

Web Security Context: Experience, Indicators, and Trust

W3C Working Draft 1 November 2007

This version:

<http://www.w3.org/TR/2007/WD-wsc-xit-20071101/>

Latest version:

<http://www.w3.org/TR/wsc-xit/>

Editors:

Thomas Roessler, [W3C](#)

Anil Saldhana, [RedHat](#)

[Comments in this color by Tim Hahn, 27 November 2007](#)

Copyright © 2007 W3C® ([MIT](#), [ERCIM](#), [Keio](#)), All Rights Reserved. W3C [liability](#), [trademark](#) and [document use](#) rules apply.

Abstract

This specification defines guidelines and requirements for the presentation and communication of Web security context information to end-users; ceremonies for secure data entry; and good practices for Web Site authors.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the [W3C technical reports index](#) at <http://www.w3.org/TR/>.

This is a First Public Working Draft of "Web Security Context: Experience, Indicators, and Trust".

The [Web Security Context Working Group](#) is publishing this document to provide a basis for initial review of and commentary on its work. The Working Group has taken an inclusive approach toward publishing various technical proposals in this First Public Working Draft. Inclusion of technical material in this document does not indicate group consensus about that material; some requirements stipulated in this document are known to be mutually exclusive. No claims as to the efficacy of usability-related material are made.

To facilitate access to relevant background, various sections of this document are annotated with references to input documents that are available from the [Working Group's Wiki](#), and to pertinent [issues](#) that the group is tracking. The documents in the wiki include background, motivation, and usability concerns on the proposals that reference them. They provide important context for understanding the potential utility of the proposals.

The Working Group expects to advance this Working Draft to Recommendation Status.

Please send comments about this document to public-usable-authentication@w3.org (with [public archive](#)).

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the [5 February 2004 W3C Patent Policy](#). W3C maintains a [public list of any patent disclosures](#) made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains [Essential Claim\(s\)](#) must disclose the information in accordance with [section 6 of the W3C Patent Policy](#).

[Table of Contents](#)

- 1 [Overview](#)
- 2 [Acknowledgements](#)
- 3 [Conformance](#)
 - 3.1 [Conformance Model](#)
 - 3.2 [Conformance Labels](#)
 - 3.2.1 [Conformance Labels for Web Content](#)
 - 3.2.2 [Conformance Labels for Web User Agents](#)
- 4 [Interaction and content model](#)
 - 4.1 [Overview](#)
 - 4.2 [Terms and definitions](#)
 - 4.2.1 [Common User Interface elements](#)
 - 4.2.2 [Attacks](#)
- 5 [Applying TLS to the Web](#)
 - 5.1 [Using TLS to secure HTTP transactions](#)
 - 5.2 [Relaxed Certificate Path Validation](#)
 - 5.3 [Certificates](#)
 - 5.3.1 [Augmented Assurance Certificates](#)
 - 5.3.2 [Attested Certificates](#)
 - 5.3.3 [Self-signed Certificates](#)
 - 5.3.4 [No Interaction Certificates](#)

- 5.3.5 [Logotype Certificate](#)
 - 5.3.6 [Interactively accepting trust anchors or certificates](#)
 - 5.3.7 [Trusted Certificates](#)
- 5.4 [Using TLS to secure Web Content](#)
- 5.5 [Change of security level](#)
 - 5.5.1 [Certificate errors](#)
 - 5.5.2 [Redirection chains](#)
 - 5.5.3 [Change against historical practice](#)
 - 5.5.4 [Explicit user expectation](#)
- 6 [Indicators and Interactions](#)
 - 6.1 [Identity and Trust Anchor Signalling](#)
 - 6.1.1 [Identity Signal](#)
 - 6.1.2 [Identity Signal Content](#)
 - 6.2 [Additional Security Context Information](#)
 - 6.3 [Page Security Score](#)
 - 6.4 [Error handling and signalling](#)
 - 6.4.1 [Basic error handling interactions](#)
 - 6.4.2 [Handling certain man in the middle attacks](#)
- 7 [Safe Web Form Editor](#)
 - 7.1 [Associating a history with a web site](#)
 - 7.2 [No support for non-TLS data exchange](#)
 - 7.3 [Creating a new relationship](#)
 - 7.4 [Reliable Text](#)
 - 7.5 [Selection of a text string](#)
 - 7.6 [On screen masking of a text string](#)
 - 7.7 [Editing the stored history](#)
 - 7.8 [Picture-in-picture defense](#)
 - 7.9 [Detecting a possible Man-In-The-Middle attack](#)
- 8 [Robustness](#)
 - 8.1 [Do not mix content and security indicators](#)
 - 8.1.1 [Conformance note](#)
 - 8.1.2 [Requirement](#)
 - 8.1.3 [Techniques](#)
 - 8.2 [Establishing a trusted path between users and browsers](#)
 - 8.3 [APIs exposed to Web content](#)
 - 8.3.1 [Requirements](#)
 - 8.3.2 [Techniques](#)
- 9 [Authoring and deployment best practices](#)
 - 9.1 [Do not put Security Indicator images to indicate trust in content](#)
 - 9.2 [Use TLS for Login Pages](#)
 - 9.3 [Use TLS Consistently across the web site](#)

- 9.4 [Redirects from Non-Secure to Secure page and vice-versa](#)
 - 10 [Usage Modes](#)
 - 10.1 [Framework](#)
 - 10.2 [Safe Browsing Mode](#)
 - 11 [Security Considerations](#)
 - 11.1 [Active attacks during initial TLS interactions](#)
 - 11.2 [Using error handling interactions as vectors for user deception](#)
 - 11.3 [Secure storage of sensitive information](#)
 - 12 [References](#)
-

1 Overview

This specification deals with the trust decisions that users must make online, and with ways to support them in making safe and informed decisions where possible.

In order to achieve that goal, this specification includes recommendations on the presentation of identity information by Web user agents; on handling errors in security protocols in a way that minimizes the trust decisions left to users, and (we hope) induces them toward safe behavior where they have to make these decisions; and on data entry interactions that (we hope, again) will make it [easier for more likely that](#) users ~~to~~ enter sensitive data into legitimate sites than ~~to~~ enter ~~them that data~~ into illegitimate sites.

Where this document specifies user interactions with a goal toward making security usable, no claim is made at this time that this goal is met: As noted in the [Status of this Document](#) section, this is an initial draft to trigger discussion and commentary; assume that what is proposed here is untested.

To complement the interaction and decision related parts of this specification, [8 Robustness](#) addresses the question of how the communication of context information needed to make decisions can be made more robust against attacks.

Finally, [9 Authoring and deployment best practices](#) is about practices for those who deploy Web Sites. It complements some of the interaction related techniques recommended in this specification. The aim of this section is to provide guidelines for creating Web sites with reduced attack surfaces against certain threats, and with usefully provided security context information.

This specification comes with two companion documents: [\[WSC-USECASES\]](#) documents the use cases and assumptions that underly this specification. [\[WSC-THREATS\]](#) documents the Working Group's threat analysis.

2 Acknowledgements

The following participants of the Web Security Context Working Group contributed to this document: Mike Beltzner, Tyler Close, Stephen Farrell, Timothy J Hahn, Phillip Hallam-Baker, Mike McCormick, Johnathan Nightingale, Yngve N. Pettersen, Thomas Roessler, Dan Schutzer, Mary Ellen Zurko.

3 Conformance

3.1 Conformance Model

This section is normative.

Normative material in this specification is marked explicitly.

This specification defines (a) requirements for user interactions and interfaces that are exposed by Web user agents and (b) good practices for Web Content. Requirements for both are phrased in terms of the definitions and concepts found under [4 Interaction and content model](#), and in terms of the definitions and concepts found under [5 Applying TLS to the Web](#). These sections are normative, and part of compliance requirements.

Sections that specify requirements for products are clearly marked with the class of product that they apply to. Preconditions that conforming products need to fulfill for the requirement to be applicable are identified. In addition to requirements, the specification includes implementation techniques. These are labelled as NECESSARY (a technique MUST be implemented in order for a product to conform with the requirement) or SUFFICIENT (a technique MAY be implemented, and implementation leads to conformance).

Throughout the specification, the [IETF RFC 2119 \[RFC2119\]](#) keywords MUST, MUST NOT, SHOULD, SHOULD NOT, MAY are applied, with their respective meanings.

Some conformance requirements might actually lead to usability testing as a prerequisite to claiming conformance. Is that a good idea, and how do we deal with it? See also [8.1.1 Conformance note, ISSUE-112](#).

3.2 Conformance Labels

This section is a placeholder for a more detailed explanation of what conformance with this specification will mean for Web Content and for Web User Agents.

3.2.1 Conformance Labels for Web Content

To be written.

3.2.2 Conformance Labels for Web User Agents

To be written.

4 Interaction and content model

4.1 Overview

This specification assumes a human user interacting with a Web user agent [which is](#) interacting with

Web resources [on the user's behalf](#). Many of the requirements specified are focused on the presentation of security context information to the user, and therefore directly relate to user interfaces. Where requirements or techniques are specific to certain modalities, these are made explicit, and are part of the preconditions for applying the requirement or technique.

When this specification speaks of a "[Web user agent](#)" to describe the application through which a user interacts with the Web, then this term is used on a conceptual level: No assumption is made about implementation details; the "Web user agent" may denote a combination of several applications, extensions to such applications, operating system features, and assistive technologies.

This specification makes [only one no-specific](#) assumption about the content with which the user ~~interacts~~[is interacting, except for one](#). ~~There~~[We assume that there](#) is a top-level [Web page](#) that is identified by a URI [\[RFC3986\]](#). This Web page might be an HTML frameset, an application running on top of a proprietary run-time environment, or a document in a format interpreted by plug-ins or external systems served as part of a Web interaction. The page's behavior might be determined by scripting, stylesheets, and other mechanisms.

Some requirements are expressed in terms of user interface components commonly found in current-generation [Web user agents](#).

[4.2 Terms and definitions](#) is expected to be consistent with the Web Content Accessibility Guidelines Version 2, [\[WCAG20\]](#).

[4.2 Terms and definitions](#)

The definitions in this section are normative. The examples are informational.

[Definition: A **Web User Agent** is any software that retrieves and presents Web content for users.]

[Definition: A **Web Page** is a resource that is referenced by a URI and is not embedded in another resource, plus any other resources [\[tjh: does this include data://, jar:// URI formatted references to resources? Other mime-types?\]](#) that are used in the rendering or intended to be rendered together with it.]

[Definition: **Web Content** is a set of Web Pages].

[4.2.1 Common User Interface elements](#)

This section defines terms for user interface elements commonly present in [Web User Agents](#). These definitions are normative.

[Definition: **Primary User Interface** denotes the portions of a [Web user agent's](#) user interface that are available to users without being solicited by a user interaction.]

Examples of primary user interface include the location bar in common desktop Web user agents, the "padlock" icon present in common desktop Web user agents, or error pages that take the place of a Web page that could not be retrieved.

[Definition: **Secondary User Interface** denotes the portions of a [Web user agent's](#) user interface that are available to the user after they are solicited by a specific user interaction.]

Examples of secondary user interface include the "Page Information" dialogue commonly found in desktop Web user agents, and the "Security Properties" dialogue that can be obtained by clicking the padlock icon in common desktop Web user agents.

[Definition: **Location Bar** is a widget in a Web user agent's user interface which displays (and often allows input of) the textual location (entered as a URL) of the resource being requested (or displayed - after the response is received).]

4.2.2 Attacks

[Definition: A **Whack-A-Mole attack** refers to a website with the malicious intent of performing an unintended action (eg. installing software that would have required a user intervention such as clicking OK on a warning dialog) or by exploiting distraction [and from](#) task-focus. The website opens a large number of new windows over the desired web content and the malicious action is performed when the user tries to close these new windows and/or clicks on a dialog that indicates a trust decision.]

[[tjh: seems like there should be more here: XSS, DNS poisoning, MITM and so on](#)]

5 Applying TLS to the Web

Please refer to the following entries in the Working Group's Wiki for relevant background information: [WhatIsASecurePage](#)

5.1 Using TLS to secure HTTP transactions

Some of the terminology in this section is known to be misleading. It will be cleaned up in a later version of this draft. [ISSUE-113](#)

The most common mechanism for applying TLS to the Web is the use of the `https` URI scheme [\[RFC2818\]](#); the alternative upgrade mechanism [\[RFC2817\]](#) is used rarely, if at all. For the purposes of this specification, the most relevant property of [\[RFC2818\]](#) is that the URI used to identify a resource includes an assertion that use of TLS is desired.

This specification uses the term [Definition: **HTTP transaction**] regardless of whether any kind of TLS protection was applied; in particular, the transactions arising when an `https` URL is dereferenced are subsumed under this term.

[Definition: (normative) An HTTP transaction is **TLS-desired** if the resource is identified through a URL with the `https` URI scheme.]

[Definition: (normative) An HTTP transaction is **TLS-protected** if an upgrade according to [\[RFC2817\]](#) is performed successfully, or if the resource was identified through a URL with the `https` URI scheme, the TLS handshake is finished successfully, and the HTTP transactions have occurred through the TLS

[communications](#) channel.]

Note that an HTTP transaction may be considered [TLS protected](#) even though weak algorithms (including NULL encryption) are negotiated.

[Definition: (normative) An HTTP transaction is **strongly TLS-protected** if it is [TLS-protected](#), an [https](#) URL was used, and the following conditions are true:]

1. the server used a [trusted certificate](#) ~~that~~[whose subjectName](#) matches the dereferenced URI's [DNSname element](#) [tjh: there is an IETF RFC reference for this, I believe]
2. [strong TLS algorithms](#) were negotiated for both confidentiality and integrity protection.

If we are reacting to certs that don't match a URL then we need a well defined matching rule. This should probably be solved by way of some appropriate normative references. [ISSUE-106](#), [ISSUE-121](#)

TLS modes that do not require the server to show a certificate (such as the DH_anon mode) do not lead to a strongly TLS-protected transaction.

[Definition: (normative) **Strong TLS algorithms** are defined as the algorithms recommended by [\[ref-ALGORITHMS\]](#).]

What reference should we have here? [ISSUE-128](#)

[Definition: (normative) An HTTP transaction is **weakly TLS-protected** if it is [TLS-protected](#), but [strong TLS protection](#) could not be achieved for one of the following reasons:]

1. cryptographic material was exchanged through an anonymous key exchange algorithm such as DH_anon
2. the cryptographic algorithms negotiated are not considered [strong](#)
3. certificates were shown which are not [attested](#), and not trusted for the destination of the transaction

Note that a situation in which an attested certificate is shown, and path validation fails, does not lead to a weakly TLS-protected interaction.

[5.2 Relaxed Certificate Path Validation](#)

This section is normative.

Basic Path Validation (section 6 of [\[RFC3280\]](#)) verifies the binding between the subject distinguished name and/or alternative name, and the subject public key, based on a path of certificates that leads to a trust anchor. As part of Basic Path Validation, Basic Certificate Processing is performed. This processing includes a verification of certificate status at the current time. Status verification is often not performed by Web user agents.

This specification defines a Relaxed Path Validation algorithm. [Definition: **Relaxed Path Validation**] differs from Basic Path Validation in its handling of certificate validity and status; specifically:

1. Instead of performing step (a)(2) of Basic Certificate Processing (section 6.1.3 of [\[RFC3280\]](#);

verification of validity date), the Web user agent verifies that the intersection of the validity period of all certificates in the path (including the entity certificate and trust anchor) is non-empty. If this intersection is empty, relaxed path validation fails.

2. Step (a) (3) (status verification) is skipped.

Note that this variant of Basic Path Validation does not require a real time source. [\[tjh: I think it would be useful to describe the pros and cons of this statement.\]](#)

5.3 Certificates

Web user agents can derive trust in the validity of identity certificates that are presented by Web servers from various sources, including user action, previous interactions, and attestation from trusted Certification Authorities by way of a valid certificate chain [\[RFC3280\]](#).

The practices used by Certification Authorities (and the information attested to) vary by CA. Whether a Certification Authority's root certificate is installed as a trust anchor is a decision typically made by Web user agent vendors and systems administrators, based on out-of-band information. [Users rarely consider this list and the list typically differs between different user agents which the same user interacts with.](#)

[Weakly TLS-protected](#) interactions may provide security services such as confidentiality protection against passive attackers, or integrity protection against active attackers (without confidentiality protection). These properties are often desirable, even if [strong TLS protection](#) cannot be achieved.

5.3.1 Augmented Assurance Certificates

Please refer to the following entries in the Working Group's Wiki for relevant background information: [RecommendationDisplayProposals/EVCerts](#)

This section is normative.

[Definition: An **Augmented Assurance (AA) Certificate** is an identity certificate that asserts that the subject entity has been authenticated by means of a process that has been audited and is designed to establish accountability in accordance with an industry standard set of criteria, e.g. [\[EVSSLCERT\]](#). The certificate chain for such a certificate MUST be validated up to a trust root that is recognized as AA-qualified by the user agent.]

AA-ness is a certificate property, e.g., established through some policy OID. Text above needs to be adapted to reflect this.

This specification assumes that Web user agents establish that a trust root is [Definition: **AA-qualified**] through security-critical application-specific out-of-band mechanism. It is further assumed that Issuer and Subject information included in Augmented Assurance Certificates is valid, and intended to be displayed to users.

Implementations MUST NOT enable users to designate trust roots as AA-qualified as part of a different interaction. Implementations MAY make user interfaces available for the purpose of designating AA-qualified trust roots. Such user interfaces MUST be focused on this specific task. In particular, the

notions of an AA-qualified trust root and an [interactively](#) accepted trust root are mutually exclusive.

Implementations MUST NOT consider a certificate that otherwise matches the definition as an Augmented Assurance Certificate if the certificate is marked as a [no interaction certificate](#)

Implementations MUST NOT use [Relaxed Path Validation](#) if the trust anchor is AA-qualified.

5.3.2 Attested Certificates

This section is normative.

[Definition: An **attested certificate** is an identity certificate issued by a Certification Authority for which the certificate chain ~~can be~~ [has been](#) validated up to a trust root that is considered qualified for this purpose.]

This specification assumes that Web user agents establish that a trust root is [Definition: **qualified to attest**] through a security-critical out-of-band mechanism that is out of scope for this specification. It is further assumed that Issuer and Subject [\[RFC3280\]](#) information included in **attested certificates** is valid, and intended to be displayed to the user.

All Augmented Assurance certificates are also considered attested certificates.

5.3.3 Self-signed Certificates

Self-signed certificates are commonly used for appliances and web sites catering to small groups of users, and essentially serve as a container for cryptographic key material in an anonymous key exchange.

They provide confidentiality protection services against passive attackers. No binding of an asserted identity to the cryptographic key is achieved; self-signed certificates are, in particular, not [attested certificates](#). However, repeated interactions with the same web site (as identified by authority and port segments of the URI) over an extensive time period that are [TLS-protected](#) and involve the same self-signed certificate create strong evidence that the user is achieving protection against an active attacker as well. Exploiting this behavior by way of a "leap of faith" is known as a common practice in deployments of the Secure Shell Connection Protocol [\[RFC4254\]](#).

[Definition: (normative) A self-signed certificate is called **proven for a destination** if it has been used for [TLS-protected](#) interactions with resources whose URIs share the same authority (domain) and port number for an extensive amount of time, the "**probation time**."] [\[tjh: need a definition for "probation time". Candidate: time period during which a self-signed certificate is accepted for use but not yet accepted for use without active user intervention on every use.\]](#)

The length of the probation time for a self-signed certificate is implementation-specific. Note that a self-signed certificate can only be proven for a specific destination. [\[tjh: Why can't two destinations use the same self-signed certificate as long as each site goes through its own probation period? Or perhaps this would violate the DNS name reference in the subjectName of the certificate. But then this would not be a self-signed certificate-specific issue.\]](#)

Much of the material in this section could be applied to *any* container for key material, including certificates from unknown CAs, and anonymous key exchange algorithms. At the same time, some information about the *invalidity* of certificates might be gained from certificates that don't lead back to a known trust anchor. [ISSUE-103](#)

5.3.4 No Interaction Certificates

Please refer to the following entries in the Working Group's Wiki for relevant background information: [RecommendationDisplayProposals/NoSecurityIndicator](#)

[Definition: (normative) An end entity certificate that contains the User Experience suppression extension [\[ref-UESOID\]](#) is called a **no-interaction-cert** .]

It is not clear what the reference should be in this section. It is also not clear whether "no interaction" is the correct terminology. See [ISSUE-119](#).

[\[tjh: I think it would help to provide an explanation of why these are useful and where they are used.\]](#)

5.3.5 Logotype Certificate

Please refer to the following entries in the Working Group's Wiki for relevant background information: [RecommendationDisplayProposals/Letterhead](#)

RFC 3709 [\[RFC3709\]](#) defines a certificate extension to embed three kinds of logotypes with an X.509 certificate, for use with public key certificates [\[RFC3280\]](#) or attribute certificates [\[RFC3281\]](#).

[Definition: **The Community logotype** [\[RFC3709\]](#) is a logotype that identifies a community.]

[Definition: **The Issuer Organization logotype** [\[RFC3709\]](#) is a logotype that identifies the organization that issued the certificate.]

[Definition: **The Subject Organization logotype** [\[RFC3709\]](#) is a logotype that identifies the organization to which the certificate was issued.]

RFC 3709 also specifies a format to encapsulate audio data with certificates. Not yet covered. [ISSUE-120](#)

[A logotype certificate is a certificate which contains one or more of these extensions.](#)

5.3.6 Interactively accepting trust anchors or certificates

[Definition: A trust anchor or certificate is **interactively accepted** if the acceptance was caused through a user interaction that happens ~~whthi~~[while](#) the user is focused on a primary task unrelated to trust and certificate management.]

For example, if a certificate is presented for acceptance by a user during ordinary Web interactions, and is accepted by the user, then this matches the test for interactive acceptance. If, however, a systems administrator (or user) adds a Certification Authority's certificate to a browser's store of trust roots, then that certificate is not considered accepted interactively.

5.3.7 Trusted Certificates

This section is normative.

The term [Definition: [[**trusted**]] **certificate**] is used to denote certificates that are trusted to bind cryptographic credentials to a set of URIs. A certificate can acquire this property in a number of ways, ranging from attested certificate attributes to out-of-band communication. This specification does not present a comprehensive list of possible trust anchors. The precise trust anchors used by a given Web User agent are implementation-dependent.

However, the following types of certificates **MUST NOT** be considered [trusted certificates](#) by Web user agents, and therefore only lead to [weakly TLS-protected](#) interactions:

- [no interaction certificates](#)
- self-signed certificates during the [probation time](#)

Further, Web user agents **MUST** consider [attested](#) certificates (including [Augmented Assurance Certificates](#)) to be trusted [and thus result in strongly TLS-protected connections](#).

[Self-signed certificates proven for a destination](#) **MAY** be considered as [trusted certificates](#) by Web user agents for the specific destination for which they are proven [and thus result in a strongly TLS-protected connections](#). However, Web user agents **MUST NOT** conclude that any assertions that may be included with the certificate are valid.

5.4 Using TLS to secure Web Content

This section is normative.

If a given [Web page](#) consists of a single resource only, then all content that the user interacts with has security properties derived from the HTTP transaction used to retrieve the content.

[Definition: A [Web page](#) is called **TLS-secured** if the top-level resource and all other resources that can affect or control the page's content and presentation have been retrieved through [strongly TLS protected HTTP connection transactions](#).]

This definition implies that inline images, stylesheets, script content, and frame content for a secure page need to be retrieved through [strongly TLS protected HTTP transactions](#) in order for the overall page to be considered TLS-secured.

5.5 Change of security level

This section and its subsections are normative. Examples are informational.

[Definition: A **change of security level** occurs when actual interactions do not match expectations established by user context, interaction history, or user agent state.] This section elaborates on the conditions under which such changes should be considered to have occurred.

Implementations **MAY** assume that a change of security level has occurred for reasons not covered in this

specification.

5.5.1 Certificate errors

If an attested certificate is shown in a TLS-protected interaction, and strong TLS protection cannot be achieved, then Web user agents MUST assume that a change of security level has occurred:

1. if the URL that is dereferenced does not match the subject certificate
2. if path verification (Basic or Relaxed) fails

5.5.2 Redirection chains

When users follow hyperlinks, user agents are often presented with a chain of redirections, maybe based on HTTP codes, maybe based on scripting. In some situations, the [re-directed to](#) link might be located on a [TLS-secured page](#), yet, the chain of redirects involves plain HTTP, or [weakly TLS-protected](#) HTTP transactions, ~~thereby leading to. If the user were following these re-redirects themselves, this would lead to~~ additional exposure, and [in particular](#) ~~ty~~ [result in](#) high potential of user confusion.

Web user agents MUST assume that a [change of security level](#) has occurred when a user interaction with a [TLS-secured page](#) that causes dereferencing of a URL leads to a chain of Web transactions that:

- does not involve user interactions
- involves [weakly TLS-protected](#) or unprotected HTTP transactions.
- terminates on a Web Site different from the one with which the user interacted to initiate the chain of redirections [\[tjh: How is “different web site” defined here? It seems like this one would make re-redirects based on legitimate corporate acquisitions and mergers problematic.\]](#)

Note that this applies whether or not the resource in which the non-interactive chain of redirections terminates is TLS protected in any manner. In particular, even if the retrieval of the final resource in the chain of redirections is [strongly TLS protected](#), clients MUST assume that a change of security levels has occurred. Also note that this section is not limited to HTTP level redirection mechanisms; it also covers redirections that are caused by scripting or HTML constructs.

Also note that this section does not apply to situations in which, e.g., an HTML form is served by way of a [strongly TLS protected transaction](#), but the user's input is submitted through plain HTTP. [\[tjh: Where is this situation documented?\]](#)

5.5.3 Change against historical practice

Web user agents that have found a resource [strongly TLS protected](#) during past interactions MUST consider an interaction with the same resource as a [change of security level](#) if that interaction is [now](#) not [strongly TLS protected](#). Web user agents that have found a resource [strongly TLS protected](#) with an [Augmented Assurance Certificate](#) SHOULD consider an interaction with the same resource as a [change of security level](#) if that interaction is not [strongly TLS protected](#) with an [Augmented Assurance Certificate](#) or is [strongly TLS protected but without an Augmented Assurance Certificate](#).

Examples of such situations include a site that usually presents a certificate issued by a trusted Certification Authority, and now changes to presenting a certificate from an unknown CA, a self-signed certificate [during the probation period](#), or a no-interaction certificate; or a site that changes self-signed certificates.

The current language is setting the stage for rather intrusive error handling when a site changes self-signed certificates from ones that a client is used to. This might be undesirable. [ISSUE-114](#)

5.5.4 Explicit user expectation

When an user manually enters a `https` scheme URL, this indicates an expectation of [strongly TLS-protected](#) behavior. If the resulting transaction is not [strongly TLS-protected](#) (e.g., a [no interaction certificate](#) is presented), a [change of security level](#) is considered to have occurred.

6 Indicators and Interactions

6.1 Identity and Trust Anchor Signalling

Please refer to the following entries in the Working Group's Wiki for relevant background information: [IdentitySignal](#)

This section specifies practices for signalling information about the identity of the Web site a user interacts with. All signals specified in this section are passive. No claim is made about the effectiveness of these signals to counter impersonation attacks.

6.1.1 Identity Signal

This section is normative. Examples are informational.

Web user agents MUST make information about the `[[identity]]` of the Web site that a user interacts with available [to users](#). This [Definition: `[[identity signal]]`] SHOULD be part of [primary user interface](#) during usage modes which entail the presence of signalling to the user that is different from solely page content. Otherwise, it MUST at least be available through [secondary user interface](#). Note that there may be usage modes during which this requirement does not apply: For example, a Web browser which is interactively switched into a no-chrome, full-screen presentation mode need not preserve any security indicators in primary user interface.

User interactions to access this [identity signal](#) MUST be consistent across all Web interactions, including interactions during which the Web user agent has no trustworthy information about the `[[identity]]` of the Web site that a user interacts with. In this case, user agents SHOULD indicate that no information is available.

User agents with a visual user interface that make the identity signal available in [primary user interface](#) SHOULD do so in a consistent visual position.

6.1.2 Identity Signal Content

Please refer to the following entries in the Working Group's Wiki for relevant background information:
[RecommendationDisplayProposals/Letterhead](#)

This section is normative. Examples are informational.

Information displayed in the [identity signal](#) MUST be derived from [attested certificates](#), from user agent state, or be otherwise authenticated. Web user agents MUST NOT use information as part of the [\[\[identity signal\]\]](#) that is taken from unauthenticated or untrusted sources.

During interactions with a [TLS-secured Web page](#) for which the top-level resource has been retrieved through a [strongly TLS-protected](#) interaction that involves an [augmented assurance certificate](#), the [identity signal](#) MUST include the Subject field's Organization attribute to inform the user about the owner of the [Web page](#).

During interactions with a [TLS-secured Web page](#) for which the top-level resource has been retrieved through a [strongly TLS-protected](#) interaction that involves an [attested certificate](#), an applicable domain name label retrieved from the [certificate](#) subject's Common Name attribute or from a subjectAltName extension MUST be displayed.

The [certificate's](#) Issuer field's Organization attribute MUST be displayed to inform the user about the party responsible for that information.

During interactions with a Web page for which any of the resources involved was retrieved through a [weakly TLS-protected](#) transaction, the [identity signal](#) must be indistinguishable from one that would be shown for an unprotected HTTP transaction, unless a [change of security level](#) has occurred.

For Web user agents that use a visual user interface capable of displaying bitmap graphics, during interactions with a [TLS-secured Web page](#) for which the top-level resource has been retrieved through a [strongly TLS-protected](#) interaction that involves an [extended validation certificate](#), the [identity signal](#) ~~[[MAY]]~~ SHOULD include display of an [\[\[issuer | community | subject\]\]](#) logotype that is embedded in the certificate using the logotype extension [\[RFC3709\]](#).

During interactions with pages that were (all or in part) retrieved through [weakly TLS-protected](#) interactions, [Web user agents](#) MUST NOT display any logotypes derived from certificates [even if those certificates contain logotype information](#).

For discussion concerning the open questions regarding logotypes in this section, see [ISSUE-96](#) and [ISSUE-98](#).

6.2 Additional Security Context Information

Please refer to the following entries in the Working Group's Wiki for relevant background information:
[PageInfoSummary](#)

This section is applicable to [secondary chrome](#) alone and will NOT affect user behavior if they do not drill down to it. Studies [\[WHALENEVIDENCE\]\[XIA\]](#) have indicated that users are less likely to

examine secondary chrome or take proactive steps to look for contextual information.

This section is normative.

[Web user agents](#) MUST provide additional security context information as described in this section through one or more user-accessible information sources. These information sources can be implemented in either [primary](#) or [secondary](#) user interface. Where security context information is provided in both primary and secondary chrome, the presentation and semantics of the presented information MUST be consistent.

The information sources MUST make the following security context information available:

1. the Web page's domain name
2. Owner information, consistent with [6.1.2 Identity Signal Content](#)
3. Verifier information, consistent with [6.1.2 Identity Signal Content](#)
4. The reason why the identity information is trusted, e.g., the fact that a certificate is [qualified to attest](#) identity information.

The information sources SHOULD make the following security context information available:

1. Whether a Web page is TLS-protected, whether the protection is weak or strong, and the reasons for the value of the protection.
2. When the Web page is TLS-protected and an [attested certificate](#) was used, whether or not a certificate status check has been performed.
3. If a certificate status check has been performed, what the result was.
4. Whether the user has visited the site in the past.
5. [Whether there are changes in the security information since the last time the user visited this site.](#)
6. Whether the user has shown credentials to this site.
7. Whether the user has stored credentials for this site.
8. Whether the site content was encrypted in transmission.
9. Whether the site content was authenticated [\[tjh: define “authenticated” in this sentence\]](#).
10. For user agents capable of displaying graphical material, any logotypes present in logotype extensions present on augmented assurance certificates.

RFC 3709 also specifies a format to encapsulate audio data with certificates. Not yet covered. [ISSUE-120](#)

Additionally, the information sources MAY make the following security context information available:

No material here, yet?

User agents that provide information about the presence or absence of Cookies [\[RFC2965\]](#) MUST NOT make any claims that suggest that the absence of cookies implies an absence of any user tracking, as there are numerous tracking and session management techniques that do not rely on Cookies.

[6.3 Page Security Score](#)

See also: [ISSUE-129](#)

Please refer to the following entries in the Working Group's Wiki for relevant background information: [RecommendationDisplayProposals/PageScore](#)

The user agent **MUST** reduce the state of all security context information made available to a single value. A partial order **MUST** be defined on the set of possible values.

The user agent **MUST** make the security context information value available to the end user, in either primary or secondary chrome.

The user agent **MUST** make the formula by which the value is calculated available to the end user. Documentation of the user agent is the likeliest place.

The form of the indicator of this value will depend on the user agent and end user abilities. The user agent **SHOULD** provide a primary chrome indicator.

6.4 Error handling and signalling

Please refer to the following entries in the Working Group's wiki for background information: [CertErr](#)

Web agents regularly encounter error conditions during TLS interaction that alter the risk profile and security context of a particular resource. Due to the highly technical nature of these errors, the details are generally useful only to advanced users. For the majority of web users the technical details actually lead to confusion and risky behavior. For most users the overall security context should be adjusted intuitively while hiding the details of the error condition.

This section specifies error handling and signalling requirements for situations in which the use of TLS protection was [desired](#), but could not be achieved. The basic approach of this section is to minimize [end user interaction at the point in time when this situation arises](#).

6.4.1 Basic error handling interactions

This section is normative.

If [weak TLS protection](#) is achieved, but no [change of security level](#) has ~~occured~~[occurred](#), Web user agents **SHOULD NOT** give error indications that interrupt the user's flow of interaction. If a change of security level has ~~occured~~[occurred](#), Web user agents **SHOULD** ~~interrupt~~[interrupt](#) the flow of interaction and provide the user with a non-technical explanation that a security-related error has ~~occured~~[occurred](#). Web user agents **SHOULD** include the ability to access a description of the error for an expert audience. Web user agents **SHOULD** provide additional technical information (including the usage of technical jargon) about the error in a secondary chrome. Web user agents **SHOULD NOT** refer the user to enter the destination site to provide feedback or obtain assistance. [Web user agents MAY keep a history of such errors so that advanced users can review security related errors which occurred](#).

The error handling interaction **SHOULD** enable the user to easily return to the user agent state prior to initiation of the last user interaction that led to the change of security level.

For Web user agents with visual user interfaces, error interactions that interrupt the flow of interaction **SHOULD** visually replace [\[tjh: replace with what? Is a better term here “restore”?\] the Web content with](#)

which the user interacted prior to the user interaction that led to the error condition.

6.4.2 Handling certain man in the middle attacks

See also [7.9 Detecting a possible Man-In-The-Middle attack](#). Discussion about [ISSUE-127](#) is relevant to this section.

This section is normative.

If a server shows a certificate that would lead to a [strongly TLS protected](#) interaction, but whose subject information does not match the URL of the current interaction, and if it is possible to construct a request URL from certificate information, then Web user agents MAY offer users the additional possibility to follow this URL.

Web user agents MUST obey the following constraints:

1. In constructing the URL from the certificate, information concerning the user's originally desired interaction MUST NOT be used.
2. The user interaction MUST explicitly mention the certificate subject's organization name.
3. The user interaction MUST trigger a safe HTTP method [\[RFC2616\]](#).

7 Safe Web Form Editor

For background material, see the Working Group's wiki: [SafeWebFormEditor](#)

Some of the material in this section needs to be reconciled with other parts of the specification, in particular [6.4 Error handling and signalling](#).

It is an open issue whether the specification of an interaction mode like the Safe Web Form Editor should be specifically limited or tailored to login transactions, or whether a more general approach is preferable. [ISSUE-111](#)

This section is normative.

This section specifies a user-driven interaction for entry of information deemed security critical by the user. This interaction integrates consumption of relevant security information by the user while facilitating the data entry task and providing a record of the user's trust decisions.

7.1 Associating a history with a web site

The safe form editor keeps a history of text strings the user has [given assigned](#) to a Web site via a special purpose text entry tool.

This section specifies the algorithm the user agent MUST use to determine if a visited web site should be considered the same as one in the history database. This algorithm is based solely on a comparison of information provided by X.509 certificate chains. Let the currently visited web site be *SiteA*, a candidate match in the stored editor bar history be *SiteB*. For each certificate chain received from *SiteB*, attempt a match against the one received from *SiteA* using the following checks:

1. If both *SiteA*'s and *SiteB*'s certificates are currently valid and they specify the same public key, there is a match; otherwise, continue with the matching algorithm.
2. If *SiteA*'s certificate was issued by a different certificate authority than *SiteB*'s certificate, there is no match; otherwise, continue with the matching algorithm. Two certificate authorities are considered the same if their certificates are identical, or if they are both installed as trusted certificate chain roots identified by the same name in the user agent's presentation to the user.
3. If both *SiteA*'s and *SiteB*'s certificates have the same value for the Subject field's Common Name (CN) attribute, there is a match; otherwise, continue with the matching algorithm.
4. If both *SiteA*'s and *SiteB*'s certificates have the same values for all of the "O", "L", "ST" and "C" attributes of the Subject field, there is a match; otherwise, continue with the matching algorithm.
5. The algorithm ends here with no match.

This matching algorithm needs to be reconciled with PKIX, and with material elsewhere in this specification. See [ISSUE-121](#) for a more detailed discussion of some of these aspects.

The user agent MUST retain sufficient information about all the certificate chains used by a web site to find a match in all cases where the above algorithm indicates there is a match.

The first check in the matching algorithm, which compares public keys, provides a means to transparently update the certificate authority used by a web site. To change certificate authorities, a site acquires a certificate for its existing public key from the new certificate authority. The site should continue using its existing public key until its user base has received the new certificate chain through visits to the site.

Both the first check in the matching algorithm and the second to last, which compares the "CN" attributes of the certificates' subject fields, provide a means to transparently update an organization's name and address. To change this certificate information, an organization acquires a certificate chain that specifies the updated information, but matches against one of these earlier checks.

The above paragraph makes assumptions about CA practices. See [ISSUE-122](#).

It is common for an organization to use multiple, unrelated domain names. The final check in the matching algorithm enables sharing of history across all the hostnames used by an organization.

7.2 No support for non-TLS data exchange

The editor bar supports safe entry of text strings by the user into a web site with which a continuous relationship has been established. The user can expect this continuity to be securely enforced and the transmissions protected from eavesdropping and tampering. Currently, these properties can only be supported on the web through the use of TLS. The editor bar MUST NOT be enabled for exchanges that are not protected by TLS. If the user hits the editor bar attention key when visiting a site not protected by [SSL/TLS](#), the user agent MUST present a message warning the user that data entered in the current form may be seen, or tampered with, by attackers. The user agent MUST offer an interaction to attempt navigating to a secure version of the current page. Exercising that interaction MUST navigate the browser to a URL constructed by changing the current page's URL scheme to a corresponding one that

uses TLS. For example, an http: URL is converted to an https: URL.

Example [Informational message to the user:](#)

This web page does not support secure information exchange, and so the editor bar cannot be used here. Information entered into this web page may be seen, or tampered with, by others. Click here to see if the web site supports a secure version of this web page. [<Button: See if Site Supports>](#) [<Button \(default\): Avoid This Site>](#)

The current text assumes that it is always possible to construct a safe and meaningful interaction that involves the https version of a URL. [ISSUE-123](#)

7.3 Creating a new relationship

When visiting a web site, the user summons the editor bar via an attention key, or a toolbar button, or in some other way. In response to this request, the editor bar searches its history database for an entry corresponding to the current web site, using the algorithm specified in [7.1 Associating a history with a web site](#). If no match is found, the text entry tool MUST NOT be enabled. Instead, the user agent MUST present a message that indicates that the user has not transmitted sensitive data to this site before. Along with this message, the user agent MUST offer distinct interactions through which (a) the user can review sites that they have transmitted sensitive data to before, and (b) the user can proceed to establish a relationship with the site they are visiting.

Example:

You have not established a relationship with this web site through the editor bar.

[\(a\)](#) You might not be at the right web site. Click here for a list of web sites you have established a relationship with.

[\(b\)](#) If you know you haven't established a relationship with this web site, and you want to do so, click here.

If the user chooses interaction (a), a list of hyperlinks to web sites in the editor bar's history database MUST be provided.

If the user chooses interaction (b), the user agent must provide a message which communicates identity information based on the TLS certificate used consistent with [6.1 Identity and Trust Anchor](#)

Signalling. The user agent MUST offer distinct interactions through which [\(a\)](#) the user can navigate to a known preferred search engine, and [\(b\)](#) the user can enter a ["petname" personally chosen name \(personal alias, nickname, reminder\)](#) for the site they are interacting with.

Example:

Example CA Inc. claims this web site belongs to "Example Bank Inc. of Sunnydale, California, USA."

[\(a\)](#) If this identification does not match the one you were expecting, click here to search for the organization you were intending to contact.

[\(b\)](#) If this identification is acceptable for you, enter a name [of your choice that](#) the editor bar will

use to refer to this web site: <text field>

If the user chooses interaction (aa), the user agent MUST navigate to the user's preferred search engine.

If the user selects the second option, the user agent MUST search the database of existing relationships to find any name similar to the newly chosen one. If any matches are found, the user is notified of the collisions and given the opportunity to instead navigate to the corresponding website. If no matches are found, the user agent updates its database of stored relationships and enables the text entry tool.

For user agents with a visual user interface, all user interfaces exposed for these interactions SHOULD visually replace the Web content currently displayed to the user.

7.4 Reliable Text

This section is phrased in terms of visual user interfaces, and includes normative back-references to some of the examples earlier in the specification. [ISSUE-124](#)

To defend against web site impersonation, the editor bar is designed to only display text strings provided by the user, or statically provided by the user agent software. Implementations MUST NOT display a text string, or graphic, from any other source within the editor bar user interface. The quoted text in the bootstrap [\[tjh: I think there needs to be a better term for this than “bootstrap”\]](#) message, the part of the message after "belongs to", MUST be distinguished from the rest of the static text. If the described certificate chain was not issued by one of the user-agent's installed and trusted certificate authorities, the message MUST NOT quote any information from the certificate, instead presenting a message meaning: "This web site's credentials are unrecognized". The same two user actions are still offered by the rest of the message.

The name chosen by the user at the end of the bootstrap interaction, called a [\[tjh: I suggest using a different term for this document. I suggest adding a reference to the petname work and indicating that this portion of the recommendation is attributed to that work.\]](#) [Definition: **petname**], MUST be the only identifier used by the editor bar user interface to refer to the named site. Each hyperlink in the list provided when the user selects the first option in the first message of the bootstrap interaction MUST use the petname as the hypertext. This list MUST also be accessible to the user via a "Contacts" option. The petname for the current web site MUST also be presented alongside the text entry tool. The user agent MUST provide a means for the user to update the petname used for a web site.

7.5 Selection of a text string

The text entry tool supports user entry of a new text string, or auto-completion of a previously entered text string. The editor MUST provide an indication of which of the two actions is being taken. The text field MUST NOT provide auto-completion of stored editor bar text strings that have not been previously submitted to the currently displayed web site.

The text entry tool history menu supports user selection of a previously entered text string. The menu MUST indicate whether or not an offered selection has been previously submitted to the currently

displayed web site. Further, the user action to select a text string previously submitted to the current site **MUST** be different from that to select a text string previously submitted to some other site. For example, text strings previously submitted to the current site could be displayed in a main menu and other text strings displayed in a sub menu. Consequently, the user would have to click more than once to select a string not previously submitted to the current site, but only once for a string that has been previously submitted.

Both of these selection mechanisms purposely require explicit action by the user. Transmission of a text string in a particular request represents user consent for use of that text string for the purpose of that request. The user agent **MUST NOT** subvert this consent by auto-filling form fields with information taken from the editor bar history. Information **MUST NOT** be transferred from the history database to the web page, except as a result of an explicit approval action by the user.

The editor bar specified in this chapter **MUST** be the only form filling feature of the user agent. A competing form filling feature would undermine the security features of the interaction created by the editor bar.

7.6 On screen masking of a text string

This section is phrased in terms of visual user agents. [ISSUE-125](#)

Some text strings, such as some passwords, are of such high value that displaying them within the user agent, where they may be seen by an onlooker, is too great a risk. This sub-section specifies a mechanism for marking a text string as one which should not be displayed by the user agent.

The text entry tool history menu **MUST** provide a means for the user to mark a text string as one which is not to be displayed on screen. Invoking this command prompts the user for a "display name". Wherever a text string would be displayed by the editor bar, the provided display name **MUST** be shown in its place, as well as an indication that the displayed text is a display name, rather than an actual text string. The auto-completion feature of the text entry tool **MUST** match keystrokes against the display name, instead of the named text string. Whatever way a display name is selected, the named text string **MUST** be form filled, not the display name text.

7.7 Editing the stored history

Each change to the editor bar history **MUST** be explicitly made by the user. In particular, a visited web site **MUST NOT** be able to add, delete, modify or read the editor bar history. A visited web site **MUST NOT** be able to receive keystrokes sent to the editor bar by the user.

Some identifiers change over time. For example, a credit card number may change when the card is re-issued. The editor bar **SHOULD** provide a means to delete a text string. Using this feature, the user can reduce interface clutter created by a text string that is no longer in use.

[\[tjh: I suggest adding text which allows for the usage of a "card" metaphor rather than only discussing a form editor/filler capability.\]](#)

Under some circumstances, a relationship between the editor bar and a site may need to be terminated. The editor bar **MUST** provide a means for a user to mark a site as one which will no longer be recognized. When this command is invoked, the editor bar **MUST** behave as if the relationship never existed, for all scenarios specified by this specification.

7.8 Picture-in-picture defense

"Picture-in-picture attack" is undefined. [ISSUE-126](#).

Many graphical user agents are vulnerable to picture-in-picture attacks. In these user agents, the editor bar **MUST** be displayed using a theme customized to the user. The user selects this theme at browser installation time and it remains forever the same. The icon for the Contacts button **MUST** also be selected by the user at installation time.

7.9 Detecting a possible Man-In-The-Middle attack

See also [6.4.2 Handling certain man in the middle attacks](#). [ISSUE-127](#) tracks a number of separate issues with this section.

If the user navigates to a web site by selecting a hyperlink provided by the Contacts button in the editor bar and the site presents a certificate chain that can not be matched to one of its previously stored certificate chains according to the algorithm in [7.1 Associating a history with a web site](#), the user agent **MUST** alert the user that credentials were provided which cannot be securely matched to those that had been seen in the past.

User agents **SHOULD** provide the option to send a notification to an alert service of the user agent's choosing.

Example:

This web site is presenting credentials which cannot be securely matched to those provided in the past. You should not continue with your current task until this error has been corrected. Click here to report this error to the web site's administrator.

[The attempt MAY be logged \(iff the user agent is configured as such\) so that an advanced user may review the failure at a later time.](#)

8 Robustness

8.1 Do not mix content and security indicators

Please refer to the following entries in the Working Group's Wiki for relevant background information: [FavIcon](#),

Certificates can be a source of untrusted information that is controlled by the attacker. [ISSUE-104](#)

We currently have no material concerning the mixing of security information and context in non-visual

environments. Is there a useful generalization of the requirement to non-visual UIs? Are there problematic known cases similar to the location bar favicon mix known for, e.g., screen readers? [ISSUE-115](#)

To the extent to which users pay attention to passive security indicators at all, noticing and understanding them is made difficult to impossible when the same signal path that is commonly used for security indicators can also be controlled by Web content. For example, the location bar commonly found on desktop browsers is often used to both display the "padlock" security indicator, and a possible "favorite icon" [\[FAVICON\]](#), which can in turn be a simple copy of the very padlock an informed and attentive user might look for.

This section includes requirements and techniques to avoid mixing of Web content and security indicators, including -- but not limited to -- the favorite icon case.

8.1.1 Conformance note

See [ISSUE-112](#).

This section is normative.

Some Requirements and Techniques in this section are phrased in terms of "parts of the user interface that are intended or commonly used to communicate trust information to users." This concept captures that users' interaction habits are often formed by interactions with a number of Web user agents that may be using different paths for signalling of security context information. Therefore, beyond showing implementation of all NECESSARY Techniques (or at least one SUFFICIENT Technique), conformance claims with the Requirement captured in this section MUST be supported by experimental studies to determine what these parts of the user interface are.

8.1.2 Requirement

This section is normative.

Web User Agents MUST NOT display material controlled by Web content in parts of the user interface that are intended or commonly used to communicate trust information to users.

This requirement applies to both [primary](#) and [secondary](#) elements of visual user interfaces.

8.1.3 Techniques

This section is normative.

The following techniques are NECESSARY. They MUST be implemented by a conforming Web user agent.

See also: [ISSUE-109](#)

- Web User Agents MUST NOT display favorite icons [\[FAVICON\]](#) within the visual context of a [Location Bar widget](#), if present.

- Web User Agents MUST NOT display favorite icons [\[FAVICON\]](#) in [secondary user interface](#) intended to enable users' trust decisions.

The following technique is neither necessary nor sufficient to claim conformance with the Requirement. However, conformance with this technique entails conformance with the necessary techniques that concern favorite icons.

- Web User Agents MAY ignore favorite icon [\[FAVICON\]](#) references that are part of Web content.

[8.2 Establishing a trusted path between users and browsers](#)

Section to be done; expected to hold material along the lines of [RobustSharedSecret](#).

[8.3 APIs exposed to Web content](#)

This section does not yet cover bookmark-related APIs that are exposed to Web content. [ISSUE-95](#)

User agents commonly allow web content to perform certain manipulations of agent UI and functionality such as opening new windows, resizing existing windows, etc. to permit [web content-based](#) customization of the user experience. These manipulations need to be properly constrained to prevent malicious sites from concealing or obscuring important elements of the browser interface, or deceiving the user into performing dangerous acts. This section includes requirements and techniques to address known attacks of this kind.

[8.3.1 Requirements](#)

This section is normative.

- Web user agents MUST prevent web content from [masquerading](#), obscuring, hiding, or disabling the security UI ([secondary UI](#)).
- Web user agents MUST NOT expose programming interfaces which permit installation of software, or execution of privileged code without user intervention.

[8.3.2 Techniques](#)

This section is normative.

- Web user agents SHOULD restrict window sizing and moving operations to the visible desktop, where applicable. This prevents attacks wherein browser chrome is obscured by moving it off the edges of the visible screen.
- Web user agents SHOULD NOT allow web content to open new windows with the browser's security UI hidden. Allowing this operation facilitates picture-in-picture attacks, where artificial chrome (usually indicating a positive security state) is supplied by the web content in place of the hidden UI.
- Web user agents MUST inform the user and request consent when web content attempts to install

or execute software outside of the browser environment.

- When informing users of this event, web user agents **MUST** employ a user interface which prevents immediate click through (e.g. with a briefly disabled OK button.) This prevents click-through and [whack a mole attacks](#) where users are encouraged by nuisance elements to continually click in a given location.
- Web user agents **SHOULD** use difficult-to-spoof UI elements that cross the chrome-content border where appropriate.
- Web user agents **MUST** prevent web content from overlaying chrome.
- Web user agents **MAY** restrict the opening of pop-up windows from web content, particularly those not initiated by user action. Creating excessive numbers of new popup windows is a technique that can be used to condition users to rapidly dismissing dialogs. This can be employed in ["whack-a-mole" attacks](#) as mentioned above.
- Web user agents which offer this restriction **SHOULD** offer a way to extend permission to individual trusted sites. Failing to do so encourages users who desire the functionality on certain sites to disable the feature universally.

9 Authoring and deployment best practices

Material concerning HTTP to HTTPS form submissions will be added to this section. [ISSUE-107](#)

9.1 Do not put Security Indicator images to indicate trust in content

This specification requires that web pages **MUST NOT** include trust indicating images such as padlocks in the web content.

9.2 Use TLS for Login Pages

An unsecure web page **MUST NOT** embed a form meant for the user to login. All login pages **MUST** be served from secure servers ie. login pages must be [TLS protected](#). An unsecure page **MAY** carry a link for the user to click to be taken to a secure page to enter login information. This link **MUST NOT** [carrydisplay](#) a padlock along with it.

9.3 Use TLS Consistently across the web site

If a web site requires [secure](#) login ([and, by section 9.2, a secure login](#)), then all sensitive transactions and presentation [which commences](#) based on the user [having authenticated with some's](#) credentials, as well as the service provided credential token itself **MUST** be protected by the same level of security. [This includes information such as cookies established during the authenticated interaction.](#) Cookies on unsecure connections are vulnerable to interception, and can be used for replay attacks even if they were set by a secure server, and servers **MUST NOT** set credential cookies from secure servers that can be sent unencrypted.

9.4 Redirects from Non-Secure to Secure page and vice-versa

Web Sites that require their users to be redirected from an unsecure web page to a secure web page MUST do it as a single step rather than multi-step (redirect to an unsecure page and then redirect again to a secure page). This specification recommends that the web page MUST use direct links to a secure page rather than using redirects.

[\[tjh: proposed new section: 9.6 Security Settings across user agents](#)

[Discuss the ability to export/import or share security settings/configuration between different user agents \(either on the same system or on different systems and form-factors\). \]](#)

10 Usage Modes

10.1 Framework

Please refer to the following entries in the Working Group's Wiki for relevant background information:
[BrowserLockDown](#)

This section is normative.

Web user agents that implement optional features of this specification MUST support the configuration of different [Definition: **usage modes**] which determine which of ~~these~~[these](#) optional features are active. A usage mode MAY cover other configuration settings of a Web user agent. A user agent SHOULD allow users to view details of why a request to perform a Web transaction was denied if this decision was based on the currently-active usage mode.

[This recommendation is applicable to any user agent which contains configurable settings for security-related processing. The end result of an implementation meeting this requirement would be a user agent which supports the specification of multiple "usage modes" and operates according to the "usage mode" which has been selected for the user's current interaction type \(or site the user is visiting\). While in a particular "usage mode", a user agent would not allow the user to stray outside of this configuration. There would be no prompting to accept or avoid access to a site - the configuration settings indicated by the "usage mode" would indicate what the choice should be. There would be no dialogs presented to the user for configuration - ideally a starter set of "well known" or "known to be trusted" configurations or "usage modes" could be provided during initial installation.](#)

[For isolated deployments, an organization could choose to deploy a set of "usage modes" such that their organization's users would use the user agent only according to the organization's recommended/mandatory terms of use \(as defined by the "usage mode" defined to the user agent by some administrative entity\).](#)

Requirement

A user agent MUST support a mode of operation whereby the user is unable to view or modify the security-related configuration settings.

A user agent MUST support the configuring of different "usage modes" and allow the specification of which "usage mode" is to be the active "usage mode".

A user agent SHOULD allow users to view details of why a request or access to a site was blocked based on "usage mode" in effect. This MAY also include a description of which configuration setting or settings contributed to the site being blocked (but displayed only on request).

A user agent SHOULD keep a history of blocked sites and the reasons for the block to allow analysis of these at a later date and time (and possibly by a person other than the user).

A user agent SHOULD support a "admin/service" mode in which "usage modes" defined and changed.

Techniques (non-normative)

Implementations may choose to support a multiple groups of configuration settings called profiles. These profiles may then be associated with different "usage modes". Implementations may allow a user to act as both an administrator of the user agent and a user of the user agent as a means of transitioning users from current status quo to fully locked down processing.

Organizations and social networking sites could build collaborative works to offer "well tested" or "popular" configuration profiles.

Which profile is employed could be made a dependent upon the site being visited. Site-specific behavior may be done based on rules about the security indicators the site is providing or based on white-list/black-list type lookups.

10.2 Safe Browsing Mode

Please refer to the following entries in the Working Group's Wiki for relevant background information:
[SafeWebBrowsingTemplate](#)

Web user agents SHOULD support a [Definition: **Safe Browsing Mode**] as one usage mode. Users SHOULD NOT be able to change the settings of this usage mode. This usage mode MUST be made available through an interaction based on a Secure Attention Sequence.

In this usage mode, interactions MUST be limited to a usage-mode specific set of sites.

Should Safe Browsing mode restrict users to a specific set of sites? [ISSUE-108](#)

Web user agents MUST require all Web transactions in this usage mode to be [strongly TLS protected](#). Use of self-signed certificates MUST be considered cause for a [change of security level](#).

The optional technique [6.4.2 Handling certain man in the middle attacks](#) MUST NOT be used.

For Web user agents with a visual user interface, Safe Browsing Mode SHOULD be visually distinguishable, [i.e. Distinctly different in chrome](#), from other usage modes.

11 Security Considerations

11.1 Active attacks during initial TLS interactions

6.4 Error handling and signalling leads to an additional exposure during the *first* TLS interaction with a site, even if that site uses attested or extended validation certificates: An active attacker can show a self-signed certificate, which will cause no warning signals to the user, except for differences in a passive security indicator (see **6.1 Identity and Trust Anchor Signalling**). Traditional web user agents react to this scenario with a dialogue box that interrupts the user's flow of interaction, but gives users the ability to override the security warning. Empirical evidence shows that this ability is typically exercised by users.

Countermeasures against this threat include the prior designation of high-value sites, for which extended validation or attested certificates are required (causing a change of security level during the attack scenario described above), and communication of certification and TLS expectations by a mechanism separate from HTTP, e.g. through authenticated DNS records.

11.2 Using error handling interactions as vectors for user deception

6.4.2 Handling certain man in the middle attacks relies on the assumption that the information communicated to the user about the site that the user really connects to is reasonably trustworthy, and will not be used to deceive the user. If this assumption is broken, an attacker may be able to deceive users about the fact that a serious attack occurred.

11.3 Secure storage of sensitive information

Like the password managers in today's user agents, the editor bar specified in **7 Safe Web Form Editor** requires storage of information whose confidentiality must be protected. Implementors should use secure storage techniques that ensure the editor bar's database is only accessed by the user. Specifying appropriate storage techniques is beyond the scope of this recommendation.

[tjh: proposed new section 11.4 View/modify of Security settings

Perusal/scanning/or viewing of security settings in the user agent, by web-delivered content MUST be avoided/severely restricted.]

12 References

ECRYPT2006

[*ECRYPT Yearly Report on Algorithms and Key Lengths \(2006 Edition\)*](#), available at <http://www.ecrypt.eu.org/documents/D.SPA.21-1.1.pdf>.

EVSSLCERT

Guidelines for the Issuance and Management of Extended Validation Certificates, CA/Browser Forum, 7 June 2007, available at http://www.cabforum.org/EV_Certificate_Guidelines.pdf.

FAVICON

How to Add a Favicon to your Site, D. Hazaël-Massieux, C. Lilley, O. Théreaux, W3C work in progress, available at <http://www.w3.org/2005/10/howto-favicon>.

NESSIE

Portfolio of recommended cryptographic primitives, New European Schemes for Signatures, Integrity, and Encryption (NESSIE), available at <https://www.cosic.esat.kuleuven.be/nessie/deliverables/decision-final.pdf>.

RFC2119

Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, S. Bradner, March 1997, available at <http://www.ietf.org/rfc/rfc2119.txt>

RFC2616

Hypertext Transfer Protocol -- HTTP/1.1, RFC 2616, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, June 1999, available at <http://www.ietf.org/rfc/rfc2616.txt>

RFC2817

Upgrading to TLS Within HTTP/1.1, RFC 2817, R. Khare, S. Lawrence, May 2000, available at <http://www.ietf.org/rfc/rfc2817.txt>

RFC2818

HTTP Over TLS, RFC 2818, E. Rescorla, May 2000, available at <http://www.ietf.org/rfc/rfc2818.txt>

RFC2965

HTTP State Management Mechanism, RFC 2965, D. Kristol, L. Montulli, October 2000, available at <http://www.ietf.org/rfc/rfc2965.txt>

RFC3280

Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 3280, R. Housley, W. Polk, W. Ford, D. Solo, April 2002, available at <http://www.ietf.org/rfc/rfc3280.txt>

RFC3281

An Internet Attribute Certificate Profile for Authorization, RFC 3281, S. Farrell, R. Housley, April 2002, available at <http://www.ietf.org/rfc/rfc3281.txt>

RFC3709

Internet X.509 Public Key Infrastructure: Logotypes in X.509 Certificates, RFC 3709, S. Santeson, R. Housley, T. Freeman, February 2004, available at <http://www.ietf.org/rfc/rfc3709.txt>

RFC3986

Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, T. Berners-Lee, R. Fielding, L. Masinter, January 2005, available at <http://www.ietf.org/rfc/rfc3986.txt>

RFC4254

The Secure Shell (SSH) Connection Protocol, RFC 4254, T. Ylonen, C. Lonvick, January 2006, available at <http://www.ietf.org/rfc/rfc4254.txt>

RSA-SIZES

TWIRL and RSA Key Size, Burt Kaliski, RSA Laboratories, revised 6 May 2003, available at <http://www.rsa.com/rsalabs/node.asp?id=2004>.

WCAG20

Web Content Accessibility Guidelines 2.0, B. Caldwell, M. Cooper, L. G. Reid, G. Vanderheiden, W3C Working Draft 17 May 2007. This version of WCAG 2.0 is work in progress. This version is <http://www.w3.org/TR/2007/WD-WCAG20-20070517/> . The [latest version](#) of WCAG 2.0 is available at <http://www.w3.org/TR/WCAG20/> .

WHALENEVIDENCE

Use of Visual Security Cues in Web Browsers, T. Whalen and K. M. Inkpen. Gathering Evidence. In Proceedings of the 2005 Conference on Graphics Interface, pages 137–144, Victoria, British Columbia, 2005.

WSC-THREATS

Web User Interaction: Threat Trees, T. Roessler, Editor, W3C Working Group Note (work in progress), 1 November 2007. This version is <http://www.w3.org/TR/2007/NOTE-wsc-threats-20071101/>. The [latest version](#) is available at <http://www.w3.org/TR/wsc-threats/> .

WSC-USECASES

Web Security Experience, Indicators and Trust: Scope and Use Cases, T. Close, Editor, W3C Working Draft (work in progress), 1 November 2007. This version is <http://www.w3.org/TR/2007/WD-wsc-usecases-20071101/>. The [latest version](#) is available at <http://www.w3.org/TR/wsc-usecases/> .

XIA

Hardening web browsers against man-in-the-middle and eavesdropping attacks, H. Xia and J. C. Brustoloni. In Proceedings of the 14th International World Wide Web Conference (WWW2005), pages 489–497. W3C/ACM, May 2005, available at <http://www2005.org/cdrom/docs/p489.pdf> .