Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2012

J. Hodges
PayPal
C. Jackson
Carnegie Mellon University
A. Barth
Google, Inc.

October 31, 2011

HTTP Strict Transport Security (HSTS)
draft-ietf-websec-strict-transport-sec-03

Abstract

This specification defines a mechanism enabling Web sites to declare themselves accessible only via secure connections, and/or for users to be able to direct their user agent(s) to interact with given sites only over secure connections. This overall policy is referred to as HTTP Strict Transport Security (HSTS). The policy is declared by Web sites via the Strict-Transport-Security HTTP Response Header Field, and/or by other means, e.g. user agent configuration.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

[Please disscuss this draft on the WebSec@ietf.org mailing list [WEBSEC].]

The HTTP protocol [RFC2616] may be used over various transports, typically the Transmission Control Protocol (TCP) [RFC0793]. However, TCP does not provide channel integrity protection, confidentiality, nor secure host identification. Thus the Secure Sockets Layer (SSL) protocol [I-D.ietf-tls-ssl-version3] and its successor Transport Layer Security (TLS) [RFC4346], were developed in order to provide channel-oriented security, and are typically layered between application protocols and TCP. [RFC2818] specifies how HTTP is layered onto TLS, and defines the Uniform Resource Identifier (URI) scheme of "https" (in practice however, HTTP user agents (UAs) typically offer their users choices among SSL2, SSL3, and TLS for secure transport). URIs themselves are specified in [RFC3986].

UAs employ various local security policies with respect to the characteristics of their interactions with web resources depending on (in part) whether they are communicating with a given web resource using HTTP or HTTP-over-a-Secure-Transport. For example, cookies ([RFC2109] and [RFC2965]) may be flagged as Secure. UAs are to send such Secure cookies to their addressed host only over a secure transport. This is in contrast to non-Secure cookies, which are returned to the host regardless of transport (although modulo other rules).

UAs typically annunciate to their users any issues with secure connection establishment, such as being unable to validate a TLS server certificate trust chain, or if a TLS server certificate is expired, or if a TLS server's domain name appears incorrectly in the TLS server certificate (see section 3.1 of [RFC2818]). Often, UAs enable users to elect to continue to interact with a web resource in the face of such issues. This behavior is sometimes referred to as "click(ing) through" security [GoodDhamijaEtAl05] [SunshineEgelmanEtAl09], and thus can be described as "click-through insecurity".

A key vulnerability enabled by click-through insecurity is the leaking of any cookies the web application may be using to manage a user's session. The threat here is that the attacker could obtain the cookies and then interact with the legitimate web application while posing as the user.

Jackson and Barth proposed an approach, in [ForceHTTPS], to enable web applications and/or users to declare that any interactions with the web application must be conducted securely, and that any issues

with establishing a secure session are to be treated as fatal and without direct user recourse. The aim is to prevent users from unintentionally downgrading their security.

This specification embodies and refines the approach proposed in [ForceHTTPS], i.e. instead of using a cookie it defines and uses an HTTP response header field, named "Strict-Transport-Security", to convey the site HSTS policy to the UA. This specification also incorporates notions from [JacksonBarth2008] in that the HSTS policy is applied on an "entire-host" basis: it applies to all TCP ports on the host. Additionally, HSTS policy can be applied to the entire domain name subtree rooted at a given host name. This enables HSTS to protect so-called "domain cookies", which are applied to all subdomains of a given domain.

1.1. Organization of this specification

This specification begins with an overview of the use cases, policy effects, threat models, and requirements for HSTS (in Section 2). Then, Section 3 defines conformance requirements. The HSTS mechanism itself is formally specified in Section 4 through Section 14.

2. Overview

This section discusses the use cases, summarizes the HTTP Strict Transport Security (HSTS) policy, and continues with a discussion of the threat model, non-addressed threats, and derived requirements.

2.1. Use Cases

The high-level use case is a combination of:

- o Web browser user wishes to interact with various web sites (some arbitrary, some known) in a secure fashion.

- o Web site deployer wishes to offer their site in an explicitly secure fashion for both their own, as well as their users', benefit.

2.2. Strict Transport Security Policy Effects

The characteristics of the HTTP Strict Transport Security policy, as applied by a UA in its interactions with a web site wielding HSTS Policy, known as a HSTS Host, is summarized as follows:

1. All insecure ("http") connections to any TCP ports on a HSTS Host are redirected by the HSTS Host to be secure connections ("https").

2. The UA terminates any secure transport connection attempts upon any and all secure transport errors or warnings, including those caused by a web application presenting self-signed certificates.

3. UAs transform insecure URI references to a HSTS Host into secure URI references before dereferencing them.

2.3. Threat Model

HSTS is concerned with three threat classes: passive network attackers, active network attackers, and imperfect web developers. However, it is explicitly not a remedy for two other classes of threats: phishing and malware. Addressed and not addressed threats are briefly discussed below. Readers may wish refer to [ForceHTTPS] for details as well as relevant citations.

2.3.1. Threats Addressed

2.3.1.1. Passive Network Attackers

When a user browses the web on a local wireless network (e.g. an 802.11-based wireless local area network) a nearby attacker can possibly eavesdrop on the user's unencrypted Internet Protocol-based connections, such as HTTP, regardless of whether or not the local wireless network itself is secured [BeckTews09]. Freely available wireless sniffing toolkits, e.g. [Aircrack-ng], enable such passive eavesdropping attacks, even if the local wireless network is operating in a secure fashion. A passive network attacker using such tools can steal session identifiers/cookies and hijack the user's web session(s), by obtaining cookies containing authentication credentials [ForceHTTPS]. For example, there exist widely-available tools, such as Firesheep (a Firefox extension) [Firesheep], which enable their wielder to obtain other local users' session cookies for various web applications.

To mitigate such threats, some Web sites support, but usually do not force, access using end-to-end secure transport -- e.g. signaled through URIs constructed with the "https" scheme [RFC2818]. This can lead users to believe that accessing such services using secure transport protects them from passive network attackers. Unfortunately, this is often not the case in real-world deployments as session identifiers are often stored in non-Secure cookies to permit interoperability with versions of the service offered over insecure transport ("Secure cookes" are those cookies containing the

"Secure" attribute [RFC2109]). For example, if the session identifier for a web site (an email service, say) is stored in a non-Secure cookie, it permits an attacker to hijack the user's session if the user's UA makes a single insecure HTTP request to the site.

2.3.1.2. Active Network Attackers

A determined attacker can mount an active attack, either by impersonating a user's DNS server or, in a wireless network, by spoofing network frames or offering a similarly-named evil twin access point. If the user is behind a wireless home router, an attacker can attempt to reconfigure the router using default passwords and other vulnerabilities. Some sites, such as banks, rely on end-to-end secure transport to protect themselves and their users from such active attackers. Unfortunately, browsers allow their users to easily opt-out of these protections in order to be usable for sites that incorrectly deploy secure transport, for example by generating and self-signing their own certificates (without also distributing their CA certificate to their users' browsers).

2.3.1.3. Web Site Development and Deployment Bugs

The security of an otherwise uniformly secure site (i.e. all of its content is materialized via "https" URIs), can be compromised completely by an active attacker exploiting a simple mistake, such as the loading of a cascading style sheet or a SWF movie over an insecure connection (both cascading style sheets and SWF movies can script the embedding page, to the surprise of many web developers -- most browsers do not issue mixed content warnings when insecure SWF files are embedded). Even if the site's developers carefully scrutinize their login page for mixed content, a single insecure embedding anywhere on the site compromises the security of their login page because an attacker can script (control) the login page by injecting script into the page with mixed content.

Note: "Mixed content" here refers to the same notion referred to as "mixed security context" later elsewhere in this specification.

2.3.2. Threats Not Addressed

2.3.2.1. Phishing

Phishing attacks occur when an attacker solicits authentication credentials from the user by hosting a fake site located on a different domain than the real site, perhaps driving traffic to the fake site by sending a link in an email message. Phishing attacks can be very effective because users find it difficult to distinguish

the real site from a fake site. HSTS is not a defense against phishing per se; rather, it complements many existing phishing defenses by instructing the browser to protect session integrity and long-lived authentication tokens [ForceHTTPS].

2.3.2.2. Malware and Browser Vulnerabilities

Because HSTS is implemented as a browser security mechanism, it relies on the trustworthiness of the user's system to protect the session. Malicious code executing on the user's system can compromise a browser session, regardless of whether HSTS is used.

2.4. Requirements

This section identifies and enumerates various requirements derived from the use cases and the threats discussed above, and lists the detailed core requirements HTTP Strict Transport Security addresses, as well as ancillary requirements that are not directly addressed.

2.4.1. Overall Requirement

- Minimize the risks to web browser users and web site deployers that are derived from passive and active network attackers, web site development and deployment bugs, as well as insecure user actions.

2.4.1.1. Detailed Core Requirements

These core requirements are derived from the overall requirement, and are addressed by this specification.

1. Web sites need to be able to declare to UAs that they should be interacted with using a strict security policy.

2. Web sites need to be able to instruct UAs that contact them insecurely to do so securely.

3. UAs need to note web sites that signal strict security policy enablement, for a web site declared time span.

4. UAs need to re-write all insecure UA "http" URI loads to use the "https" secure scheme for those web sites for which secure policy is enabled.

5. Web site administrators need to be able to signal strict security policy application to subdomains of higher-level domains for which strict security policy is enabled, and UAs need to enforce such policy.

6. For example, both example.com and foo.example.com could set policy for bar.foo.example.com.

7. UAs need to disallow security policy application to peer domains, and/or higher-level domains, by domains for which strict security policy is enabled.

8. For example, neither bar.foo.example.com nor foo.example.com can set policy for example.com, nor can bar.foo.example.com set policy for foo.example.com. Also, foo.example.com cannot set policy for sibling.example.com.

9. UAs need to prevent users from clicking-through security warnings. Halting connection attempts in the face of secure transport exceptions is acceptable.

Note: A means for uniformly securely meeting the first core requirement above is not specifically addressed by this specification (see Section 13.4 "Bootstrap MITM Vulnerability"). It may be addressed by a future revision of this specification or some other specification. Note also that there are means by which UA implementations may more fully meet the first core requirement, see Section 10 "UA Implementation Advice".

2.4.1.2. Detailed Ancillary Requirements

These ancillary requirements are also derived from the overall requirement. They are not normatively addressed in this specification, but could be met by UA implementations at their implementor's discretion, although meeting these requirements may be complex.

1. Disallow "mixed security context" (also known as "mixed-content") loads (see section 5.3 "Mixed Content" in [W3C.WD-wsc-ui-20100309]).

2. Facilitate user declaration of web sites for which strict security policy is enabled, regardless of whether the sites signal HSTS Policy.

3. Conformance Criteria

This specification is written for hosts and user agents (UAs).

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this

document are to be interpreted as described in [RFC2119].

A conformant host is one that implements all the requirements listed in this specification that are applicable to hosts.

A conformant user agent is one that implements all the requirements listed in this specification that are applicable to user agents.

3.1. Document Conventions

Note: ..is a note to the reader. These are points that should be expressly kept in mind and/or considered.

Warning: This is how a warning is shown. These are things that can have suboptimal downside risks if not heeded.

4. Terminology

Terminology is defined in this section.

ASCII case-insensitive comparison
: means comparing two strings exactly, codepoint for codepoint, except that the characters in the range U+0041 .. U+005A (i.e. LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER Z) and the corresponding characters in the range U+0061 .. U+007A (i.e. LATIN SMALL LETTER A to LATIN SMALL LETTER Z) are considered to also match. See [Unicode6] for details.

codepoint
: is a colloquial contraction of Code Point, which is any value in the Unicode codespace; that is, the range of integers from 0 to 10FFFF(hex) [Unicode6].

domain name
: domain names, also referred to as DNS Names, are defined in [RFC1035] to be represented outside of the DNS protocol itself (and implementations thereof) as a series of labels separated by dots, e.g. "example.com" or "yet.another.example.org". In the context of this specification, domain names appear in that portion of a URI satisfying the reg-name production in "Appendix A. Collected ABNF for URI" in [RFC3986], and the host component from the Host HTTP header field production in section 14.23 of [RFC2616].

Note: The domain names appearing in actual URI instances and matching the aforementioned production components may or may not be FQDNs.

domain name label is that portion of a domain name appearing "between the dots", i.e. consider "foo.example.com": "foo", "example", and "com" are all domain name labels.

Effective Request URI
is a URI, identifying the target resource, that can be inferred by an HTTP host for any given HTTP request it receives. Such inference is necessary because HTTP requests often do not contain a complete "absolute" URI identifying the target resource. See Section 12 "Constructing an Effective Request URI", below.

FQDN
is an acronym for Fully-qualified domain name. A FQDN is a domain name that includes all higher level domains relevant to the named entity (typically a HSTS Host in the context of this specification). If one thinks of the DNS as a tree-structure with each node having its own domain name label, a FQDN for a specific node would be its label followed by the labels of all the other nodes between it and the root of the tree. For example, for a host, a FQDN would include the label that identifies the particular host, plus all domains of which the host is a part, up to and including the top-level domain (the root domain is always null) [RFC1594].

HTTP Strict Transport Security
is the overall name for the combined UA- and server-side security policy defined by this specification.

HTTP Strict Transport Security Host
is a HTTP host implementing the server aspects of the HSTS policy.

HTTP Strict Transport Security Policy
is the name of the combined overall UA- and server-side facets of the behavior specified in this specification.

HSTS	See HTTP Strict Transport Security.
HSTS Host	See HTTP Strict Transport Security Host.
HSTS Policy	See HTTP Strict Transport Security Policy.
Known HSTS Host	is a HSTS Host for which the UA has a HSTS Policy in effect.
Local policy	is comprised of policy rules deployers specify and which are often manifested as "configuration settings".
MITM	is an acronym for man-in-the-middle. See "man-in-the-middle attack" in [RFC4949].
Request URI	is the URI used to cause a UA to issue an HTTP request message.
UA	is a an acronym for user agent. For the purposes of this specification, a UA is an HTTP client application typically actively manipulated by a user [RFC2616] .

5. Syntax

This section defines the syntax of the new header this specification introduces. It also provides a short description of the function the header.

The Section 6 "Server Processing Model" section details how hosts are to use this header. Likewise, the Section 7 "User Agent Processing Model" section details how user agents are to use this header.

5.1. Strict-Transport-Security HTTP Response Header Field

The Strict-Transport-Security HTTP response header field indicates to a UA that it MUST enforce the HSTS Policy in regards to the host emitting the response message containing this header field.

The ABNF syntax for the Strict-Transport-Security HTTP Response Header field is:

```
 Strict-Transport-Security = "Strict-Transport-Security" ":"
                                directive *( ";" [ directive ] )
```

STS directives:

```
 directive          = max-age | includeSubDomains | STS-d-ext

 max-age            = "max-age" "=" delta-seconds

 includeSubDomains = "includeSubDomains"
```

The max-age directive MUST appear once in the Strict-Transport-Security header field value. The includeSubDomains directive MAY appear once. The order of appearance of directives in the Strict-Transport-Security header field value is not significant.

Additional directives extending the the semantic functionality of the Strict-Transport-Security header field may be defined in other specifications, using the STS directive extension point (STS-d-ext) syntax:

```
 STS-d-ext     = token [ "=" ( token | quoted-string ) ]
```

Defined in [RFC2616]:

```
 delta-seconds = <1*DIGIT, defined in [RFC2616], Section 3.3.2>
 token         = <token, defined in [RFC2616], Section 2.2>
 quoted-string = <quoted-string, defined in [RFC2616], Section 2.2>
```

5.1.1. max-age

max-age specifies the number of seconds, after the recption of the Strict-Transport-Security HTTP Response Header, during which the UA regards the host the message was received from as a Known HSTS Host (see also Section 7.1.1 "Noting a HSTS Host", below). The delta-seconds production is specified in [RFC2616].

5.1.2. includeSubDomains

includeSubDomains is a flag which, if present, signals to the UA that the HSTS Policy applies to this HSTS Host as well as any subdomains of the host's FQDN.

5.2. Examples

The below HSTS header field stipulates that the HSTS policy is to remain in effect for one year (there are approximately 31 536 000 seconds in a year), and the policy applies only to the domain of the HSTS Host issuing it:

```
Strict-Transport-Security: max-age=31536000
```

The below HSTS header field stipulates that the HSTS policy is to remain in effect for approximately six months and the policy applies only to the domain of the issuing HSTS Host and all of its subdomains:

```
Strict-Transport-Security: max-age=15768000 ; includeSubDomains
```

6. Server Processing Model

This section describes the processing model that HSTS Hosts implement. The model is comprised of two facets: the first being the processing rules for HTTP request messages received over a secure transport (e.g. TLS [RFC4346], SSL [I-D.ietf-tls-ssl-version3], or perhaps others, the second being the processing rules for HTTP request messages received over non-secure transports, i.e. over TCP/IP [RFC0793].

6.1. HTTP-over-Secure-Transport Request Type

When replying to an HTTP request that was conveyed over a secure transport, a HSTS Host SHOULD include in its response message a Strict-Transport-Security HTTP Response Header that MUST satisfy the grammar specified above in Section 5.1 "Strict-Transport-Security HTTP Response Header Field". If a Strict-Transport-Security HTTP Response Header is included, the HSTS Host MUST include only one such header.

Note: Including the Strict-Transport-Security HTTP Response Header is stipulated as a "SHOULD" in order to accomodate various server- and network-side caches and load-balancing configurations where it may be difficult to uniformly emit Strict-Transport-Security HTTP Response Headers on behalf of a given HSTS Host.

Establishing a given host as a Known HSTS Host, in the context of a given UA, MAY be accomplished over the HTTP protocol by correctly returning, per this specification, at least one valid Strict-Transport-Security HTTP Response Header to the UA. Other mechanisms, such as a client-side pre-loaded Known HSTS Host list MAY also be used. E.g. see Section 10 "UA Implementation Advice".

6.2. HTTP Request Type

If a HSTS Host receives a HTTP request message over a non-secure transport, it SHOULD send a HTTP response message containing a Status-Code of 301 and a Location header field value containing either the HTTP request's original Effective Request URI (see Section 12 "Constructing an Effective Request URI", below) altered as necessary to have a URI scheme of "https", or a URI generated according to local policy (which SHOULD employ a URI scheme of "https").

Note: The above behavior is a "SHOULD" rather than a "MUST" because:

* There are risks in server-side non-secure-to-secure redirects [owaspTLSGuide].

* Site deployment characteristics -- e.g. a site that incorporates third-party components may not behave correctly when doing server-side non-secure-to-secure redirects in the case of being accessed over non-secure transport, but does behave correctly when accessed uniformly over secure transport. The latter is the case given a HSTS-capapble UA that has already noted the site as a Known HSTS Host (by whatever means, e.g. prior interaction or UA configuration).

A HSTS Host MUST NOT include the Strict-Transport-Security HTTP Response Header in HTTP responses conveyed over non-secure transport.

7. User Agent Processing Model

This section describes the HTTP Strict Transport Security processing model for UAs. There are several facets to the model, enumerated by the following subsections.

This processing model assumes that the UA implements IDNA2008 [RFC5890], or possibly IDNA2003 [RFC3490], as noted in Section 11 "Internationalized Domain Names for Applications (IDNA): Dependency and Migration". It also assumes that all domain names manipulated in this specification's context are already IDNA-canonicalized as

outlined in Section 8 "Domain Name IDNA-Canonicalization" prior to the processing specified in this section.

The above assumptions mean that this processing model also specifically assumes that appropriate IDNA and Unicode validations and character list testing have occured on the domain names, in conjunction with their IDNA-canonicalization, prior to the processing specified in this section. See the IDNA-specific security considerations in Section 13.2 "Internationalized Domain Names" for rationale and further details.

7.1. Strict-Transport-Security Response Header Field Processing

If an HTTP response, received over a secure transport, includes a Strict-Transport-Security HTTP Response Header field, conforming to the grammar specified in Section 5.1 "Strict-Transport-Security HTTP Response Header Field" (above), and there are no underlying secure transport errors or warnings (see Section 7.3, below), the UA MUST either:

- Note the host as a Known HSTS Host if it is not already so noted (see Section 7.1.1 "Noting a HSTS Host", below),

or,

- Update its cached information for the Known HSTS Host if the max-age and/or includeSubDomains header field value tokens are conveying information different than that already maintained by the UA.

Note: The max-age value is essentially a "time to live" value relative to the reception time of the Strict-Transport-Security HTTP Response Header.

If a UA receives more than one Strict-Transport-Security header field in a HTTP response message over secure transport, then the UA MUST process only the first such header field.

Otherwise:

- If an HTTP response is received over insecure transport, the UA MUST ignore any present Strict-Transport-Security HTTP Response Header(s).

- The UA MUST ignore any Strict-Transport-Security HTTP Response Headers not conforming to the grammar specified in Section 5.1 "Strict-Transport-Security HTTP Response Header Field" (above).

7.1.1. Noting a HSTS Host

If the substring matching the host production from the Request-URI, that the host responded to, syntactically matches the IP-literal or IPv4address productions from section 3.2.2 of [RFC3986], then the UA MUST NOT note this host as a Known HSTS Host.

Otherwise, if the substring does not congruently match a presently known HSTS Host, per the matching procedure specified in Section 7.1.2 "Known HSTS Host Domain Name Matching" below, then the UA MUST note this host as a Known HSTS Host, caching the HSTS Host's domain name and noting along with it the expiry time of this information, as effectively stipulated per the given max-age value, as well as whether the includeSubDomains flag is asserted or not.

7.1.2. Known HSTS Host Domain Name Matching

A UA determines whether a domain name represents a Known HSTS Host by looking for a match between the query Domain Name and the UA's set of Known HSTS Hosts.

1. Compare the query domain name string with the Domain Names of the UA's set of Known HSTS Hosts. For each Known HSTS Host's domain name, the comparison is done with the query domain name label-by-label using an ASCII case-insensitive comparison beginning with the rightmost label, and continuing right-to-left, and ignoring separator characters (see clause 3.1(4) of [RFC3986].

   * If a label-for-label match between an entire Known HSTS Host's domain name and a right-hand portion of the query domain name is found, then the Known HSTS Host's domain name is a superdomain match for the query domain name.

     For example:

     ```
     Query Domain Name:          bar.foo.example.com

     Superdomain matched
     Known HSTS Host DN:             foo.example.com
     ```

     At this point, the query domain name is ascertained to effectively represent a Known HSTS Host. There may also be additional matches further down the domain name label tree, up to and including a congruent match.

   * If a label-for-label match between a Known HSTS Host's domain name and the query domain name is found, i.e. there are no

further labels to compare, then the query domain name congruently matches this Known HSTS Host.

For example:

Query Domain Name: foo.example.com

Congruently matched
Known HSTS Host DN: foo.example.com

The query domain name is ascertained to represent a Known HSTS Host. However, if there are also superdomain matches, the one highest in the tree asserts the HSTS Policy for this Known HSTS Host.

* Otherwise, if no matches are found, the query domain name does not represent a Known HSTS Host.

7.2. URI Loading and Port Mapping

Whenever the UA prepares to "load", also known as "dereference", any URI where the host component of the authority component of the URI [RFC3986] matches that of a Known HSTS Host (either as a congruent match or as a superdomain match where the superdomain Known HSTS Host has includeSubDomains asserted), then before proceeding with the load:

If the URI's scheme is "http", then the UA MUST replace the URI scheme with "https", and,

if the URI contains an explicit port component [RFC3986] of "80", then the UA MUST convert the port component to be "443", or,

if the URI contains an explicit port component that is not equal to "80", the port component value MUST be preserved, otherwise,

if the URI does not contain an explicit port component, the UA MUST NOT add one.

Otherwise, if the URI's scheme is "https", then the UA MUST NOT modify the URI before dereferencing it.

Note that the implication of the above steps is that the HSTS policy applies to all TCP ports on a host advertising the HSTS policy.

7.3. Errors in Secure Transport Establishment

When connecting to a Known HSTS Host, the UA MUST terminate the connection (see also Section 10 "UA Implementation Advice", below) if there are any errors (e.g. certificate errors), whether "warning" or "fatal" or any other error level, with the underlying secure transport. This includes any issues with certificate revocation checking whether via the Certificate Revocation List (CRL) [RFC5280], or via the Online Certificate Status Protocol (OCSP) [RFC2560].

7.4. HTTP-Equiv <Meta> Element Attribute

UAs MUST NOT heed http-equiv="Strict-Transport-Security" attribute settings on <meta> elements in received content.

7.5. **Interstitially Missing** Strict-Transport-Security Response Header Field

If a UA receives HTTP responses from a Known HSTS Host over a secure channel, but they are missing the Strict-Transport-Security Response Header Field, the UA MUST continue to treat the host as a Known HSTS Host until the max-age value for the knowledge that Known HSTS Host is reached. Note that the max age could be infinite for a given Known HSTS Host. For example, if the Known HSTS Host is part of a pre-configured list that is implemented such that the list entries never "age out".

8. Domain Name IDNA-Canonicalization

An IDNA-canonicalized domain name is the string generated by the following algorithm, whose input must be a valid Unicode-encoded (in NFC form [Unicode6]) string-serialized domain name:

1. Convert the domain name to a sequence of individual domain name label strings.

2. When implementing IDNA2008, convert each label that is not a Non-Reserved LDH (NR-LDH) label, to an A-label. See Section 2.3.2 of [RFC5890] for definitions of the former and latter, refer to Sections 5.3 through 5.5 of [RFC5891] for the conversion algorithm and requisite input validation and character list testing procedures.

   Otherwise, when implementing IDNA2003, convert each label using the "ToASCII" conversion in Section 4 of [RFC3490] (see also the definition of "equivalence of labels" in Section 2 of the latter specification).

3. Concatenate the resulting labels, separating each label from the next with (".") a %x2E character.

See also Section 11 "Internationalized Domain Names for Applications (IDNA): Dependency and Migration" and Section 13.2 "Internationalized Domain Names" of this specification for further details and considerations.

9. Server Implementation Advice

This section is non-normative.

HSTS Policy expiration time considerations:

- Server implementations and deploying web sites need to consider whether they are setting an expiry time that is a constant value into the future, e.g. by constantly sending the same max-age value to UAs. For exmple:

  ```
  Strict-Transport-Security: max-age=778000
  ```

  A max-age value of 778000 is 90 days. Note that each receipt of this header by a UA will require the UA to update its notion of when it must delete its knowledge of this Known HSTS Host. The specifics of how this is accomplished is out of the scope of this specification.

- Or, whether they are setting an expiry time that is a fixed point in time, e.g. by sending max-age values that represent the remaining time until the expiry time.

- A consideration here is whether a deployer wishes to have signaled HSTS Policy expiry time match that for the web site's domain certificate.

Considerations for using HTTP Strict Transport Security in conjunction with self-signed public-key certificates:

- If a web site/organization/enterprise is generating their own secure transport public-key certificates for web sites, and that organization's root certificate authority (CA) certificate is not typically embedded by default in browser CA certificate stores, and if HSTS Policy is enabled on a site identifying itself using a self-signed certificate, then secure connections to that site will fail, per the HSTS design. This is to protect against various active attacks, as discussed above.

- However, if said organization strongly wishes to employ self-signed certificates, and their own CA in concert with HSTS, they can do so by deploying their root CA certificate to their users' browsers. They can also, in addition or instead, distribute to their users' browsers the end-entity certificate(s) for specific hosts. There are various ways in which this can be accomplished (details are out of scope for this specification). Once their root CA cert is installed in the browsers, they may employ HSTS Policy on their site(s).

  Note: Interactively distributing root CA certs to users, e.g. via email, and having the users install them, is arguably training the users to be susceptible to a possible form of phishing attack, see Section 13.6 "Bogus Root CA Certificate Phish plus DNS Cache Poisoning Attack".

10. UA Implementation Advice

This section is non-normative.

In order to provide users and web sites more effective protection, UA implementors should consider including features such as:

- Failing secure connection establishment on any warnings or errors, as noted in Section 7.3 "Errors in Secure Transport Establishment", should be done with no user recourse. This means that the user should not be presented with an explanatory dialog giving her the option to proceed. Rather, it should be treated similarly to a server error where there is nothing further the user can do with respect to interacting with the target web application, other than wait and re-try.

  Essentially, "any warnings or errors" means anything that would cause the UA implementation to annunciate to the user that something is not entirely correct with the connection establishment.

  Not doing this, i.e., allowing user recourse such as "clicking-through warning/error dialogs", is a recipe for a Man-in-the-Middle attack. If a web application advertises HSTS, then it is opting into this scheme, whereby all certificate errors or warnings cause a connection termination, with no chance to "fool" the user into making the wrong decision and compromising themselves.

- Disallowing "mixed security context" (also known as "mixed-content") loads (see section 5.3 "Mixed Content" in

[W3C.WD-wsc-ui-20100309]).

Note: In order to provide behavioral uniformity across UA implementations, the notion of mixed security context aka mixed-content will require (further) standardization work, e.g. to more clearly define the term(s) and to define specific behaviors with respect to it.

In order to provide users effective controls for managing their UA's caching of HSTS Policy, UA implementors should consider including features such as:

- o Ability to delete UA's cached HSTS Policy on a per HSTS Host basis.

  Note: Adding such a feature should be done very carefully in both the user interface and security senses. Deleting a cache entry for a Known HSTS Host should be a very deliberate and well-considered act -- it shouldn't be something users get used to just "clicking through" in order to get work done. Also, it shouldn't be possible for an attacker to inject script into the UA that silently and programmatically removes entries from the UA's cache of Known HSTS Hosts.

In order to provide users and web sites more complete protection, UAs could offer advanced features such as these:

- o Ability for users to explicitly declare a given Domain Name as representing a HSTS Host, thus seeding it as a Known HSTS Host before any actual interaction with it. This would help protect against the Section 13.4 "Bootstrap MITM Vulnerability".

  Note: Such a feature is difficult to get right on a per-site basis -- see the discussion of "rewrite rules" in section 5.5 of [ForceHTTPS]. For example, arbitrary web sites may not materialize all their URIs using the "https" scheme, and thus could "break" if a UA were to attempt to access the site exclusively using such URIs. Also note that this feature would complement, but is independent of the following described facility.

- o Facility whereby web site administrators can have UAs pre-configured with HSTS Policy for their site(s) by the UA vendor(s) -- in a manner similar to how root CA certificates are embedded in browsers "at the factory". This would help protect against the Section 13.4 "Bootstrap MITM Vulnerability".

Note: Such a facility complements the preceding described feature.

11. Internationalized Domain Names for Applications (IDNA): Dependency and Migration

Textual domain names on the modern Internet may contain one or more "internationalized" domain name labels. Such domain names are referred to as "internationalized domain names" (IDNs). The specification suites defining IDNs and the protocols for their use are named "Internationalized Domain Names for Applications (IDNA)". At this time, there are two such specification suites: IDNA2008 [RFC5890] and its predecessor IDNA2003 [RFC3490].

IDNA2008 obsoletes IDNA2003, but there are differences between the two specifications, and thus there can be differences in processing (e.g. converting) domain name labels that have been registered under one from those registered under the other. There will be a transition period of some time during which IDNA2003-based domain name labels will exist in the wild. User agents SHOULD implement IDNA2008 [RFC5890] and MAY implement [RFC5895] (see also Section 7 of [RFC5894]) or [UTS46] in order to facilitate their IDNA transition. If a user agent does not implement IDNA2008, the user agent MUST implement IDNA2003.

12. Constructing an Effective Request URI

This section specifies how an HSTS Host must construct the Effective Request URI for a received HTTP request.

HTTP requests often do not carry an absolute-URI ([RFC3986], Section 4.3) for the target resource; instead, the URI needs to be inferred from the Request-URI, Host header field, and connection context. The result of this process is called the "effective request URI (ERU)". The "target resource" is the resource identified by the effective request URI.

12.1. ERU Fundamental Definitions

The first line of an HTTP request message, Request-Line, is specified by the following ABNF from [RFC2616], section 5.1:

```
Request-Line   = Method SP Request-URI SP HTTP-Version CRLF
```

The Request-URI, within the Request-Line, is specified by the following ABNF from [RFC2616], section 5.1.2:

```
Request-URI    = "*" | absoluteURI | abs_path | authority
```

The Host request header field is specified by the following ABNF from [RFC2616], section 14.23:

```
Host = "Host" ":" host [ ":" port ]
```

12.2. Determining the Effective Requrest URI

If the Request-URI is an absoluteURI, then the effective request URI is the Request-URI.

If the Request-URI uses the abs_path form or the asterisk form, and the Host header field is present, then the effective request URI is constructed by concatenating:

- o the scheme name: "http" if the request was received over an insecure TCP connection, or "https" when received over a TLS/ SSL-secured TCP connection, and,
- o the octet sequence "://", and,
- o the host, and the port (if present), from the Host header field, and
- o the Request-URI obtained from the Request-Line, unless the Request-URI is just the asterisk "*".

If the Request-URI uses the abs_path form or the asterisk form, and the Host header field is not present, then the effective request URI is undefined.

Otherwise, when Request-URI uses the authority form, the effective request URI is undefined.

Effective request URIs are compared using the rules described in [RFC2616] Section 3.2.3, except that empty path components MUST NOT be treated as equivalent to an absolute path of "/".

12.2.1. Effective Requrest URI Examples

Example 1: the effective request URI for the message

```
GET /pub/WWW/TheProject.html HTTP/1.1
Host: www.example.org:8080
```

(received over an insecure TCP connection) is "http", plus "://", plus the authority component "www.example.org:8080", plus the

request-target "/pub/WWW/TheProject.html". Thus it is: "http://www.example.org:8080/pub/WWW/TheProject.html".

Example 2: the effective request URI for the message

```
OPTIONS * HTTP/1.1
Host: www.example.org
```

(received over an SSL/TLS secured TCP connection) is "https", plus "://", plus the authority component "www.example.org". Thus it is: "https://www.example.org".

13. Security Considerations

13.1. The Need for includeSubDomains

Without the includeSubDomains directive, a web application would not be able to adequately protect so-called "domain cookies" (even if these cookies have their "Secure" flag set and thus are conveyed only on secure channels). These are cookies the web application expects UAs to return to any and all subdomains of the web application.

For example, suppose example.com represents the top-level DNS name for a web application. Further suppose that this cookie is set for the entire example.com domain, i.e. it is a "domain cookie", and it has its Secure flag set. Suppose example.com is a Known HSTS Host for this UA, but the includeSubDomains flag is not set.

Now, if an attacker causes the UA to request a subdomain name that is unlikely to already exist in the web application, such as "https://uxdhbpahpdsf.example.com/", but the attacker has established somewhere and registered in the DNS, then:

1. The UA is unlikely to already have an HSTS policy established for "uxdhbpahpdsf.example.com", and,

2. The HTTP request sent to uxdhbpahpdsf.example.com will include the Secure-flagged domain cookie.

3. If "uxdhbpahpdsf.example.com" returns a certificate during TLS establishment, and the user clicks through any warning that might be annunciated (it is possible, but not certain, that one may obtain a requisite certificate for such a domain name such that a warning may or may not appear), then the attacker can obtain the Secure-flagged domain cookie that's ostensibly being protected.

Without the "includeSubDomains" directive, HSTS is unable to protect

such Secure-flagged domain cookies.

13.2. Internationalized Domain Names

Internet security relies in part on the DNS and the domain names it hosts. Domain names are used by users to identify and connect to Internet hosts and other network resources. For example, Internet security is compromised if a user entering an internationalized domain name (IDN) is connected to different hosts based on different interpretations of the IDN.

The processing models specified in this specification assume that the domain names they manipulate are IDNA-canonicalized, and that the canonicalization process correctly performed all appropriate IDNA and Unicode validations and character list testing per the requisite specifications (e.g., as noted in Section 8 "Domain Name IDNA-Canonicalization"). These steps are necessary in order to avoid various potentially compromising situations.

In brief, some examples of issues that could stem from lack of careful and consistent Unicode and IDNA validations are things such as unexpected processing exceptions, truncation errors, and buffer overflows, as well as false-positive and/or false-negative domain name matching results. Any of the foregoing issues could possibly be leveraged by attackers in various ways.

Additionally, IDNA2008 [RFC5890] differs from IDNA2003 [RFC3490] in terms of disallowed characters and character mapping conventions. This situation can also lead to false-positive and/or false-negative domain name matching results, resulting in, for example, users possibly communicating with unintended hosts, or not being able to reach intended hosts.

For details, refer to the Security Considerations sections of [RFC5890], [RFC5891], and [RFC3490], as well as the specifications they normatively reference. Additionally, [RFC5894] provides detailed background and rationale for IDNA2008 in particular, as well as IDNA and its issues in general, and should be consulted in conjunction with the former specifications.

13.3. Denial of Service (DoS)

HSTS could be used to mount certain forms of DoS attacks, where attackers cause UAs to set fake HSTS headers for legitimate sites available only insecurely (e.g. social network service sites, wikis, etc.).

13.4. Bootstrap MITM Vulnerability

The bootstrap MITM (Man-In-The-Middle) vulnerability is a vulnerability users and HSTS Hosts encounter in the situation where the user manually enters, or follows a link, to a HSTS Host using a "http" URI rather than a "https" URI. Because the UA uses an insecure channel in the initial attempt to interact with the specified serve, such an initial interaction is vulnerable to various attacks [ForceHTTPS] .

Note: There are various features/facilities that UA implementations may employ in order to mitigate this vulnerability. Please see Section 10 UA Implementation Advice.

13.5. Network Time Attacks

Active network attacks can subvert network time protocols (like NTP) - making this header less effective against clients that trust NTP and/or lack a real time clock. Network time attacks are therefore beyond the scope of the defense. Note that modern operating systems use NTP by default.

13.6. Bogus Root CA Certificate Phish plus DNS Cache Poisoning Attack

If an attacker can convince users of, say, https://bank.example.com (which is protected by HSTS Policy), to install their own version of a root CA certificate purporting to be bank.example.com's CA, e.g. via a phishing email message with a link to such a certificate -- then, if they can perform an attack on the users' DNS, e.g. via cache poisoning, and turn on HSTS Policy for their fake bank.example.com site, then they have themselves some new users.

14. IANA Considerations

Below is the Internet Assigned Numbers Authority (IANA) Provisional Message Header Field registration information per [RFC3864].

```
Header field name:           Strict-Transport-Security
Applicable protocol:         HTTP
Status:                      provisional
Author/Change controller:    TBD
Specification document(s):   this one
```

15. References

15.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC1594] Marine, A., Reynolds, J., and G. Malkin, "FYI on Questions and Answers - Answers to Commonly asked "New Internet User" Questions", RFC 1594, March 1994.

[RFC1983] Malkin, G., "Internet Users' Glossary", RFC 1983, August 1996.

[RFC2109] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", RFC 2109, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC2965] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", RFC 2965, October 2000.

[RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform

Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[RFC5894] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, August 2010.

[RFC5895] Resnick, P. and P. Hoffman, "Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008", RFC 5895, September 2010.

[Unicode6] The Unicode Consortium, "The Unicode Standard, Version 6.0 - Core Specification", Unicode 6.0.0, Mountain View, CA, The Unicode Consortium ISBN 978-1-936213-01-6, 2011, <http://www.unicode.org/versions/Unicode6.0.0/>.

[W3C.WD-html5-20100304] Hyatt, D. and I. Hickson, "HTML5", World Wide Web Consortium WD WD-html5-20100304, March 2010, <http://www.w3.org/TR/2010/WD-html5-20100304>.

15.2. Informative References

[Aircrack-ng] d'Otreppe, T., "Aircrack-ng", Accessed: 11-Jul-2010, <http://www.aircrack-ng.org/>.

[BeckTews09] Beck, M. and E. Tews, "Practical Attacks Against WEP and

WPA", Second ACM Conference on Wireless Network Security Zurich, Switzerland, 2009, <http://wirelesscenter.dk/Crypt/wifi-security-attacks/Practical%20Attacks%20Against%20WEP%20and%20WPA.pdf>.

[Firesheep]
Various, "Firesheep", Wikipedia Online, on-going, <https://secure.wikimedia.org/wikipedia/en/wiki/Firesheep>.

[ForceHTTPS]
Jackson, C. and A. Barth, "ForceHTTPS: Protecting High-Security Web Sites from Network Attacks", In Proceedings of the 17th International World Wide Web Conference (WWW2008) , 2008, <https://crypto.stanford.edu/forcehttps/>.

[GoodDhamijaEtAl05]
Good, N., Dhamija, R., Grossklags, J., Thaw, D., Aronowitz, S., Mulligan, D., and J. Konstan, "Stopping Spyware at the Gate: A User Study of Privacy, Notice and Spyware", In Proceedings of Symposium On Usable Privacy and Security (SOUPS) Pittsburgh, PA, USA, July 2005, <http://people.ischool.berkeley.edu/~rachna/papers/spyware_study.pdf>.

[I-D.ietf-tls-ssl-version3]
Freier, A., Karlton, P., and P. Kocher, "The SSL Protocol Version 3.0", draft-ietf-tls-ssl-version3 (work in progress), November 1996, <http://tools.ietf.org/html/draft-ietf-tls-ssl-version3-00>.

[JacksonBarth2008]
Jackson, C. and A. Barth, "Beware of Finer-Grained Origins", Web 2.0 Security and Privacy Oakland, CA, USA, 2008, <http://www.adambarth.com/papers/2008/jackson-barth-b.pdf>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC2396] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[SunshineEgelmanEtAl09]
Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., and

L. Cranor, "Crying Wolf: An Empirical Study of SSL Warning Effectiveness", In Proceedings of 18th USENIX Security Symposium Montreal, Canada, Augus 2009, <http://www.usenix.org/events/sec09/tech/full_papers/sunshine.pdf>.

[UTS46] Davis, M. and M. Suignard, "Unicode IDNA Compatibility Processing", Unicode Technical Standards # 46, 2010, <http://unicode.org/reports/tr46/>.

[W3C.WD-wsc-ui-20100309] Saldhana, A. and T. Roessler, "Web Security Context: User Interface Guidelines", World Wide Web Consortium LastCall WD-wsc-ui-20100309, March 2010, <http://www.w3.org/TR/2010/WD-wsc-ui-20100309>.

[WEBSEC] "WebSec -- HTTP Application Security Minus Authentication and Transport", <https://www.ietf.org/mailman/listinfo/websec>.

[owaspTLSGuide] Coates, M., Wichers, d., Boberski, M., and T. Reguly, "Transport Layer Protection Cheat Sheet", Accessed: 11-Jul-2010, <http://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet>.

Appendix A. Design Decision Notes

This appendix documents various design decisions.

1. Cookies aren't appropriate for HSTS Policy expression as they are potentially mutable (while stored in the UA), therefore an HTTP header field is employed.

2. We chose to not attempt to specify how "mixed security context loads" (aka "mixed-content loads") are handled due to UA implementation considerations as well as classification difficulties.

3. A HSTS Host may update UA notions of HSTS Policy via new HSTS header field values. We chose to have UAs honor the "freshest" information received from a server because there is the chance of a web site sending out an errornous HSTS Policy, such as a multi-year max-age value, and/or an incorrect includeSubDomains flag. If the HSTS Host couldn't correct such errors over protocol, it would require some form of annunciation to users and manual intervention on their part, which could be a non-trivial problem.

4. HSTS Hosts are identified only via domain names -- explicit IP address identification of all forms is excluded. This is for simplification and also is in recognition of various issues with using direct IP address identification in concert with PKI-based security.

Appendix B. Acknowledgments

The authors thank Devdatta Akhawe, Michael Barrett, Paul Hoffman, Yoav Nir, Julian Reschke, Tom Ritter, Peter Saint-Andre, Sid Stamm, Maciej Stachowiak, Andy Steingrubl, Brandon Sterne, Martin Thomson, Daniel Veditz, and all the other websec working group participants for their review and contributions.

Thanks to Julian Reschke for his elegant re-writing of the effective request URI text, which he did when incorporating the ERU notion into the HTTPbis work. Subsequently, the ERU text in this spec was lifted from Julian's work in [I-D.draft-ietf-httpbis-p1-messaging-16] and adapted to the [RFC2616] ABNF.

Appendix C. Change Log

[RFCEditor: please remove this section upon publication as an RFC.]

Changes are grouped by spec revision listed in reverse issuance order.

C.1. For draft-ietf-websec-strict-transport-sec

Changes from -02 to -03:

1. Completely re-wrote the STS header ABNF to be fully based on RFC2616, rather than a hybrid of RFC2616 and httpbis. [no submitted issue ticket as yet]

2. Updated section on "Constructing an Effective Request URI" to remove references to RFC3986. Addresses issue ticket #14. <http://trac.tools.ietf.org/wg/websec/trac/ticket/14>

3. Reference RFC5890 rather than RFC3490 for IDNA. Updated IDNA-specific language, e.g. domain name canonicalization and IDNA dependencies. [no submitted issue ticket as yet]

Changes from -01 to -02:

1. Updated Section 7.2 "URI Loading and Port Mapping" fairly thoroughly in terms of refining the presentation of the steps, and to ensure the various aspects of port mapping are clear. Nominally fixes issue ticket #1 <http://trac.tools.ietf.org/wg/websec/trac/ticket/1>

2. Removed dependencies on [I-D.draft-ietf-httpbis-p1-messaging-15]. Thus updated STS ABNF in Section 5.1 "Strict-Transport-Security HTTP Response Header Field" by lifting some productions entirely from [I-D.draft-ietf-httpbis-p1-messaging-15] and leveraging [RFC2616]. Addresses issue ticket #2 <http://trac.tools.ietf.org/wg/websec/trac/ticket/2>.

3. Updated Effective Request URI section and definition to use language from [I-D.draft-ietf-httpbis-p1-messaging-15] and ABNF from [RFC2616]. Fixes issue ticket #3 <http://trac.tools.ietf.org/wg/websec/trac/ticket/3>.

4. Added explicit mention that the HSTS policy applies to all TCP ports of a host advertising the HSTS policy. Nominally fixes issue ticket #4 <http://trac.tools.ietf.org/wg/websec/trac/ticket/4>

5. Clarified the need for the "includeSubDomains" directive, e.g. to protect Secure-flagged domain cookies. In Section 13.1 "The Need for includeSubDomains". Nominally fixes issue ticket #5 <http://trac.tools.ietf.org/wg/websec/trac/ticket/5>

6. Cited Firesheep as real-live threat in Section 2.3.1.1 "Passive Network Attackers". Nominally fixes issue ticket #6 <http://trac.tools.ietf.org/wg/websec/trac/ticket/6>.

7. Added text to Section 10 "UA Implementation Advice" justifying connection termination due to tls warnings/errors. Nominally fixes issue ticket #7 <http://trac.tools.ietf.org/wg/websec/trac/ticket/7>.

8. Added new subsection Section 7.5 "Interstitially Missing Strict-Transport-Security Response Header Field". Nominally fixes issue ticket #8 <http://trac.tools.ietf.org/wg/websec/trac/ticket/8>.

9. Added text to Section 7.3 "Errors in Secure Transport Establishment" explicitly note revocation check failures as errors causing connection termination. Added references to [RFC5280] and [RFC2560]. Nominally fixes issue ticket #9

<http://trac.tools.ietf.org/wg/websec/trac/ticket/9>.

10. Added a sentence, noting that distributing specific end-entity certs to browsers will also work for self-signed/private-CA cases, to Section 9 "Server Implementation Advice" Nominally fixes issue ticket #10 <http://trac.tools.ietf.org/wg/websec/trac/ticket/10>.

11. Moved "with no user recourse" language from Section 7.3 "Errors in Secure Transport Establishment" to Section 10 "UA Implementation Advice". This nominally fixes issue ticket #11 <http://trac.tools.ietf.org/wg/websec/trac/ticket/11>.

12. Removed any and all dependencies on [I-D.draft-ietf-httpbis-p1-messaging-15], instead depending on [RFC2616] only. Fixes issue ticket #12 <http://trac.tools.ietf.org/wg/websec/trac/ticket/12>.

13. Removed the inline "XXX1" issue because no one had commented on it and it seems reasonable to suggest as a SHOULD that web apps should redirect incoming insecure connections to secure connections.

14. Removed the inline "XXX2" issue because it was simply for raising consciousness about having some means for distributing secure web application metadata.

15. Removed "TODO1" because description prose for "max-age" in the Note following the ABNF in Section 5 seems to be fine.

16. Decided for "TODO2" that "the first STS header field wins". TODO2 had read: "Decide UA behavior in face of encountering multiple HSTS headers in a message. Use first header? Last?". Removed TODO2.

17. Added Section 1.1 "Organization of this specification" for readers' convenience.

18. Moved design decision notes to be a proper appendix Appendix A.

Changes from -00 to -01:

1. Changed the "URI Loading" section to be "URI Loading and Port Mapping".

2. [HASMAT] reference changed to [WEBSEC].

3. Changed "server" -> "host" where applicable, notably when discussing "HSTS Hosts". Left as "server" when discussing e.g. "http server"s.

4. Fixed minor editorial nits.

Changes from draft-hodges-strict-transport-sec-02 to draft-ietf-websec-strict-transport-sec-00:

1. Altered spec metadata (e.g. filename, date) in order to submit as a WebSec working group Internet-Draft.

C.2. For draft-hodges-strict-transport-sec

Changes from -01 to -02:

1. updated abstract such that means for expressing HSTS Policy other than via HSTS header field is noted.

2. Changed spec title to "HTTP Strict Transport Security (HSTS)" from "Strict Transport Security". Updated use of "STS" acronym throughout spec to HSTS (except for when specifically discussing syntax of Strict-Transport-Security HTTP Response Header field), updated "Terminology" appropriately.

3. Updated the discussion of "Passive Network Attackers" to be more precise and offered references.

4. Removed para on nomative/non-normative from "Conformance Criteria" pending polishing said section to IETF RFC norms.

5. Added examples subsection to "Syntax" section.

6. Added OWS to maxAge production in Strict-Transport-Security ABNF.

7. Cleaned up explanation in the "Note:" in the "HTTP-over-Secure-Transport Request Type" section, folded 3d para into "Note:", added conformance clauses to the latter.

8. Added exaplanatory "Note:" and reference to "HTTP Request Type" section. Added "XXX1" issue.

9. Added conformance clause to "URI Loading".

10. Moved "Notes for STS Server implementors:" from "UA Implementation dvice " to "HSTS Policy expiration time considerations:" in "Server Implementation Advice", and also

noted another option.

11. Added cautionary "Note:" to "Ability to delete UA's cached HSTS Policy on a per HSTS Server basis".

12. Added some informative references.

13. Various minor editorial fixes.

Changes from -00 to -01:

1. Added reference to HASMAT mailing list and request that this spec be discussed there.

Authors' Addresses

Jeff Hodges
PayPal
2211 North First Street
San Jose, California 95131
US

Email: Jeff.Hodges@PayPal.com

Collin Jackson
Carnegie Mellon University

Email: collin.jackson@sv.cmu.edu

Adam Barth
Google, Inc.

Email: ietf@adambarth.com
URI: http://www.adambarth.com/