnings even under benign scenarios. User attention is finite and excessive warnings tire users out, leading to "click-whirr" responses that just dismiss all warnings without due thought [19]. Improvements in warning design do not help here as they provide benefit only if users pay attention *to each and every warning* [44].

Unfortunately, user attention is also *shared*: any website that triggers a TLS warning depletes the pool. For example, even if `bank.com` is cautious in its TLS deployment, warnings due to `incompetentsite.com` can still tire users and train them to click through all warnings. It remains a tragedy of the commons that no *individual* web site has sufficient incentive to conserve user attention.

We study TLS errors that trigger browser warnings. Based on our study, we give recommendations on improving the validation logic to treat the shared attention pool more respectfully. For our discussion, we use the broad term *false warning* to refer to any TLS-related dialog box raised during the lifetime of connection unless it represents an indicator of an actual man-in-the-middle (MITM) attack. This includes errors due to server misconfiguration (e.g., incomplete certificate chains) as well as self-signed certificates and name validation errors. We discuss the errors we study in more detail in Section 5.2.

We base our study on an extensive data set containing TLS activity of more than 300,000 users, collected passively at egress points of ten operational network sites over a nine-month period. Different from prior TLS analyses relying on active scans (e.g., [12, 46]), our data set reflects what

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

[1]For ease of exposition, we use the term TLS to refer only to the TLS protocol as used on the web, and not the TLS protocol in its full generality.

users actually perceive as they browse the web, including accounting for the heavy-tailed nature of website popularity. Compared to a previous study of a passive collection [18], our work encompasses a notably larger user population and focuses on the user's perspective.

Our results indicate a clear opportunity, and *need*, for reducing false warnings. When examining all 3.9 billion TLS connections in our data set, we find that 1.54% of them trigger a false warning, an unacceptably high number since benign scenarios are orders of magnitude more common than attacks. In Section 6, we give concrete suggestions to reduce the number of end-user warnings, based on an extensive analysis of the cases triggering them.

A novelty of our work concerns the use of browser logic for certificate validation, including all its quirks and optimizations. TLS implementations in browsers have evolved along with specifications and exploits over the past fifteen years. We noticed significant differences between results returned by browser code compared to the results of using the OpenSSL libraries relied on by previous work [18,46]: we were able to validate 95.97% of the chains in our database, as opposed to 88.16% with OpenSSL. Section 4 further discusses the subtleties of certificate validation in browsers.

In summary, we make the following contributions:

- We discuss how browsers validate TLS certificates and highlight the importance of relying on browser code for such measurement studies. We identify scenarios where the browser libraries differ from generic libraries like OpenSSL (Section 4).
- Using passive network measurement on a population of about 300,000 users over a nine-month period, we measure the frequency and nature of TLS errors on the web (Section 5).
- Based on our measurement, we make concrete recommendations to browser vendors to reduce the amount of false warnings (Section 6).

Our analysis code is available online [1]. During our research, we also identified and reported two security and privacy bugs in Firefox's implementation of TLS [32,33].

We structure the rest of the paper as follows: after discussing related work in Section 2, Section 3 revisits TLS and introduces the terminology we use. Section 4 discusses how browsers validate certificates along with scenarios in which their behavior diverges from libraries like OpenSSL. Section 5 explains our measurement infrastructure and the errors we study. Finally, we discuss the results and provide recommendations in Section 6 before concluding in Section 7.

2. RELATED WORK

Warnings Science. Krosnick and Alwin's dual path model represents a standard model to understand human behavior when faced with questions [20]. According to this model, humans rely upon two distinct strategies when presented with a question. Users who *optimize* read and interpret the question, retrieve relevant information from long-term memory, and make a decision based on their disposition and beliefs. In contrast, users who *satisfice* do not fully understand the question, retrieve only salient cues from short-term memory, and rely on simple heuristics to make a decision.

Grosklags et al. extend this model to argue that security interface designers should treat user attention as a finite resource, a *lump of attention* shared by all [4]. Consuming user attention should happen infrequently, and uncontrolled consumption can lead to a tragedy of the commons scenario. Repetitive warnings cause habituation: once users click through the same warning often enough, they will switch from reading the warning to quickly clicking through without reading (i.e., satisficing). Users pay less attention to each subsequent warning, and this effect carries over to other, similar warnings [4]. The link between repetition and satisficing has been observed for Windows UAC dialogs, Android install-time warnings, and browser security warnings [11,14,28,44].

Usability of Security Warnings. In 2005, an eye tracker based study by Whalen et al. found that users rarely pay attention to "passive" security indicators such as the lock icon in web browsers [48]. Dhamija et al. challenged users to recognize phishing attacks and found that 23% of their subjects ignored phishing warnings while 68% ignored the active, interstitial TLS warnings [8]. Similarly, Sunshine et al. found users clicked through active warnings [44]. Sunshine et al. proposed an improved warning design based on existing warning design literature, and observed noticeably lower click-through rates with their new design. Nevertheless, a large number of users still clicked through the new warnings, and the authors suggested research on reducing or eliminating browser warnings. Our work constitutes a step in that direction.

Sotirakopoulos et al. replicated the Sunshine study in 2010 and found that users learned how to bypass the new warning design [43]. Further, they noted that user studies comparing new warnings with old warnings exhibit a bias towards users not accustomed to the new warnings. Their results indicate that habituation, and not warning design, is the main cause of high click-through rates in TLS warnings.

TLS Measurements. The past years witnessed a noticeable increase of security incidents involving certificate authorities, rendering the global certificate infrastructure an attractive subject of study. The Electronic Frontier Foundation (EFF) popularized the study of SSL infrastructure by publishing certificate data sets obtained from actively scanning the entire IPv4 address space on port 443 in mid-2010 and in early 2012 [12], yielding 5.5 million [18] and 7.2 million [25] distinct certificates each. As we discuss in Section 3, servers can rely on the SNI extension to dispatch the correct certificate for a given domain. IP address scans miss these certificates. They also cannot reliably determine if the names in the certificate match the DNS names users use to access the site.

Holz et al. [18] provide a comprehensive measurement of the TLS infrastructure. In addition to incorporating the 2010 EFF data set, the authors crawl the top one million Alexa domains from different vantage points, as well as gather certificate and connection data via passive network monitoring of a German research network. They find that for their active scan sets about 60% of the chains validate. Vratonjic et al. [46] also crawl the top one million Alexa sites and extract 300,000 certificates (including 48% duplicates), of which only 16% validate.

Our work differs in scale and goals. We characterize TLS behavior as seen for 300,000 users, *on the wire*. Studies relying on the Alexa lists suffer from the biases of the Alexa dataset [49]. Active scans and measurements based on rankings give equal weight to all websites, which undermines their accuracy due to the long-tailed nature of traffic on the web.

A key novelty of our work involves using the same code that browsers use for validating certificates. Two examples illustrate the importance of recreating the browser validation logic correctly. First, previous work used the OpenSSL libraries, which do not cache previously seen intermediates. This may result in a higher number of chain errors than that experienced by end users. Second, Holz et al. focused only on chain validation, and ignored name comparison for their passive measurement. Validating the name in the certificate with the DNS label that the user wanted to connect to is a critical component of the validation process, and the only protection TLS offers against a MITM attack. We explore these subtleties further in Section 4.4.

Evolving TLS Trust Mechanisms. Recent work has also explored alternatives to the CA trust system. Perspectives [47] offers a SSH-like trust model to the existing PKI infrastructure. Perspectives and its follow-up Convergence [6] both rely on network views for trusting a certificate. Our study makes similar assumptions, and implicitly trusts all certificates seen on the network backbone.

Other proposals for alternate trust models include the Certificate Transparency project [23], the Country-based trust model [42], and the Sovereign Keys proposal [45]. While novel trust models hold the promise of reducing false warnings, they are still nascent and widespread adoption will take time. Our work focuses on the TLS landscape of today, but we hope our measurements can influence the development of these proposals.

3. BACKGROUND

In this section we give an overview of the TLS ecosystem, define terminology that we will use for the rest of the paper, and discuss the state of TLS support in browsers and web servers. We also discuss the need for warnings in browsers and the status quo of browser TLS warnings.

3.1 The TLS Ecosystem

TLS aims at providing end-to-end encryption for communication over the Internet. It uses public-key cryptography for the initial key exchange, followed by symmetric encryption for the rest of the protocol. The initial public-key handshake requires the client to authenticate the server's public key through an out-of-band mechanism. After authenticating the server, client and server negotiate a cipher suite and a symmetric key to use for future communication.

The SSL protocol (TLS's predecessor) originated as a mechanism for encrypted communication on the web. Despite its design as a generic transport layer mechanism, the web remains its main user. Clients are usually browsers, and servers typically web servers such as Apache `httpd` and Microsoft IIS. Server authentication occurs via a *certificate* signed by a trusted *certification authority.* Below we go into the details of TLS as used in the web. As a running example, we consider the case of a user Alice that wants to connect to `bob.com` via TLS. `bob.com` relies on the `Honest` certification authority to authenticate to Alice.

Concretely, when Alice visits `https://www.bob.com`, her browser connects to `www.bob.com` over port 443 where the server responds with a certificate. Then, the browser validates that a trusted certification authority signed it, and checks that it pertains to `www.bob.com` and no other host. If these checks succeed, the browser uses the public key in the certificate to setup a secure channel with `www.bob.com`.

3.1.1 Certification Authorities

Server authentication is critical to TLS' guarantees against an active network attacker. Certification authorities, or certificate authorities, or simply CAs, provide this functionality on the web. At the minimum, CAs authenticate a server's public-key by signing a *certificate* tying a particular public key to one or more DNS labels. For example, in the certificate presented by `www.bob.com`, the `Honest` authority certifies Bob's public-key as belonging to `www.bob.com`. The user agents (browsers) ship with a set of "trusted" CAs called the *root store.* Server authentication proceeds only if an authority in the root store signed the presented certificate.

Intermediate Authorities. A CA in the trusted root store can also create and sign certificates for other *intermediate* CAs; the trust relationship is transitive. For example, Bob can present a certificate signed by Carol, and Carol one signed by Honest. The browser trusts Bob's certificate if Honest is part of the root store. This Bob$\rightarrow$Carol$\rightarrow$Honest path forms a certificate *chain.* The intermediate certificate can be part of the certificate sent by the web server, or the web server (Bob) can provide a URL for an intermediate (Carol) via the Authority Information Access (AIA) certificate extension. This reduces bandwidth use since the browser can cache common intermediates. It also makes the TLS server configuration easier, because the server operator does not have to worry about providing a correct certificate chain from the intermediate and up.

Authenticating Servers. A critical assumption of TLS lies in the correctness of certificates issued by an authority. In particular, `Honest` must only issue a certificate for `www.bob.com` after validating that the public-key in question actually belongs to the owner of `www.bob.com`. Typically, CAs rely on email to the domain to verify ownership. Since email is not a secure mechanism, an attacker on the path between the authority and the owner of the website (`bob.com`) can easily receive a certificate for the same (`bob.com`). Extended validation (EV) certificates, indicated in the browser URL bar by a distinctive green color and the name of the entity owning the certificate, involve further checks, such as physical presence requirements. It is not clear users understand the difference between regular domain validated (DV) and EV certificates. An attacker with a DV certificate can present it in place of the legitimate domain's EV certificate, and the web browser will accept it. Only the missing green bar in the browser interface would indicate the absence of an EV certificate.

3.1.2 Web Servers

Web servers ask the CA for a certificate tying the public key to their DNS label. When Alice connects to the web server, the server presents this certificate, and Alice then continues with the rest of the session.

Name-based virtual hosting complicates this simple scenario. For example, consider the case of a web server that hosts both `bob.com` and `example.com`. When the server receives a connection from Alice's browser, it needs to know which domain Alice wants to connect to so that it can present the correct certificate. The Server Name Indication (SNI) extension [10] allows a client to indicate the intended domain, allowing the server to return the appropriate certificate. Apache's `httpd` supports SNI since 2009, while Microsoft's IIS added support in 2012. As we discuss in Section 5.1, we rely on SNI support in browsers (not servers) for our study.

Browsers have widespread support for SNI, as we discuss below.

3.1.3 *Browsers*

As the most common client-facing component of TLS, web browsers exert a tremendous influence on the evolution of TLS. By controlling the root certificate store, browsers determine the list of CAs that the web relies on to provide security. Furthermore, by supporting specific cipher suites and TLS extensions browsers influence their widespread adoption.

As of this writing, all desktop browsers support TLS 1.0, only Google Chrome supports TLS 1.1, and no desktop browser supports TLS 1.2 by default. Further, all modern desktop browsers support SNI. For mobile devices, Android supports SNI since version 3.0 and iOS supports SNI since version 4.0. Current versions of all major browsers, except for Mozilla Firefox, support the AIA extension. All major browsers also cache any valid intermediate certificates seen in the past.[2]

3.2 TLS Warnings

Relevant to our work is the behavior of browsers in case of a failure during server authentication. In addition to a man-in-the-middle (MITM) attack, authentication failures also occur in a wide variety of benign scenarios (Section 5.2). It is difficult for browsers to distinguish malicious scenarios from benign ones.[3] Instead, browsers present users with a warning page informing them of the error, and warning of a possible MITM attack, *for all authentication errors*. Browsers also allow users to bypass the warning, and continue with their session; in effect, forcing the users to distinguish a benign scenario (false warning) from a malicious one—a distinction most users cannot make.

Warnings raised due to TLS errors belong to two broad categories. First, there exist interstitial warnings shown when the top-level page contains an error. This has been the focus of previous warnings research in browsers. Another class of warnings occur when secondary resources, e.g., images, scripts, etc., result in a TLS error. Browsers currently show a mixed content warning when secondary resources on a page fail to load due to a TLS error.

Recall the Lump of Attention model (Section 2). Our study assumes that mixed content and top-level page warnings both consume user attention. Over-consumption of user attention is an externality on the security of the whole TLS ecosystem. Understanding the reasons for TLS warnings provides us with insight into the current state of TLS, as well as concrete guidance on areas to focus on for improvement. Reducing the number of false warnings raised by browsers is key to improving assurance in the TLS ecosystem. Rare warnings encourage optimizing behavior over satisficing and push users to understanding the warnings before making an informed decision. This work focuses on understanding the prevalence of TLS errors to enable reducing the number of warnings, not on improved warning design. Previous work on warning usability also stressed the importance of reducing the number of warnings [43, 44].

[2]During our research, we discovered that this caching occurs even in Firefox's private browsing mode (Bug 808331 [33]).

[3]In the case of websites with pinned certificates [5] or HSTS support [16], browsers can make this distinction and show an error instead of a warning.

4. BROWSER VALIDATION BEHAVIOR

Like most complex web standards, certificate validation logic in browsers evolved simultaneously with and ahead of specifications like X.509 [7, 41]. Browsers need to support a wide variety of erroneous behaviors, and need to balance security, usability, standards, and backwards compatibility. In cases of underspecified behavior, browsers can behave differently from standard system libraries like OpenSSL, GnuTLS, and SChannel. In some cases, browsers intentionally deviate from the specification for backwards compatibility.

In this section, we go into the details of how the Network Security Services (NSS) [36] library, used by Firefox (and Chrome on Linux), validates a certificate. We also discuss the subtleties of reproducing this behavior for our analysis. For ease of exposition, we break down certificate validation into three separate steps: chain building (Section 4.1), chain validation (Section 4.2), and name validation (Section 4.3). In reality, these steps do not execute sequentially and often intertwine. We end the section with a comparison of the OpenSSL library (used in previous work) and NSS (the library we rely on).

4.1 Chain building

After receiving a certificate from the web server, the browser needs to generate an appropriate permutation of certificates that chains up to a trusted certificate. The set of trusted certificates in the root store changes over time, and our measurement infrastructure needs to ensure that it uses the correct root store based on the timestamp of the connection.

Ideally, the web server correctly presents the whole chain, including any intermediates, in the reply sent to the browser. In practice, this is often not the case: servers may only send the end-host certificate, not include any chain, present an incomplete chain, include additional unneeded certificates, contain duplicate certificates, or have the wrong order.

To the best of our knowledge, all major browsers cache a valid intermediate certificate seen in a connection, and reuse it to validate connections in the future. Browsers try to build a valid certificate chain using the information the server sent as well as any other intermediate certificates in cache. Such caching implies that the browser's ability to validate a specific certificate depends on the current state of its certificate store. A user visiting a website which does not supply all necessary certificates will see a warning if the browser has not seen the missing intermediate in the past.

This is a widespread problem. In our data, 8.13% of the valid chains exhibited this behavior. Instead of including the required intermediate as part of the certificate chain, websites can include a URI in the AIA field that points to the intermediate. Browsers either reuse existing intermediates, or download them from the URI listed. Unfortunately, Firefox lacks AIA support, and Chrome offers preliminary support.[4] Internet Explorer supports the AIA extension, and Microsoft properties commonly rely on it.

Chain building can also face the opposite problem: extra certificates give the library multiple paths to choose. Browsers differ in their behavior for such a scenario. Firefox chooses one path, and raises a warning if the chosen path does not work. While trying to follow a single path to its end,

[4]For example, Chrome does not use HTTP proxy settings for AIA fetches.

Firefox can get stuck in a loop [30]. Chrome tries multiple paths, and uses any path that succeeds.[5]

4.2 Chain Validation

At the end of chain building, the browser has found a permutation of certificates, starting from that of the website and ending at a trusted certificate in the root store. The browser then proceeds to check this chain for expiration, revocation, and name/length constraints.

Expiration. Certificates are valid for fixed periods. The `Not Before` and `Not After` fields of the certificate encode this information. For each certificate in the chain, the browser verifies that the current date falls in between the period defined in the certificate. Since we run our analysis offline after the connection took place, it is critical that we use the appropriate time when validating a certificate chain.

Revocation. Malicious actors can steal private keys, or exploit vulnerabilities in certificate issuance systems to get certificates for DNS labels that they do not control. Worse, attackers can also issue themselves an intermediate certificate, allowing them to MITM arbitrary traffic. Certificate revocation lists (CRLs) [7] and the OCSP service [41] provide a mechanism to check the validity of an intermediate or leaf certificate. The browser needs to check every certificate in the chain for revocation. For extreme cases, like the recent DigiNotar incident [24], browsers hardcode a list of untrusted certificates [35]. A notable exception here applies to Chrome, which relies on its own distribution mechanism for certificate revocation information [21] and does not make any CRL or OCSP requests.

Name and Path Length Constraints. Certificate Authorities can place usage limits on the intermediate certificates they issue. The path length constraint limits the number of intermediate certificates below the issued certificate. A number of zero means that an intermediate cannot issue any intermediate certificates. Similarly, name constraints can limit the intermediate to only issue certificates for certain sub-domains. For example, an intermediate constrained to `*.example.com` cannot issue certificates for `*.bank.com`. All modern browsers support name and path constraints, but since authorities rarely employ it, browser support remains relatively untested.

4.3 Name Validation

After a browser has built a certificate chain it deems valid, it checks that the hostname the user tried to connect to matches a name in the certificate. Note that this is the only defense against MITM attacks, and as such represents a critical step of the certificate validation process. An attacker can always get a perfectly valid chain for his own `attacker.com` domain, and present it when the user tries to connect to `bank.com`. Only the name verification check prevents a successful man-in-the-middle attack.

The *Common Name* (CN) field in the certificate subject contains the domain name(s) for which it is valid. For ease of administration, common names can contain wildcards to cover all sub-domains (e.g., a certificate for `*.paypal.com`).

NSS' name validation function restricts the occurrence of asterisks in the certificate subject to the initial part of a domain name. Further, the asterisk only matches one level of names. For example, `*.example.com` cannot match `one.two.example.com`.

The *Subject Alternative Name* (SAN) field enables a certificate to include a list of names, instead of just one. This allows an owner of multiple domains to share the TLS infrastructure. For example, YouTube serves a certificate valid for (amongst others) `*.google.com` as well as `*.youtube.com`. In our data, we have certificates listing up to 545 different names in this field.

RFC 2818 [39] and RFC 6125 [40] describe how HTTP uses TLS and specifies that clients should only consider the last (most specific) common name field in a certificate subject. It also requires clients to ignore the common names if the SAN extension is present.

4.4 NSS vs. OpenSSL

A novelty of our work concerns the use of TLS libraries to emulate browser behavior. Specifically, we opt to use NSS instead of OpenSSL to validate certificate chains. While OpenSSL serves as an all-purpose low-level cryptography library for easy and deep access to cryptographic routines, NSS provides a higher level TLS abstraction in the browser context. Firefox and Google Chrome rely on the NSS libraries for SSL/TLS.

For example, the chain-building algorithm of OpenSSL is strict and can reject chains with superfluous certificates. In contrast, NSS maintains its own database to keep track of any valid intermediate certificates encountered in the past, self-signed site certificates added by the user, and cases where a user permanently overrides a security error. In addition, NSS contains a hard-coded list of root certificates. During certificate validation, NSS tries to use all available certificates to build a valid chain. NSS' chain resolution algorithm is lenient and can accept chains rejected by OpenSSL.

This divergence resembles the divergence in HTML parsing in browsers and parsing libraries. For backwards compatibility and usability reasons, browsers *need* to be more forgiving of HTML syntax than most libraries. Similarly, the NSS library *needs* to be more forgiving than a library like OpenSSL.

The difference in chain validation directly affects the results of any measurement study. Using NSS, we were able to validate 95.97% of all the different chains we saw. In contrast, OpenSSL was only able to validate 88.16% of the unmodified chains.

Furthermore, OpenSSL does not offer any built-in functionality to check if a certificate validates for a given hostname. Checking the hostname against the certificate is the only defense against MITM attacks. Previous work either skipped name validation [18] or wrote a custom validation function [46]. Validating a host name to a certificate is non-trivial. Recently, Georgiev et al. found a number of critical vulnerabilities in certificate validation due to developers having to implement their own code for host name validation, instead of relying on the library [15]. Fahl et al. presented similar findings in Android applications [13].

5. METHODOLOGY

Having discussed the modus operandi of browser validation, we now study the prevalence of TLS errors on the web. We briefly explain our passive monitoring infrastructure in Section 5.1 and refer to Amann et al. for further details [2]. Section 5.2 presents a categorization of the errors we study. Finally, we present and discuss our results in Section 6.

[5]A flag in NSS allows the choice of either Chrome or Firefox behavior.

5.1 Measurement Infrastructure

The ICSI networking group collects TLS/SSL session and certificate information from ten research, university, and government networks. These networks represent a user base of about 300,000 users. This data collection began in early 2012 and the number of contributing sites has steadily increased. Overarching goals of this data collection effort include enabling empirical research of the TLS ecosystem as well as helping to understand its evolution and design. Amann et al. provide a full introduction to the data collection infrastructure [2].

Infrastructure. Each of our data providers operate an internal network from which TLS connections originate to the Internet. As the traffic crosses the network border, the Bro network monitor [37] inspects the traffic for policy violations and intrusions.

Bro's dynamic protocol detection identifies SSL/TLS traffic independent of the transport-layer port [9]. At each site, we provide the operators with a script that, for each TLS/SSL connection, logs the SNI extension header value (if available), the complete server certificate chain, and the timestamp of the connection. Due to privacy concerns, our script does not record any information that identifies a client system directly. Every hour, the script uploads the ASCII-formatted log-files to a storage machine at ICSI. Some sites process nearly two million SSL/TLS connections at peak hours.

Data. Our data covers a period of nine months, ranging from February to November 2012. We successfully captured a total of 11.5 billion SSL connections on all ports. Of these, 10.2 billion connections connect to port 443 (the default HTTPS port). We further filter these connections by removing connections due to grid computing, connections that resumed a previous session and did not exchange any certificate, and connections that did not have the SNI field set. This leaves us with 3.9 billion connections that exchange at least one certificate and exhibit a SNI value. The total number of distinct SNIs in our final data is 9.8 million. The total number of distinct certificates is 496,742.

Reproducing Browser Validation Logic. Using NSS outside the browser context poses a number of challenges. Because the developers of NSS geared the library towards browsers, it lacks convenient APIs for standalone usage. NSS does not come with extensive documentation and example code; requiring aspiring users to dig deep into the code base. Furthermore, NSS is aimed at browsers and lacks some of the functionality needed for large-scale analysis. We developed wrapper libraries to facilitate certificate validation with NSS, OpenSSL (for comparison), as well as a NSS patch which allows verification of large numbers of certificate chains. Our code is freely available online under an open source license [1]. We used NSS 3.13.6 with the aforementioned patch to generate all numbers.

5.2 Categorizing TLS Errors in Browsers

In this study, we focus on a subset of errors that cause an *overridable* warning to appear, i.e., a warning that allows users to continue despite a TLS error. In Firefox, users can only override errors related to certificate validation. Similar to the structure in Section 4, we classify these errors based on where they occur, namely, during chain building, chain validation, and name validation.

Recall that the three phases discussed above are only for exposition, and do not correspond to any modularization in

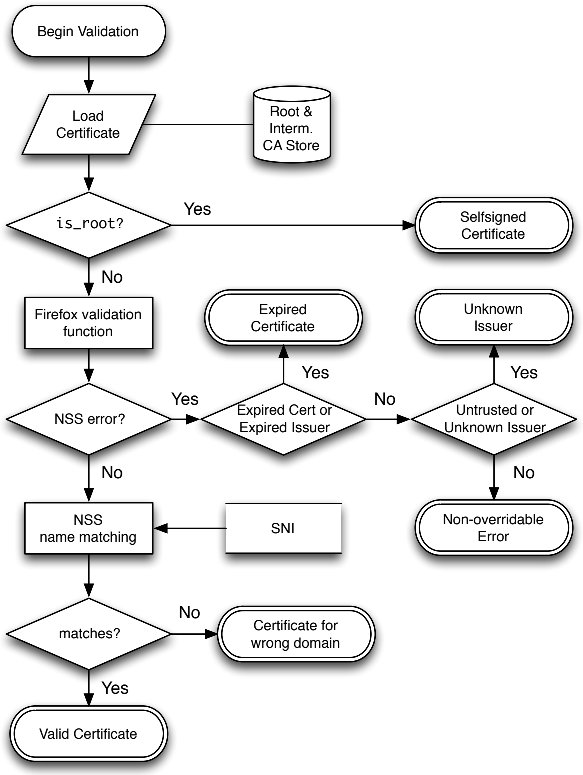

Figure 1: Certificate validation error flowchart. The root and intermediate store contains all the valid intermediates in our full dataset.

the NSS code. As a result, our classification of errors also does not directly correspond to NSS error codes. Figure 1 presents the algorithm we use for translating the NSS responses into our categorization. We expand on this further below.

5.2.1 Chain Building Errors

An error during chain building occurs if the browser cannot find a permutation of certificates that links a certificate for the website to a trusted root.

1a. Unknown Issuer. This error occurs when the browser is unable to create a chain from the server certificate to a trusted certificate authority. This typically happens when the server certificate issuer is neither present in the NSS root store, nor is it one of the valid intermediates we saw throughout our experiment.

1b. Self-signed Certificate. A self-signed certificate is a certificate with an unknown issuer whose subject and issuer are the same. We created a separate category for this error due to its prevalence. Personal use devices, such as routers, music players, and disk drives, commonly use self-signed certificates. We use the NSS `is_root` flag to classify certificates as self-signed. This is the same flag used by the Firefox browser to identify self-signed certificates for its warning page.

1c. Incomplete Chain. In its response, a website can either include all the intermediates it needs to chain back to a trusted root, or include an AIA field pointing to the requisite intermediate certificates, or just hope that the missing intermediates are present in the user's cache. Among the major browsers, only Firefox does not support the AIA extension and may fail to build a working chain in the absence of a needed intermediate certificate. Measuring the prevalence of this error is subtle since Firefox caches all valid intermediates it has seen in the past, even across browser restarts. We bound this error by testing all unique chains with caching turned on and caching turned off. The exact number of errors experienced by each user depends on the individual user's browsing history. Since our data does not allow us to identify individual users, exact measurement is impossible.

5.2.2 Chain Validation Errors

These errors map directly to the individual phases discussed in Section 4.2.

2a. Expired Certificates. Recall that a certificate is only valid for a specific period. If a server sends a certificate that is no longer valid, but was valid in the past, NSS classifies it as an expired certificate. While most servers renew their certificates before they expire, we also find several that fail to renew before the expiration date.

2b. Revoked Certificates. In addition to security reasons, CAs use revocation for administrative reasons. Chrome maintains its own revocation list, free of revocations caused by administrative reasons [21]. In contrast, Firefox and Internet Explorer check CRLs and use OCSP requests to check the status of certificates. Our measurements allow us to bound the impact of revocation, and individual design decisions (e.g., Chrome's decision) on user visible warnings.

5.2.3 Name Validation Errors

A name validation error occurs when a name is not found in the certificate matching the domain that the user connected to. Figure 2 illustrates our classification algorithm that further breaks down a name validation error into the subcategories we explain below. We focus on possible changes in the browser name validation logic that could ameliorate these errors.

3a. WWW Mismatches. A certificate for `bank.com` does not work for `www.bank.com` and vice versa. In the past, browser developers turned down requests to change this behavior [29]. Our study helps empirically measure the impact of this decision. Note that we reuse the NSS name validation function, and do not write our own validation function for this category.

3b. Relaxed Matching. Recall from Section 4.3 that an asterisk in a name can only match one level of names in the initial part of the DNS label. A more permissive function that matches asterisks to an arbitrary number of sub-domains (while still ensuring that it is the same TLD+1) could accept more certificates and reduce the number of warnings shown to users. We measured this via our own implementation of such a relaxed matching algorithm, which is available online as part of our code release [1].

3c. Registered Domain Match. If a user connecting to `sub.example.com` receives a certificate for `example.com`, it is arguably a lower threat than an invalid certificate chain. Note that browsers already do not fully isolate sub-domains; e.g., the cookie policy does not isolate sub-domains. Arguably, a name validation error in which the SNI and the presented certificate share the registered domain is lower risk than if the presented certificate is for a totally different domain.

Identifying the registered domain in a given DNS label is tricky: each top-level domain (e.g., `.in`) has its own policy on the suffixes under which users can directly register names. For example, the registered domain for `test.example.in` is `example.in`, but it is not `gov.in` for `example.gov.in`. This is because `.in` allows registrations under both the `.in` as well as `.gov.in` suffixes. We use the Mozilla maintained public suffix list to identify the registered domains for a given DNS label [38].

3d. Multiple Names in certificate. As required by the standard [39], Firefox ignores multiple common names in a certificate and chooses the last common name in the certificate. Firefox also ignores the common name field if the subject alternative name field is present. We measure the impact of this behavior by manually extracting all the names in a certificate and, for each, directly calling the name validation function used by Firefox.

In addition to the errors discussed above, a number of other errors, which we seldom encounter, are not overridable by user. Examples for these errors are invalid encoded certificates, name constraint errors, or path length errors. Since we focus only on user-overridable errors, we ignore them.

5.3 Limitations

We assume that the network monitor does not encounter man-in-the-middle attacks. Practically, this means that an active network attacker near the web server can affect our measurements. We deem this a reasonable assumption since CAs already make a similar assumption while issuing DV certificates.

We collect data from a large number of academic and research institutions, representing nearly 300,000 users. The data collected may not represent the global TLS ecosystem. As part of our agreement, the data we collect cannot identify individual users. Thus, our data may over-represent a single user or a group of users.

We only look at connections using the SNI extension. All modern browsers send the SNI extension. 38% of connections in our data do not use the SNI extension. The warnings for this fraction could diverge from rest of the connections.

As we noted in Section 5.2.1, due to the caching of intermediates, we can only bound the errors in chain building. Similarly, Firefox allows users to cache certificate error overrides and to import additional root CAs. Thus, our error measurement may not represent the warnings shown to users.

Our model assumes that top-level warnings and mixed-content warnings consume the same user attention budget. If users separate these two warnings, our numbers might not apply directly. Unfortunately, the network monitor cannot easily distinguish top-level loads from secondary loads. Further, mixed content warnings also occur due to including HTTP content in a HTTPS page. Our measurements do not measure the impact of these errors on the user attention budget.

6. RESULTS AND DISCUSSION

98.46% of the filtered connections validate correctly, implying a false warning rate of 1.54%. The massive difference in the base-rates of hijacked connections versus connections that occur in benign scenarios makes the 1.54% error rate crushing. For example, consider what happens if an attack occurs only once in a million connection attempts. A 1.54% false warning rate means that the million connections cause

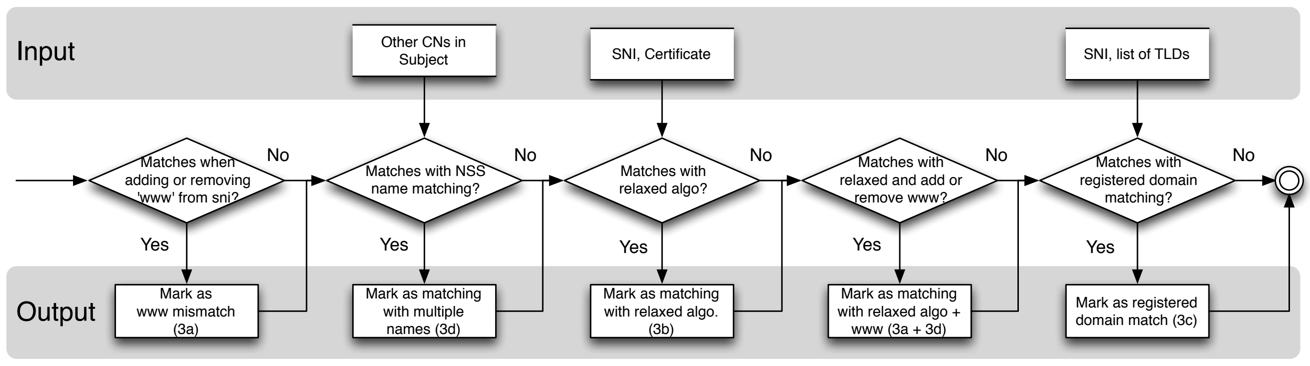
Input
Other CNs in Subject
SNI, Certificate
SNI, list of TLDs
Matches when adding or removing 'www' from sni?
Matches with NSS name matching?
Matches with relaxed algo?
Matches with relaxed and add or remove www?
Matches with registered domain matching?
No
Yes
Output
Mark as www mismatch (3a)
Mark as matching with multiple names (3d)
Mark as matching with relaxed algo. (3b)
Mark as matching with relaxed algo + www (3a + 3d)
Mark as registered domain match (3c)

Figure 2: Name validation error decision tree. Note the edges going from each output back to a decision node: each certificate can end up in multiple outputs.

	Error	*Connections*	*Certificates*
1a.	Unknown Issuer	70.51%	5,027
1b.	Self Signed Certificates	2.99%	6,126
2a.	Expired Certificates	7.65%	21,522
3.	Name Validation Error	18.82%	12,146

Table 1: Break up of benign errors into categories. The connection column indicates percentage of all erroneous connections.

15,401 warnings, out of which 15,400 are false warnings. It is not surprising that humans train themselves to ignore the warnings, thus clicking through the one true warning. Reducing the false warnings rate is critical to improving TLS security.

In this section, we discuss our results in full details based on the error categorization we presented in Section 5.2. For each error category, we measure the number of connections we see the error for. Users access some erroneous services an overwhelming number of times. To reduce the impact of such services, we also measure the number of certificates we see for each error category. Note that a single certificate corresponds to multiple connections, and thus can fall into multiple categories.

We structure the discussion into three parts: first, we discuss errors caused by server misconfigurations/errors. This includes chain validation errors (Section 6.1) and name validation errors (Section 6.2). Another class of errors occurs due to browser design decisions, such as AIA and revocation support. We measure and discuss the impact of these decisions in Section 6.3.

Based on our analysis, we also make concrete recommendations for reducing warning fatigue. Our recommendations center around three approaches: warning design to help focus user attention on high-risk events, browser modifications to conserve user attention, and technical innovations to ease TLS deployment and reduce errors.

6.1 Chain Validation Errors

Results. Table 1 breaks up the benign errors we see into the categories defined in Section 5.2. The first column lists the number of errors as a percentage of total connection errors, and the second column lists the number of certificates that manifest the particular error. Note that a particular certificate can occur multiple times in the second column: e.g., the same certificate can raise a name validation error in one connection and an expired certificate error in another. Below, we analyze these results and provide recommendations.

Unknown Issuer and Self-Signed Certificates. The overwhelming majority (73.50%) of erroneous connections involve unknown issuers or self-signed certificates. This indicates that a large number of websites opted out of the CA infrastructure. Unfortunately, an unknown issuer also represents a high-risk scenario. Reducing the severity of the warnings in such cases is infeasible. Instead, we emphasize technical measures to reduce the prevalence of such warnings in benign scenarios.

One of the reasons administrators opt-out of the CA infrastructure is the perceived cost of buying a valid certificate from a trusted authority. Recently, StartCom, a certificate authority trusted by all modern browsers, started offering free TLS certificates for simple use-cases [26].

Recommendation 1: We urge the community to advocate the use of free TLS certificates via authorities like StartCom [26]. Such advocacy, or evangelism, previously saw success in pushing for web standards as well as the ongoing push for mobile web standards [27, 34]. Anecdotal evidence suggests low awareness of these free certificates.

In some scenarios, the current CA infrastructure does not offer the flexibility needed. For example, a router manufacturer does not know the DNS label that the router will map to, and thus cannot get a certificate in advance.

Recommendation 2: We also suggest increasing the momentum on new standards like DNS-based Authentication of Named Entities (DANE) [17], which communicates public keys via DNSSEC. This allows the site or device in question to declare its public-key without relying on any issuer.

Our measurements also indicate the powerful usability advantages of network-view based approaches such as Convergence [6]. These systems can massively reduce the number of false warnings, and thus make actual attacks stand out. Unfortunately, these solutions involve contacting a "notary" server when connecting to a domain. This is a notable privacy and performance issue. Recent research aims to achieve Convergence-like guarantees in a privacy preserving and high performance manner [2], but further research is needed.

Error	*Connections*	*Certificates*
WWW Mismatches	1.17%	7.92%
Multiple Names	1.21%	0.03%
Relaxed Match	50.40%	7.24%
Relaxed with WWW	51.54%	13.87%
TLD Match	56.93%	29.73%

Table 2: Break up of name validation errors into sub-categories, as a percentage of total name validation errors.

Expired Certificates. As Table 1 demonstrates, expired certificates are the most common form of erroneous certificates (Column 2 Table 1). We examined the number of connections accessing an expired certificate and found that the median is four accesses, and the third quartile lies at twelve accesses. This indicates that expired certificate errors do not occur in popular services, but are common in the long tail.

To better understand the access patterns of expired certificates, we examined all certificates that occur in at least one connection with a timestamp larger than the `Not After` value. We then computed Δ, the difference of the connection timestamp to the certificate's expiration date. Since multiple connections may retrieve the same certificate, we see more than one Δ per certificate, and thus summarize all connections accessing the same certificate with the minimum, maximum, and median of the Δ values for each certificate.

For the minimum/median/maximum estimators, the first quartile lies at 2.8/5.2/6.8 days and the median at 35.8/52.8/64.7 days. In other words, 25% of all expired certificates are accessed only for a week after their expiry. This suggests that 25% of domains causing an expired certificate error renew their certificate within a week of expiry.

From a cryptographic standpoint, an expired certificate is no weaker than a valid one. Using expired certificates does not affect confidentiality and integrity of the communication, and raising a warning adversely affects the user attention budget.

This raises the question: why not accept all expired certificates, regardless of the expiration time? Domain ownership on the web is not constant. Ignoring expiration dates can allow past owners to serve visitors using an old certificate. We make the following recommendation:

Recommendation 3: Accept certificates that expired in the last week without an interstitial warning. Rely on an info-bar instead. Since expired certificates are low-risk, consuming user attention for such a scenario is not compelling. Instead, an info-bar, informing the user that the website will stop working in a week can warn the website administrator, without tiring out users.

The attack discussed above can also occur for one week in our proposal. We believe that the need to conserve user attention trumps the low risk of such an attack.

6.2 Name Validation Errors

Results. Table 2 breaks down the name validation errors into the sub-categories we defined in Section 5.2.3. The numbers denote the percentage of all name validation errors: e.g., 1.17% (7.92%) of all connections (certificates) with name validation errors involved a WWW mismatch. Unlike the previous table, a given connections can occur in more than one category. Below, we discuss the results further.

As seen in Table 1, name validation errors form the second most common category of errors. Note that these errors occur even in newer systems like Convergence. We find that Firefox's restrictive policy on multiple names has low impact on user attention. WWW mismatches also have a low impact in the number of connections, but a much higher impact when only looking at certificates. We consider name validation errors caused by WWW mismatches low risk.

Recommendation 4: Tolerate WWW mismatches in the certificate validation logic. Alternatively, browser vendors should show a different "low-risk" warning in such scenarios.

50.40% (7.24%) of connections (certificates) with name validation errors validate if we switch to matching multiple levels with a $*$. Accepting `www` mismatches increases this number to 51.51% (13.87%). The wide prevalence of such errors indicates a misunderstanding: website administrators are not aware of the browser's limited glob expansion strategy. In one of the bugs we filed during this work, a NSS developer also suggested switching to a relaxed matching scheme [32]. While the standard recommends against a relaxed name validation scheme, it does not prohibit it [40].

Recommendation 5: Use a more relaxed name validation algorithm that accepts multiple levels for an asterisk.

A significant number of errors occur where the certificate and the connection targets match in their registered domain. We consider this a low risk scenario and make the following recommendation:

Recommendation 6: Modify the warning for sub-domain mismatches, and help focus user attention on the high-risk scenarios. For example, the warning shown when the `admin.bank.com` server presents a certificate for `www.bank.com` should indicate a lower risk than the warning shown when the `bank.com` server presents a certificate for the unrelated `attacker.com`.

While a full manual analysis of the remaining errors is difficult, it appears that a large number of errors occur due to content distribution networks (CDNs) such as Akamai, EdgeCast, CloudFlare. Popular websites like Facebook rely on CDNs to serve content. We find a number of connections with a SNI value of facebook.com but a server certificate for a CDN such as Akamai. We believe that the reason for the prevalence of these errors is the difficulty of detecting and diagnosing such errors.

Consider an image load with an `onerror` event handler that logs load failure. The browser's error event does not provide any information about the cause of a load error. Thus, the certificate error at the CDN, likely caused due to a misconfiguration, is indistinguishable from a missing image (a 404) or an error in the user's network connection. We posit that this makes it difficult for large-scale websites such as Facebook to detect and diagnose these intermittent TLS issues with their CDN providers.

Recommendation 7: Browsers should provide the `onerror` event with information about the certificate validation error, if any. This information allows the website administrator to log and track down any issue.

	Error	*Connections*	*Number*
1c.	Incomplete Chains	21,449,989	29,661
1c.	Incomplete Chains - AIA	1,623,047	450
2b.	Revoked Certificates - Chrome	103,472	301
2b.	Revoked Certificates - CRLs	3,364,809	1,519

Table 3: Impact of browser design decisions on false warnings.

In the past, Facebook raised a similar concern with browser handling of script errors, and browser vendors modified code to accommodate Facebook's need to diagnose such errors [3]. As TLS achieves wider "always-on" deployment, easily diagnosing errors will become critical.

6.3 Impact of Browser Design Decisions

Results. Table 3 outlines the results of our measurement of the impact of AIA, caching, and revocation lists on false warnings. The first two rows measure the impact of intermediate caching and AIA fetching: 21,449,989 connections fail with both disabled, 1,623,047 fail with just intermediate caching disabled. The next two rows measure the impact of revocation lists used: the third row measures the connections that fail due to Chrome's CRL, while the last row measures the connections that fail with the normal CRLs. We discuss the results further below.

Incomplete Chains. Recall that incomplete chains can still verify due to cached certificates. To measure the impact of caching of intermediate certificates, we disabled caching and tried to validate all unique chains in our data. This caused 8.13% of the valid, unique chains (or 29,661 chains) to fail validation.

The AIA extension allows a server to send an incomplete chain, but include a URL to fetch the requisite intermediates from. Among the major browsers, only Firefox does not fetch intermediates mentioned in this field. To measure the impact of this decision, we took the 29,661 chains that failed due to a missing intermediate, and tried to validate them with AIA fetching enabled. We found that 98.48% of chains that did not validate with caching disabled successfully validated with AIA fetching enabled.

Disabling AIA support puts an unnecessary burden on end user attention budget. Despite being valid, each of the 19,826,942 connections and 29,211 certificates can cause a TLS warning on a clean cache. Problems with enabling AIA support include possible privacy implications. Alternative approaches like preloading the browser with all intermediate authorities can also achieve similar results, without the privacy impact [31].

Recommendation 8: Enable AIA support or preload all known intermediate authorities in the browser cache.

Revoked Certificates. CAs use revocation for administrative as well as security reasons. While Firefox uses the CRL as published by CAs, Google Chrome relies on its own revocation list. Currently, Chrome's revocation list contains 18,972 certificates. The CRLs for all the intermediate and root authorities in our database have 917,284 certificates.

As seen in Table 3, we see five times the certificates present in CRLs compared to certificates present in Chrome's list. This difference translates to nearly thirty three times as many errors.

Recommendation 9: Switch to separate administrative and security revocation lists, with distinct warnings for each. Since the number of connections using certificates in the security revocation list is small, browsers should consider errors due to the security revocation lists a hard fail, and not consume the user attention with an overridable warning.

It is not clear what browser behavior for the administrative revocations should be. While the CRL format offers a field to describe the reason for revocation, most authorities do not use it. In our data, 70.9% of the CRL entries do not include a reason. Further research into the causes of these revocations is needed to shed light on this issue.

7. CONCLUSION

Browsers do not consider the dangers of habituation when showing a warning. This, coupled with the prevalence of false warnings, reduces the security of the TLS protocol, as users train themselves to click through warnings. By measuring the prevalence of different types of false warnings, we provide a framework for browsers to reevaluate their current warning mechanisms and conserve user attention. We also presented a number of concrete recommendations based on our analysis. We have shared our data and results with browser vendors, and already received positive and encouraging feedback.

8. ACKNOWLEDGMENTS

We would like to thank Adrienne Porter-Felt, Matthew Finifter, Michael McCoyd, Kurt Thomas, Paul Pearce, Jeff Hodges, and Warren He for taking time to read drafts of this paper. We would also like to thank Brian Smith, Nasko Oskov, and Adam Langley for taking time to answer our queries about browser internals.

This research was supported by Intel through the ISTC for Secure Computing; by the Air Force Office of Scientific Research under MURI grant numbers 22178970-4170 and FA9550-08-1-0352; by the National Science Foundation under grant numbers OCI-1032889, 0831501-CT-L, CCF-0424422, and 0842695; by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD); and by the Office of Naval Research under MURI Grant Number N000140911081. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF, the AFOSR, the ONR, the DAAD, or Intel.

9. REFERENCES

[1] Amann, J. Github repositories. `https://github.com/0xxon/tls-validation-measurement`.

[2] Amann, J., Vallentin, M., Hall, S., and Sommer, R. Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service. Tech. Rep. TR-12-014, International Computer Science Institute, Nov. 2012. `http://www.icsi.berkeley.edu/pubs/techreports/ICSI_TR-12-014.pdf`.

[3] Barth, A. X-Script-Origin, we hardly knew ye, Oct 2011. `http://www.schemehostport.com/2011/10/x-script-origin-we-hardly-knew-ye.html`.

[4] Böhme, R., and Grossklags, J. The Security Cost of Cheap User Interaction. In *Proceedings of the 2011 New Security Paradigms Workshop* (2011), ACM, pp. 67–82.

[5] Chromium Authors. HSTS Preload and Certificate Pinning List. https://src.chromium.org/viewvc/chrome/trunk/src/net/base/transport_security_state_static.json.

[6] Convergence. http://www.convergence.io.

[7] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[8] Dhamija, R., Tygar, J., and Hearst, M. Why Phishing Works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (2006), ACM, pp. 581–590.

[9] Dreger, H., Feldmann, A., Mai, M., Paxson, V., and Sommer, R. Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection. In *USENIX Security Symposium* (2006).

[10] Eastlake, D. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066 (Proposed Standard), Jan. 2011.

[11] Egelman, S., Cranor, L. F., and Hong, J. You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings. In *Proceedings of The 26th SIGCHI Conference on Human Factors in Computing Systems (CHI)* (2008).

[12] Electronic Frontier Foundation. The EFF SSL Observatory. https://www.eff.org/observatory.

[13] Fahl, S., Harbach, M., Muders, T., Smith, M., Baumgärtner, L., and Freisleben, B. Why Eve and Mallory Love Android: An Analysis of Android SSL (In)security. In *Proceedings of the 2012 ACM conference on Computer and communications security* (New York, NY, USA, 2012), CCS '12, ACM, pp. 50–61.

[14] Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)* (2012).

[15] Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D., and Shmatikov, V. The most dangerous code in the world: validating SSL certificates in non-browser software. In *ACM Conference on Computer and Communications Security* (2012), pp. 38–49.

[16] Hodges, J., Jackson, C., and Barth, A. Http strict transport security (hsts). *Internet Engineering Task Force (IETF) RFC draft* (2011).

[17] Hoffman, P., and Schlyter, J. The DNS-Based Authentication of Named Entities (DANE): TLSA Protocol. RFC 6698, Aug. 2012.

[18] Holz, R., Braun, L., Kammenhuber, N., and Carle, G. The SSL landscape: a thorough analysis of the x.509 PKI using active and passive measurements. In *Proc. 2011 ACM SIGCOMM conference on Internet measurement conference* (2011).

[19] Karlof, C., Tygar, J., and Wagner, D. Conditioned-safe Ceremonies and a User Study of an Application to Web Authentication. In *Sixteenth Annual Network and Distributed Systems Security Symposium (NDSS 2009)* (February 2009).

[20] Krosnick, J. Response strategies for coping with the cognitive demands of attitude measures in surveys. *Applied cognitive psychology 5*, 3 (1991), 213–236.

[21] Langley, A. Revocation checking and Chrome's CRL. http://www.imperialviolet.org/2012/02/05/crlsets.html, Feb. 2012.

[22] Langley, A. SSL Interstitial Bypass Rates, February 2012. http://www.imperialviolet.org/2012/07/20/sslbypassrates.html.

[23] Laurie, B., Langley, A., and Kasper, E. Certificate Authority Transparency and Auditability. https://tools.ietf.org/html/draft-laurie-pki-sunlight-02.

[24] Leavitt, N. Internet security under attack: The undermining of digital certificates. *Computer 44*, 12 (2011), 17–20.

[25] Lenstra, A., Hughes, J., Augier, M., Bos, J., Kleinjung, T., and Wachter, C. Ron was wrong, Whit is right. *IACR eprint archive 64* (2012).

[26] Ltd., S. StartSSL Free Certificate. https://www.startssl.com/?app=1.

[27] MozillaWiki: Mobile Web Evangelism. https://wiki.mozilla.org/Mobile/Evangelism.

[28] Motiee, S., Hawkey, K., and Beznosov, K. Do Windows Users Follow the Principle of Least Privilege? Investigating User Account Control Practices. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)* (2010).

[29] Mozilla Bugzilla. Bug 364667: Tolerate mismatch between certificate host and actual host if the difference is only "www.". https://bugzil.la/364667.

[30] Mozilla Bugzilla. Bug 634074: Cannot validate valid certificate chain when looping/cross-signed certs are involved. https://bugzil.la/634074.

[31] Mozilla Bugzilla. Bug 657228 – Preload all known intermediate certificates for CAs in our root store. https://bugzil.la/657228.

[32] Mozilla Bugzilla. Bug 806281: NSS doesn't use publicsuffix list. https://bugzil.la/806281.

[33] Mozilla Bugzilla. Bug 808331: Intermediates Cached in Private Browsing Mode. https://bugzil.la/808331.

[34] Mozilla Bugzilla. Tech Evangelism Bugs. http://is.gd/sqCtFm.

[35] MozillaWiki: Maintaining Confidence in Root Certificates. https://wiki.mozilla.org/CA:MaintenanceAndEnforcement#Actively_Distrusting_a_Certificate.

[36] Network Security Services (NSS). https://www.mozilla.org/projects/security/pki/nss/.

[37] Paxson, V. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks 31*, 23-24 (1999), 2435–2463.

[38] Public Suffix List. http://publicsuffix.org.

[39] Rescorla, E. HTTP Over TLS. RFC 2818 (Informational), May 2000.

[40] Saint-Andre, P., and Hodges, J. Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security. RFC 6125, March 2011.

[41] Santesson, S., and Hallam-Baker, P. Online Certificate Status Protocol Algorithm Agility. RFC 6277 (Proposed Standard), June 2011. http://www.ietf.org/rfc/rfc6277.txt.

[42] Soghoian, C., and Stamm, S. Certified lies: Detecting and defeating government interception attacks against ssl (short paper). *Financial Cryptography and Data Security* (2012), 250–259.

[43] Sotirakopoulos, A., Hawkey, K., and Beznosov, K. On the challenges in usable security lab studies: lessons learned from replicating a study on SSL warnings. In *Proceedings of the Seventh Symposium on Usable Privacy and Security* (New York, NY, USA, 2011), SOUPS '11, ACM, pp. 3:1–3:18.

[44] Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., and Cranor, L. F. Crying Wolf: An Empirical Study of SSL Warning Effectiveness. In *Proceedings of the 18th Usenix Security Symposium* (2009).

[45] The Electronic Frontier Foundation. The Sovereign Keys Project. `https://www.eff.org/sovereign-keys`.

[46] Vratonjic, N., Freudiger, J., Bindschaedler, V., and Hubaux, J. The inconvenient truth about web certificates. *Workshop on Economics of Information Security and Privacy* (2011), 79–117.

[47] Wendlandt, D., Andersen, D. G., and Perrig, A. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In *USENIX Annual Technical Conference* (2008).

[48] Whalen, T., and Inkpen, K. Gathering evidence: use of visual security cues in web browsers. In *Proceedings of Graphics Interface 2005* (2005), Canadian Human-Computer Communications Society, pp. 137–144.

[49] Wikipedia. Accuracy of Ranking by Alexa Toolbar, 2012. `https://en.wikipedia.org/wiki/Alexa_Internet#Accuracy_of_ranking_by_the_Alexa_Toolbar`.