Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 12, 2011

J. Hodges
PayPal
C. Jackson
Carnegie Mellon University
A. Barth
University of California
Berkeley
July 11, 2010

# HTTP Strict Transport Security (HSTS)
draft-hodges-strict-transport-sec-02

## 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.

Table of Contents

## 1. Introduction

[ Please disscuss this draft on the hasmat@ietf.org mailing list [HASMAT]. ]

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 server 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 Universal 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 server only over a secure transport. This is in contrast to non-Secure cookies, which are returned to the server 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 server certificate trust chain, or if a server certificate is expired, or if a server's domain name appears incorrectly in the 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" .

Jackson and Barth proposed an approach, in [ForceHTTPS], to enable web sites and/or users to declare that such issues 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], e.g. a HTTP response header field is used to convey site policy to the UA rather than a cookie.

## 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:

- Web browser user wishes to discover, or be introduced to, and/or utilize various web sites (some arbitrary, some known) in a secure fashion.

- 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 Server, is summarized as follows:

1. All insecure ("http") connections to a HSTS Server are redirected by the HSTS Server to be secure connections ("https").

2. The UA terminates, without user recourse, any secure transport connection attempts upon any and all secure transport errors or warnings, including those caused by a site presenting self-signed certificates.

3. UAs transform insecure URI references to a HSTS Server 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. A passive network attacker using such tools can steal session identifiers and hijack the user's web session(s), by obtaining cookies containing authentication credentials for example [ForceHTTPS].

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 12.2 "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 servers and user agents (UAs).

In this specification, the words MUST, MUST NOT, MAY, and SHOULD are to be interpreted as described in [RFC2119].

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

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.

[[XXXn: Some of the more major known issues are marked like this (where "n" in "XXXn" is a number). --JeffH]]

[[TODOn: Things to fix (where "n" in "TODOn" is a number). --JeffH]]

## 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 [Unicode5] 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) [Unicode5].

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 that can be constructed by an HTTP server for any given HTTP request sent to it. Some HTTP requests do not contain a contiguous representation of the URI identifying the resource being addressed by the HTTP request. Rather, different portions of a resource's URI may be mapped to both the Request-Line header field and the Host header field in an HTTP request message [I-D.ietf-httpbis-p1-messaging]. The HTTP server coalesces these URI fragments and constructs an equivalent of the Request-URI that was used by the UA to generate the received HTTP request message.

See Section 11 "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 Server 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 Server
is a HTTP server 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 Policy See HTTP Strict Transport Security Policy.

HSTS Server See HTTP Strict Transport Security Server.

Known HSTS Server is a HSTS Server for which the UA has an 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 servers 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 server 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" ":"  OWS  STS-v  OWS

; value
STS-v      = STS-d
           / STS-d  *( OWS ";" OWS  STS-d  OWS )

; STS directive
STS-d      = STS-d-cur / STS-d-ext

; defined STS directives
STS-d-cur  = maxAge / includeSubDomains

maxAge     = "max-age"  OWS  "="  OWS  delta-seconds  v-ext

includeSubDomains =  [ "includeSubDomains" ]  v-ext


; extension points
STS-d-ext  = name      ; STS extension directive

v-ext      = value     ; STS extension value

name       = token

value      = OWS / %x21-3A  /  %x3C-7E   ; i.e. optional white space, or
           ;     [ ! .. : ]  [ < .. ~ ] any visible chars other than ";"

;  productions imported from [ID.ietf-httpbis-p1-messaging]:

token

OWS        ; Optional White Space
```

Note: [I-D.ietf-httpbis-p1-messaging] is used as the ABNF basis in order to ensure that the new header has equivalent parsing rules to the header fields defined in that same specification. Also:

1. Quoted-string literals in the above ABNF stanza are case-insensitive.

2. In order to correctly match the grammar above, the Strict-Transport-Security HTTP Response Header MUST include at least a max-age directive with at least a single-digit value for delta-seconds.

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 Server (see also Section 7.1.1 "Noting a HSTS Server", below). The delta-seconds production is specified in [RFC2616].

[[TODO1: The above para wrt max-age may need further refinement. --JeffH]]

includeSubDomains is a flag which, if present, signals to the UA that the HSTS Policy applies to this HSTS Server as well as any subdomains of the server'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 Server issuing it:

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

The below HSTS header field is 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 Server and all of its subdomains:

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

# 6. Server Processing Model

This section describes the processing model that HSTS Servers 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 Server 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 Server 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 Server.

Establishing a given host as a Known HSTS Server, 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 Server list MAY also be used. E.g. see Section 10 "UA Implementation Advice".

### 6.2. HTTP Request Type

If a HSTS Server 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 11 "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 Server (by whatever means, e.g. prior interaction or UA configuration).

[[XXX1: perhaps the "SHOULD" in the above behavior should be a "MAY" given the reasons it's presently not a "MUST". --JeffH]]

A HSTS Server MUST NOT include the Strict-Transport-Security HTTP Response Header in HTTP responses conveyed over a 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.

Also, this processing model assumes that all Domain Names manipulated in this specification's context are already in ASCII Compatible Encoding (ACE) format as specified in [RFC3490]. If this is not the case in some situation, use the operation given in Section 8 "Domain Name ToASCII Conversion Operation" to convert any encountered internationalized Domain Names to ACE format before processing them.

### 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, the UA MUST either:

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

or,

- Update its cached information for the Known HSTS Server 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.

[[TODO2: Decide UA behavior in face of encountering multiple HSTS

headers in a message. Use first header? Last? --JeffH]]

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 Server

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

Otherwise, if the substring does not congruently match a presently known HSTS Server, per the matching procedure specified in Section 7.1.2 "Known HSTS Server Domain Name Matching" below, then the UA MUST note this server as a Known HSTS Server, caching the HSTS Server'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 Server Domain Name Matching

A UA determines whether a Domain Name represents a Known HSTS Server by looking for a match between the query Domain Name and the UA's set of Known HSTS Servers.

1. Compare the query Domain Name string with the Domain Names of the UA's set of Known HSTS Servers. For each Known HSTS Server'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 Server's Domain Name and a right-hand portion of the query Domain Name is found, then the Known HSTS Server'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 Server DN: foo.example.com

At this point, the query Domain Name is ascertained to effectively represent a Known HSTS Server. 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 Server'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 Server.

  For example:

  Query Domain Name: foo.example.com

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

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

* Otherwise, if no matches are found, the query Domain Name does not represent a Known HSTS Server.

## 7.2. URI Loading

Whenever the UA prepares to "load", also known as "dereference", any URI where the host production of the URI [RFC3986] matches that of a Known HSTS Server -- either as a congruent match or as a superdomain match where the superdomain Known HSTS Server has includeSubDomains asserted -- and the URI's scheme is "http", then the UA "MUST" replace the URI scheme with "https" before proceeding with the load.

## 7.3. Errors in Secure Transport Establishment

When connecting to a Known HSTS Server, the UA MUST terminate the connection with no user recourse if there are any errors (e.g. certificate errors), whether "warning" or "fatal" or any other error level, with the underlying secure transport.

### 7.4. HTTP-Equiv <Meta> Element Attribute

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

## 8. Domain Name ToASCII Conversion Operation

This operation converts a string-serialized Domain Name possibly containing arbitrary Unicode characters [Unicode5] into a string-serialized Domain Name in ASCII Compatible Encoding (ACE) format as specified in [RFC3490].

The operation is:

- Apply the IDNA conversion operation (section 4 of [RFC3490]) to the string, selecting the ToASCII operation and setting both the AllowUnassigned and UseSTD3ASCIIRules flags.

## 9. Server Implementation Advice

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 Server. 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 without user recourse, 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. 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 12.4 "Bogus Root CA Certificate Phish plus DNS Cache Poisoning Attack".

## 10. UA Implementation Advice

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

- 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:

- Ability to delete UA's cached HSTS Policy on a per HSTS Server 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 Server 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 Servers.

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

- Ability for users to explicitly declare a given Domain Name as representing a HSTS Server, thus seeding it as a Known HSTS Server before any actual interaction with it. This would help protect against the Section 12.2 "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.

- 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 12.2 "Bootstrap MITM Vulnerability".

  Note: Such a facility complements the preceding described feature.

[[XXX2: These latter items beg the question of having some means of secure web site metadata and policy discovery and acquisition. There is extant work that may be of interest, e.g. the W3C POWDER work, OASIS XRI/XRD work (as well as XRDS-Simple), and "Link-based Resource Descriptor Discovery" (draft-hammer-discovery). --JeffH]]

## 11. Constructing an Effective Request URI

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

The first line of an HTTP request message is specified by the following ABNF ([I-D.ietf-httpbis-p1-messaging] section 4.1):

```
Request-Line   = Method SP request-target SP HTTP-Version CRLF
```

The request-target is following ABNF ([I-D.ietf-httpbis-p1-messaging] section 4.1.2):

```
request-target = "*"
               / absolute-URI
               / ( path-absolute [ "?" query ] )
               / authority
```

Additionally, many HTTP requests contain an additional Host request header field. It is specified by the following ABNF ([I-D.ietf-httpbis-p1-messaging] section 4.1.2):

```
Host = "Host:" OWS Host-v
Host-v = uri-host [ ":" port ]
```

Thus an example HTTP message containing the above header fields is:

```
GET /hello.txt HTTP/1.1
Host: www.example.com
```

Another example is:

```
GET HTTP://www.example.com/hello.txt  HTTP/1.1
```

An HSTS Server constructs the Effective Request URI using the following ABNF grammar (which imports some productions from the above ABNF for Request-Line, request-target, and Host:

```
Effective-Request-URI = absolute-URI-present / path-absolute-form

absolute-URI-present  = absolute-URI

path-absolute-form  = scheme "://" Host-v path-absolute [ "?" query ]
```

where:

scheme is "http" if the request was received over insecure transport, or scheme is "https" if the request was received over secure transport.

For example, if the request message contains a request-target

component that matches the grammar of absolute-URI, then the Effective-Request-URI is simply the value of the absolute-URI component. Otherwise, the Effective-Request-URI is a combination, per the path-absolute-form production, of the Host-v, path-absolute, and query components from the request-target and Host components of the request message.

[[TODO3: This is a first SWAG at this section. Fix/add prose as appropriate, fix ABNF as needed per review. --JeffH]]

## 12. Security Considerations

### 12.1. Denial of Service (DoS)

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

### 12.2. Bootstrap MITM Vulnerability

The bootstrap MITM (Man-In-The-Middle) vulnerability is a vulnerability users and HSTS Servers encounter in the situation where the user manually enters, or follows a link, to a HSTS Server 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.

### 12.3. 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.

### 12.4. 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.

## 13. 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 |

## 14. 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 Server 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 Server 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 Servers 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.

## 15. References

### 15.1. Normative References

[I-D.ietf-httpbis-p1-messaging]
Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., and J. Reschke, "HTTP/1.1, part 1: URIs, Connections, and Message Parsing", draft-ietf-httpbis-p1-messaging-09 (work in progress), March 2010.

[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.

[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.

[Unicode5] The Unicode Consortium, "The Unicode Standard, Version 5.0", Boston, MA, Addison-Wesley ISBN 0-321-48091-0, 2007.

[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>.

[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>.

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

[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>.

[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>.

[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>.

[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. Acknowledgments

The authors thank Michael Barrett, Sid Stamm, Maciej Stachowiak, Andy Steingrubl, Brandon Sterne, Daniel Veditz for their review and contributions.

## Appendix B. Change Log

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
University of California Berkeley

Email: abarth@eecs.berkeley.edu