HTTP Strict Transport Security (HSTS)

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, such as user agent configuration, for example.

Status of This Memo

Copyright Notice

   Copyright (c) 2012 IETF Trust and the persons identified as the
   document authors. All rights reserved.

   This document is subject to BCP 78 and the IETF Trust's Legal
   Provisions Relating to IETF Documents
   (http://trustee.ietf.org/license-info) in effect on the date of
   publication of this document. Please review these documents
   carefully, as they describe your rights and restrictions with respect
   to this document. Code Components extracted from this document must
   include Simplified BSD License text as described in Section 4.e of
   the Trust Legal Provisions and are provided without warranty as
   described in the Simplified BSD License.

Table of Contents

1. Introduction

   HTTP [RFC2616] may be used over various transports, typically the
   Transmission Control Protocol (TCP). However, TCP does not provide
   channel integrity protection, confidentiality, or secure host
   identification. Thus, the Secure Sockets Layer (SSL) protocol
   [RFC6101] and its successor, Transport Layer Security (TLS) [RFC5246]
   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 use either TLS or SSL3, depending
   upon a combination of negotiation with the server and user
   preferences).

   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's host using HTTP or HTTP-over-Secure-Transport. For
   example, cookies ([RFC6265]) 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 subject to
   other rules).

UAs typically announce 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 host'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's
host in the face of such issues. This behavior is sometimes referred
to as "click(ing) through" security [GoodDhamijaEtAl05]
[SunshineEgelmanEtAl09]; thus, it can be described as "click-through
insecurity".

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

Jackson and Barth proposed an approach, in [ForceHTTPS], to enable
web resources to declare that any interactions by UAs with the web
resource must be conducted securely and that any issues with
establishing a secure transport session are to be treated as fatal
and without direct user recourse. The aim is to prevent click-
through insecurity and address other potential threats.

This specification embodies and refines the approach proposed in
[ForceHTTPS]. For example, rather than using a cookie to convey
policy from a web resource's host to a UA, it defines an HTTP
response header field for this purpose. Additionally, a web
resource's host may declare its policy to apply to the entire domain
name subtree rooted at its host name. This enables HTTP Strict
Transport Security (HSTS) to protect so-called "domain cookies",
which are applied to all subdomains of a given web resource's host
name.

This specification also incorporates notions from [JacksonBarth2008]
in that policy is applied on an "entire-host" basis: it applies to
HTTP (only) over any TCP port of the issuing host.

Note that the policy defined by this specification is distinctly
different than the "same-origin policy" defined in "The Web Origin
Concept" [RFC6454]. These differences are summarized in Appendix B.

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. Section 4 defines
   terminology relevant to this document. The HSTS mechanism itself is
   formally specified in Sections 5 through 15.

1.2. Document Conventions

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

2. Overview

   This section discusses the use cases, summarizes the 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 their own, as well as their users', benefit.

2.2. HTTP Strict Transport Security Policy Effects

   The effects of the HSTS Policy, as applied by a conformant UA in
   interactions with a web resource host wielding such policy (known as
   an HSTS Host), are summarized as follows:

   1. UAs transform insecure URI references to an HSTS Host into secure
       URI references before dereferencing them.

   2. The UA terminates any secure transport connection attempts upon
       any and all secure transport errors or warnings.

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. Threats that are addressed, as well
   as threats that are not addressed, are briefly discussed below.

Readers may wish to refer to Section 2 of [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 web browser extension) [Firesheep], that 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 cookies" are those cookies containing the
"Secure" attribute [RFC6265]). 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 certification authority (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 (Shockwave Flash)
movie over an insecure connection (both cascading style sheets and
SWF movies can script the embedding page, to the surprise of many web
developers, plus some browsers do not issue so-called "mixed content
warnings" when SWF files are embedded via insecure connections).
Even if the site's developers carefully scrutinize their login page
for "mixed content", a single insecure embedding anywhere on the
overall site compromises the security of their login page because an
attacker can script (i.e., control) the login page by injecting code
(e.g., a script) into another, insecurely loaded, site page.

NOTE: "Mixed content" as used above (see also Section 5.3 in
        [W3C.REC-wsc-ui-20100812]) refers to the notion termed "mixed
        security context" in this specification and should not be
        confused with the same "mixed content" term used in the
        context of markup languages such as XML and HTML.

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 also lists the
   detailed core requirements that HTTP Strict Transport Security
   addresses, as well as ancillary requirements that are not directly
   addressed.

2.4.1. Overall Requirement

   o Minimize, for web browser users and web site deployers, the risks
      that are derived from passive and active network attackers, web
      site development and deployment bugs, and 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
       accessed 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 retain persistent data about web sites that signal
       strict security policy enablement, for time spans declared by the
       web sites. Additionally, UAs need to cache the "freshest" strict
       security policy information, in order to allow web sites to
       update the information.

   4. UAs need to rewrite 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.

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

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

       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.

   7. UAs need to prevent users from "clicking through" security
       warnings. Halting connection attempts in the face of secure
       transport exceptions is acceptable. See also Section 12.1 ("No
       User Recourse").

   NOTE: A means for uniformly securely meeting the first core
          requirement above is not specifically addressed by this
          specification (see Section 14.6 ("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 12 ("User
          Agent 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" loads (see Section 2.3.1.3).

   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.

   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.

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].

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 [Unicode] 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) [Unicode].

domain name:

   is also referred to as "DNS name" and is 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 a fully qualified domain name.

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 9 ("Constructing an Effective Request URI").

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 conformant host implementing the HTTP server aspects of the
   HSTS Policy. This means that an HSTS Host returns the
   "Strict-Transport-Security" HTTP response header field in its HTTP
   response messages sent over secure transport.

HTTP Strict Transport Security Policy:

   is the name of the combined overall UA- and server-side facets of
   the behavior defined 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 an HSTS Host for which the UA has an HSTS Policy in effect;
   i.e., the UA has noted this host as a Known HSTS Host. See
   Section 8.1.1 ("Noting an HSTS Host - Storage Model") for
   particulars.

Local policy:

   comprises policy rules that deployers specify and that 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.
      See also "Effective Request URI".

   UA:

      is 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].

   unknown HSTS Host:

      is an HSTS Host that the user agent has not noted.

5. HSTS Mechanism Overview

   This section provides an overview of the mechanism by which an HSTS
   Host conveys its HSTS Policy to UAs and how UAs process the HSTS
   Policies received from HSTS Hosts. The mechanism details are
   specified in Sections 6 through 15.

5.1. HSTS Host Declaration

   An HTTP host declares itself an HSTS Host by issuing to UAs an HSTS
   Policy, which is represented by and conveyed via the
   Strict-Transport-Security HTTP response header field over secure
   transport (e.g., TLS). Upon error-free receipt and processing of
   this header by a conformant UA, the UA regards the host as a Known
   HSTS Host.

5.2. HSTS Policy

   An HSTS Policy directs UAs to communicate with a Known HSTS Host only
   over secure transport and specifies policy retention time duration.

   HSTS Policy explicitly overrides the UA processing of URI references,
   user input (e.g., via the "location bar"), or other information that,
   in the absence of HSTS Policy, might otherwise cause UAs to
   communicate insecurely with the Known HSTS Host.

An HSTS Policy may contain an optional directive -- includeSubDomains
-- specifying that this HSTS Policy also applies to any hosts whose
domain names are subdomains of the Known HSTS Host's domain name.

5.3. HSTS Policy Storage and Maintenance by User Agents

UAs store and index HSTS Policies based strictly upon the domain
names of the issuing HSTS Hosts.

This means that UAs will maintain the HSTS Policy of any given HSTS
Host separately from any HSTS Policies issued by any other HSTS Hosts
whose domain names are superdomains or subdomains of the given HSTS
Host's domain name. Only the given HSTS Host can update or can cause
deletion of its issued HSTS Policy. It accomplishes this by sending
Strict-Transport-Security HTTP response header fields to UAs with new
values for policy time duration and subdomain applicability. Thus,
UAs cache the "freshest" HSTS Policy information on behalf of an HSTS
Host. Specifying a zero time duration signals the UA to delete the
HSTS Policy (including any asserted includeSubDomains directive) for
that HSTS Host. See Section 8.1 ("Strict-Transport-Security Response
Header Field Processing") for details. Additionally, Section 6.2
presents examples of Strict-Transport-Security HTTP response header
fields.

5.4. User Agent HSTS Policy Enforcement

When establishing an HTTP connection to a given host, however
instigated, the UA examines its cache of Known HSTS Hosts to see if
there are any with domain names that are superdomains of the given
host's domain name. If any are found, and of those if any have the
includeSubDomains directive asserted, then HSTS Policy applies to the
given host. Otherwise, HSTS Policy applies to the given host only if
the given host is itself known to the UA as an HSTS Host. See
Section 8.3 ("URI Loading and Port Mapping") for details.

6. Syntax

This section defines the syntax of the Strict-Transport-Security HTTP
response header field and its directives, and presents some examples.

Section 7 ("Server Processing Model") then details how hosts employ
this header field to declare their HSTS Policy, and Section 8 ("User
Agent Processing Model") details how user agents process the header
field and apply the HSTS Policy.

6.1. Strict-Transport-Security HTTP Response Header Field

   The Strict-Transport-Security HTTP response header field (STS 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 (Augmented Backus-Naur Form) syntax for the STS header field
   is given below. It is based on the Generic Grammar defined in
   Section 2 of [RFC2616] (which includes a notion of "implied linear
   whitespace", also known as "implied *LWS").

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

      directive   = directive-name ["=" directive-value]
      directive-name   = token
      directive-value   = token | quoted-string

   where:

      token   = <token, defined in [RFC2616], Section 2.2>
      quoted-string = <quoted-string, defined in [RFC2616], Section 2.2>

   The two directives defined in this specification are described below.
   The overall requirements for directives are:

   1. The order of appearance of directives is not significant.

   2. All directives MUST appear only once in an STS header field.
       Directives are either optional or required, as stipulated in
       their definitions.

   3. Directive names are case-insensitive.

   4. UAs MUST ignore any STS header field containing directives, or
       other header field value data, that does not conform to the
       syntax defined in this specification.

   5. If an STS header field contains directive(s) not recognized by
       the UA, the UA MUST ignore the unrecognized directives, and if
       the STS header field otherwise satisfies the above requirements
       (1 through 4), the UA MUST process the recognized directives.

   Additional directives extending the semantic functionality of the STS
   header field can be defined in other specifications, with a registry
   (having an IANA policy definition of IETF Review [RFC5226]) defined
   for them at such time.

      NOTE: Such future directives will be ignored by UAs implementing
             only this specification, as well as by generally
             non-conforming UAs. See Section 14.2 ("Non-Conformant User
             Agent Implications") for further discussion.

6.1.1. The max-age Directive

   The REQUIRED "max-age" directive specifies the number of seconds,
   after the reception of the STS header field, during which the UA
   regards the host (from whom the message was received) as a Known HSTS
   Host. See also Section 8.1.1 ("Noting an HSTS Host - Storage
   Model"). The delta-seconds production is specified in [RFC2616].

   The syntax of the max-age directive's REQUIRED value (after
   quoted-string unescaping, if necessary) is defined as:

    max-age-value = delta-seconds

    delta-seconds = <1*DIGIT, defined in [RFC2616], Section 3.3.2>

      NOTE: A max-age value of zero (i.e., "max-age=0") signals the UA to
             cease regarding the host as a Known HSTS Host, including the
             includeSubDomains directive (if asserted for that HSTS Host).
             See also Section 8.1 ("Strict-Transport-Security Response
             Header Field Processing").

6.1.2. The includeSubDomains Directive

   The OPTIONAL "includeSubDomains" directive is a valueless directive
   which, if present (i.e., it is "asserted"), signals the UA that the
   HSTS Policy applies to this HSTS Host as well as any subdomains of
   the host's domain name.

6.2. Examples

   The HSTS header field below stipulates that the HSTS Policy is to
   remain in effect for one year (there are approximately 31536000
   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 HSTS header field below stipulates that the HSTS Policy is to
   remain in effect for approximately six months and that the policy
   applies to the domain of the issuing HSTS Host and all of its
   subdomains:

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

The max-age directive value can optionally be quoted:

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

The HSTS header field below indicates that the UA must delete the
entire HSTS Policy associated with the HSTS Host that sent the header
field:

    Strict-Transport-Security: max-age=0

The HSTS header field below has exactly the same effect as the one
immediately above because the includeSubDomains directive's presence
in the HSTS header field is ignored when max-age is zero:

    Strict-Transport-Security: max-age=0; includeSubDomains

7. Server Processing Model

   This section describes the processing model that HSTS Hosts
   implement. The model comprises two facets: the first being the
   processing rules for HTTP request messages received over a secure
   transport (TLS [RFC5246] or SSL [RFC6101]; see also Section 14.1
   ("Underlying Secure Transport Considerations")), and the second being
   the processing rules for HTTP request messages received over
   non-secure transports, such as TCP.

7.1. HTTP-over-Secure-Transport Request Type

   When replying to an HTTP request that was conveyed over a secure
   transport, an HSTS Host SHOULD include in its response message an STS
   header field that MUST satisfy the grammar specified above in
   Section 6.1 ("Strict-Transport-Security HTTP Response Header Field").
   If an STS header field is included, the HSTS Host MUST include only
   one such header field.

   Establishing a given host as a Known HSTS Host, in the context of a
   given UA, MAY be accomplished over HTTP, which is in turn running
   over secure transport, by correctly returning (per this
   specification) at least one valid STS header field to the UA. Other
   mechanisms, such as a client-side pre-loaded Known HSTS Host list,
   MAY also be used; e.g., see Section 12 ("User Agent Implementation
   Advice").

   NOTE: Including the STS header field is stipulated as a "SHOULD" in
          order to accommodate various server- and network-side caches
          and load-balancing configurations where it may be difficult to
          uniformly emit STS header fields on behalf of a given HSTS
          Host.

7.2. HTTP Request Type

   If an HSTS Host receives an HTTP request message over a non-secure
   transport, it SHOULD send an HTTP response message containing a
   status code indicating a permanent redirect, such as status code 301
   (Section 10.3.2 of [RFC2616]), and a Location header field value
   containing either the HTTP request's original Effective Request URI
   (see Section 9 ("Constructing an Effective Request URI")) altered as
   necessary to have a URI scheme of "https", or a URI generated
   according to local policy with a URI scheme of "https".

   NOTE: The above behavior is a "SHOULD" rather than a "MUST" due to:

      * Risks in server-side non-secure-to-secure redirects
         [OWASP-TLSGuide].

      * Site deployment characteristics. For example, 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 an HSTS-capable UA that has
         already noted the site as a Known HSTS Host (by whatever means,
         e.g., prior interaction or UA configuration).

   An HSTS Host MUST NOT include the STS header field in HTTP responses
   conveyed over non-secure transport.

8. 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 13
   ("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 10 ("Domain Name IDNA-Canonicalization") prior to
   the processing specified in this section.

      NOTE: [RFC3490] is referenced due to its ongoing relevance to
             actual deployments for the foreseeable future.

   The above assumptions mean that this processing model also
   specifically assumes that appropriate IDNA and Unicode validations
   and character list testing have occurred 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 14.10 ("Internationalized Domain Names")
for rationale and further details.

8.1. Strict-Transport-Security Response Header Field Processing

If an HTTP response, received over a secure transport, includes an
STS header field, conforming to the grammar specified in Section 6.1
("Strict-Transport-Security HTTP Response Header Field"), and there
are no underlying secure transport errors or warnings (see
Section 8.4), the UA MUST either:

o Note the host as a Known HSTS Host if it is not already so noted
   (see Section 8.1.1 ("Noting an HSTS Host - Storage Model")),

or

o Update the UA's cached information for the Known HSTS Host if
   either or both of the max-age and includeSubDomains header field
   value tokens are conveying information different than that already
   maintained by the UA.

   The max-age value is essentially a "time to live" value relative
   to the reception time of the STS header field.

   If the max-age header field value token has a value of zero, the
   UA MUST remove its cached HSTS Policy information (including the
   includeSubDomains directive, if asserted) if the HSTS Host is
   known, or the UA MUST NOT note this HSTS Host if it is not yet
   known.

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

Otherwise:

o If an HTTP response is received over insecure transport, the UA
   MUST ignore any present STS header field(s).

o The UA MUST ignore any STS header fields not conforming to the
   grammar specified in Section 6.1 ("Strict-Transport-Security HTTP
   Response Header Field").

8.1.1. Noting an HSTS Host - Storage Model

   If the substring matching the host production from the Request-URI
   (of the message to which the host responded) 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 Known HSTS
   Host's domain name, per the matching procedure specified in
   Section 8.2 ("Known HSTS Host Domain Name Matching"), 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 directive is asserted or
   not. See also Section 11.2 ("HSTS Policy Expiration Time
   Considerations").

   The UA MUST NOT modify the expiry time or the includeSubDomains
   directive of any superdomain matched Known HSTS Host.

   A Known HSTS Host is "expired" if its cache entry has an expiry date
   in the past. The UA MUST evict all expired Known HSTS Hosts from its
   cache if, at any time, an expired Known HSTS Host exists in the
   cache.

8.2. Known HSTS Host Domain Name Matching

   A given domain name may match a Known HSTS Host's domain name in one
   or both of two fashions: a congruent match, or a superdomain match.
   Alternatively, there may be no match.

   The steps below determine whether there are any matches, and if so,
   of which fashion:

      Compare the given domain name with the domain name of each of the
      UA's unexpired Known HSTS Hosts. For each Known HSTS Host's
      domain name, the comparison is done with the given domain name
      label-by-label (comparing only labels) using an ASCII case-
      insensitive comparison beginning with the rightmost label, and
      continuing right-to-left. See also Section 2.3.2.4 of [RFC5890].

      * Superdomain Match

         If a label-for-label match between an entire Known HSTS Host's
         domain name and a right-hand portion of the given domain name
         is found, then this Known HSTS Host's domain name is a
         superdomain match for the given domain name. There could be
         multiple superdomain matches for a given domain name.

For example:

Given domain name (DN):   qaz.bar.foo.example.com

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

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

*   Congruent Match

If a label-for-label match between a Known HSTS Host's domain
name and the given domain name is found -- i.e., there are no
further labels to compare -- then the given domain name
congruently matches this Known HSTS Host.

For example:

Given domain name: foo.example.com

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

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

8.3. URI Loading and Port Mapping

Whenever the UA prepares to "load" (also known as "dereference") any
"http" URI [RFC3986] (including when following HTTP redirects
[RFC2616]), the UA MUST first determine whether a domain name is
given in the URI and whether it matches a Known HSTS Host, using
these steps:

1. Extract from the URI any substring described by the host
component of the authority component of the URI.

2. If the substring is null, then there is no match with any Known
HSTS Host.

3. Else, if the substring is non-null and syntactically matches the
IP-literal or IPv4address productions from Section 3.2.2 of
[RFC3986], then there is no match with any Known HSTS Host.

4. Otherwise, the substring is a given domain name, which MUST be
    matched against the UA's Known HSTS Hosts using the procedure in
    Section 8.2 ("Known HSTS Host Domain Name Matching").

5. If, when performing domain name matching any superdomain match
    with an asserted includeSubDomains directive is found, or, if no
    superdomain matches with asserted includeSubDomains directives
    are found and a congruent match is found (with or without an
    asserted includeSubDomains directive), then before proceeding
    with the load:

       The UA MUST replace the URI scheme with "https" [RFC2818], and

       if the URI contains an explicit port component 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.

       NOTE: These steps ensure that the HSTS Policy applies to HTTP
               over any TCP port of an HSTS Host.

    NOTE: In the case where an explicit port is provided (and to a
            lesser extent with subdomains), it is reasonably likely that
            there is actually an HTTP (i.e., non-secure) server running on
            the specified port and that an HTTPS request will thus fail
            (see item 6 in Appendix A ("Design Decision Notes")).

8.4. Errors in Secure Transport Establishment

   When connecting to a Known HSTS Host, the UA MUST terminate the
   connection (see also Section 12 ("User Agent Implementation Advice"))
   if there are any errors, whether "warning" or "fatal" or any other
   error level, with the underlying secure transport. For example, this
   includes any errors found in certificate validity checking that UAs
   employ, such as via Certificate Revocation Lists (CRLs) [RFC5280], or
   via the Online Certificate Status Protocol (OCSP) [RFC2560], as well
   as via TLS server identity checking [RFC6125].

8.5. HTTP-Equiv <Meta> Element Attribute

   UAs MUST NOT heed http-equiv="Strict-Transport-Security" attribute
   settings on <meta> elements [W3C.REC-html401-19991224] in received
   content.

8.6. Missing Strict-Transport-Security Response Header Field

   If a UA receives HTTP responses from a Known HSTS Host over a secure
   channel but the responses are missing the STS header field, the UA
   MUST continue to treat the host as a Known HSTS Host until the
   max-age value for the knowledge of that Known HSTS Host is reached.
   Note that the max-age value could be effectively infinite for a given
   Known HSTS Host. For example, this would be the case if the Known
   HSTS Host is part of a pre-configured list that is implemented such
   that the list entries never "age out".

9. 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 absoluteURI for the target
   resource; instead, the URI needs to be inferred from the Request-URI,
   Host header field, and connection context ([RFC2616], Sections 3.2.1,
   5.1.2, and 5.2). The result of this process is called the "effective
   request URI (ERU)". The "target resource" is the resource identified
   by the effective request URI.

9.1. 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]

9.2. Determining the Effective Request 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 "/".

9.2.1. Effective Request 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".

10. Domain Name IDNA-Canonicalization

   An IDNA-canonicalized domain name is the output string generated by
   the following steps. The input is a putative domain name string
   ostensibly composed of any combination of "A-labels", "U-labels", and
   "NR-LDH labels" (see Section 2 of [RFC5890]) concatenated using some
   separator character (typically ".").

   1.   Convert the input putative domain name string to an order-
        preserving sequence of individual label strings.

   2.   When implementing IDNA2008, convert, validate, and test each
        A-label and U-label found among the sequence of individual label
        strings, using the procedures defined in Sections 5.3 through 5.5
        of [RFC5891].

        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 [RFC3490]).

   3.   If no errors occurred during the foregoing step, concatenate all
        the labels in the sequence, in order, into a string, separating
        each label from the next with a %x2E (".") character. The
        resulting string, known as an IDNA-canonicalized domain name, is
        appropriate for use in the context of Section 8 ("User Agent
        Processing Model").

        Otherwise, errors occurred. The input putative domain name
        string was not successfully IDNA-canonicalized. Invokers of this
        procedure should attempt appropriate error recovery.

   See also Sections 13 ("Internationalized Domain Names for
   Applications (IDNA): Dependency and Migration") and 14.10
   ("Internationalized Domain Names") of this specification for further
   details and considerations.

11. Server Implementation and Deployment Advice

   This section is non-normative.

11.1. Non-Conformant User Agent Considerations

   Non-conformant UAs ignore the Strict-Transport-Security header field;
   thus, non-conformant user agents do not address the threats described
   in Section 2.3.1 ("Threats Addressed"). Please refer to Section 14.2
   ("Non-Conformant User Agent Implications") for further discussion.

11.2. 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, or whether they are setting an expiry time that is a
   fixed point in time.

   The "constant value into the future" approach can be accomplished by
   constantly sending the same max-age value to UAs.

   For example, a max-age value of 7776000 seconds is 90 days:

      Strict-Transport-Security: max-age=7776000

   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 "fixed point in time" approach can be accomplished by sending
   max-age values that represent the remaining time until the desired
   expiry time. This would require the HSTS Host to send a newly
   calculated max-age value in each HTTP response.

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

   Additionally, server implementers should consider employing a default
   max-age value of zero in their deployment configuration systems.
   This will require deployers to willfully set max-age in order to have
   UAs enforce the HSTS Policy for their host and will protect them from
   inadvertently enabling HSTS with some arbitrary non-zero duration.

11.3. Using HSTS in Conjunction with Self-Signed Public-Key
       Certificates

   If all four of the following conditions are true...

   o a web site/organization/enterprise is generating its own secure
      transport public-key certificates for web sites, and

   o that organization's root certification authority (CA) certificate
      is not typically embedded by default in browser and/or operating
      system CA certificate stores, and

   o HSTS Policy is enabled on a host identifying itself using a
      certificate signed by the organization's CA (i.e., a "self-signed
      certificate"), and

   o this certificate does not match a usable TLS certificate
      association (as defined by Section 4 of the TLSA protocol
      specification [RFC6698]),

   ...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 wishes to employ its own CA, and self-
   signed certificates, in concert with HSTS, it can do so by deploying
   its root CA certificate to its users' browsers or operating system CA
   root certificate stores. It can also, in addition or instead,
   distribute to its 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
   its root CA certificate is installed in the browsers, it may employ
   HSTS Policy on its site(s).

   Alternatively, that organization can deploy the TLSA protocol; all
   browsers that also use TLSA will then be able to trust the
   certificates identified by usable TLS certificate associations as
   denoted via TLSA.

   NOTE: Interactively distributing root CA certificates 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 14.8 ("Bogus Root CA
          Certificate Phish plus DNS Cache Poisoning Attack"). Thus,
          care should be taken in the manner in which such certificates
          are distributed and installed on users' systems and browsers.

11.4. Implications of includeSubDomains

   The includeSubDomains directive has practical implications meriting
   careful consideration; two example scenarios are:

   o An HSTS Host offers unsecured HTTP-based services on alternate
      ports or at various subdomains of its HSTS Host domain name.

   o Distinct web applications are offered at distinct subdomains of an
      HSTS Host, such that UAs often interact directly with these
      subdomain web applications without having necessarily interacted
      with a web application offered at the HSTS Host's domain name (if
      any).

   The sections below discuss each of these scenarios in turn.

11.4.1. Considerations for Offering Unsecured HTTP Services at
         Alternate Ports or Subdomains of an HSTS Host

   For example, certification authorities often offer their CRL
   distribution and OCSP services [RFC2560] over plain HTTP, and
   sometimes at a subdomain of a publicly available web application that
   may be secured by TLS/SSL. For example, <https://ca.example.com/> is
   a publicly available web application for "Example CA", a
   certification authority. Customers use this web application to
   register their public keys and obtain certificates. "Example CA"
   generates certificates for customers containing
   <http://crl-and-ocsp.ca.example.com/> as the value for the "CRL
   Distribution Points" and "Authority Information Access:OCSP"
   certificate fields.

   If ca.example.com were to issue an HSTS Policy with the
   includeSubDomains directive, then HTTP-based user agents implementing
   HSTS that have interacted with the ca.example.com web application
   would fail to retrieve CRLs and fail to check OCSP for certificates,
   because these services are offered over plain HTTP.

   In this case, Example CA can either:

   o not use the includeSubDomains directive, or

   o ensure that HTTP-based services offered at subdomains of
      ca.example.com are also uniformly offered over TLS/SSL, or

   o offer plain HTTP-based services at a different domain name, e.g.,
      crl-and-ocsp.ca.example.NET, or

   o utilize an alternative approach to distributing certificate status
      information, obviating the need to offer CRL distribution and OCSP
      services over plain HTTP (e.g., the "Certificate Status Request"
      TLS extension [RFC6066], often colloquially referred to as "OCSP
      Stapling").

   NOTE: The above points are expressly only an example and do not
          purport to address all the involved complexities. For
          instance, there are many considerations -- on the part of CAs,
          certificate deployers, and applications (e.g., browsers) --
          involved in deploying an approach such as "OCSP Stapling".
          Such issues are out of scope for this specification.

11.4.2. Considerations for Offering Web Applications at Subdomains of
         an HSTS Host

   In this scenario, an HSTS Host declares an HSTS Policy with an
   includeSubDomains directive, and there also exist distinct web
   applications offered at distinct subdomains of the HSTS Host such
   that UAs often interact directly with these subdomain web
   applications without having necessarily interacted with the HSTS
   Host. In such a case, the UAs will not receive or enforce the HSTS
   Policy.

   For example, the HSTS Host is "example.com", and it is configured to
   emit the STS header field with the includeSubDomains directive.
   However, example.com's actual web application is addressed at
   "www.example.com", and example.com simply redirects user agents to
   "https://www.example.com/".

   If the STS header field is only emitted by "example.com" but UAs
   typically bookmark -- and links (from anywhere on the web) are
   typically established to -- "www.example.com", and "example.com" is
   not contacted directly by all user agents in some non-zero percentage
   of interactions, then some number of UAs will not note "example.com"
   as an HSTS Host, and some number of users of "www.example.com" will
   be unprotected by HSTS Policy.

   To address this, HSTS Hosts should be configured such that the STS
   header field is emitted directly at each HSTS Host domain or
   subdomain name that constitutes a well-known "entry point" to one's
   web application(s), whether or not the includeSubDomains directive is
   employed.

Thus, in our example, if the STS header field is emitted from both
"example.com" and "www.example.com", this issue will be addressed.
Also, if there are any other well-known entry points to web
applications offered by "example.com", such as "foo.example.com",
they should also be configured to emit the STS header field.

12. User Agent Implementation Advice

   This section is non-normative.

   In order to provide users and web sites more effective protection, as
   well as controls for managing their UA's caching of HSTS Policy, UA
   implementers should consider including features such as the
   following:

12.1. No User Recourse

   Failing secure connection establishment on any warnings or errors
   (per Section 8.4 ("Errors in Secure Transport Establishment")) should
   be done with "no user recourse". This means that the user should not
   be presented with a 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 retry.

   Essentially, "any warnings or errors" means anything that would cause
   the UA implementation to announce 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 issues an HSTS Policy, then it is
   implicitly opting into the "no user recourse" approach, whereby all
   certificate errors or warnings cause a connection termination, with
   no chance to "fool" users into making the wrong decision and
   compromising themselves.

12.2. User-Declared HSTS Policy

   A user-declared HSTS Policy is the ability for users to explicitly
   declare a given domain name as representing an HSTS Host, thus
   seeding it as a Known HSTS Host before any actual interaction with
   it. This would help protect against the bootstrap MITM vulnerability
   as discussed in Section 14.6 ("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, an "HSTS pre-loaded
             list" feature (see Section 12.3).

12.3. HSTS Pre-Loaded List

   An HSTS pre-loaded list is a facility whereby web site administrators
   can have UAs pre-configured with HSTS Policy for their site(s) by the
   UA vendor(s) -- a so-called "pre-loaded list" -- in a manner similar
   to how root CA certificates are embedded in browsers "at the
   factory". This would help protect against the bootstrap MITM
   vulnerability (Section 14.6).

      NOTE: Such a facility would complement a "user-declared HSTS Policy"
             feature (Section 12.2).

12.4. Disallow Mixed Security Context Loads

   "Mixed security context" loads happen when a web application
   resource, fetched by the UA over a secure transport, subsequently
   causes the fetching of one or more other resources without using
   secure transport. This is also generally referred to as "mixed
   content" loads (see Section 5.3 ("Mixed Content") in
   [W3C.REC-wsc-ui-20100812]) but should not be confused with the same
   "mixed content" term that is also used in the context of markup
   languages such as XML and HTML.

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

12.5. HSTS Policy Deletion

   HSTS Policy deletion is the ability to delete a 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 that users
             get used to doing as a matter of course: e.g., just "clicking

through" in order to get work done. Also, implementations
need to guard against allowing an attacker to inject code,
e.g., ECMAscript, into the UA that silently and
programmatically removes entries from the UA's cache of Known
HSTS Hosts.

13. 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. In order to facilitate their
   IDNA transition, user agents SHOULD implement IDNA2008 [RFC5890] and
   MAY implement [RFC5895] (see also Section 7 of [RFC5894]) or [UTS46].
   If a user agent does not implement IDNA2008, the user agent MUST
   implement IDNA2003.

14. Security Considerations

   This specification concerns the expression, conveyance, and
   enforcement of the HSTS Policy. The overall HSTS Policy threat
   model, including addressed and unaddressed threats, is given in
   Section 2.3 ("Threat Model").

   Additionally, the sections below discuss operational ramifications of
   the HSTS Policy, provide feature rationale, discuss potential HSTS
   Policy misuse, and highlight some known vulnerabilities in the HSTS
   Policy regime.

14.1. Underlying Secure Transport Considerations

   This specification is fashioned to be independent of the secure
   transport underlying HTTP. However, the threat analysis and
   requirements in Section 2 ("Overview") in fact presume TLS or SSL as
   the underlying secure transport. Thus, employment of HSTS in the
   context of HTTP running over some other secure transport protocol
   would require assessment of that secure transport protocol's security

model in conjunction with the specifics of how HTTP is layered over
it in order to assess HSTS's subsequent security properties in that
context.

14.2. Non-Conformant User Agent Implications

Non-conformant user agents ignore the Strict-Transport-Security
header field; thus, non-conformant user agents do not address the
threats described in Section 2.3.1 ("Threats Addressed").

This means that the web application and its users wielding
non-conformant UAs will be vulnerable to both of the following:

o Passive network attacks due to web site development and deployment
   bugs:

      For example, if the web application contains any insecure
      references (e.g., "http") to the web application server, and if
      not all of its cookies are flagged as "Secure", then its
      cookies will be vulnerable to passive network sniffing and,
      potentially, subsequent misuse of user credentials.

o Active network attacks:

      For example, if an attacker is able to place a "man in the
      middle", secure transport connection attempts will likely yield
      warnings to the user, but without HSTS Policy being enforced,
      the present common practice is to allow the user to "click
      through" and proceed. This renders the user and possibly the
      web application open to abuse by such an attacker.

This is essentially the status quo for all web applications and their
users in the absence of HSTS Policy. Since web application providers
typically do not control the type or version of UAs their web
applications interact with, the implication is that HSTS Host
deployers must generally exercise the same level of care to avoid web
site development and deployment bugs (see Section 2.3.1.3) as they
would if they were not asserting HSTS Policy.

14.3. Ramifications of HSTS Policy Establishment Only over Error-Free
       Secure Transport

The user agent processing model defined in Section 8 ("User Agent
Processing Model") stipulates that a host is initially noted as a
Known HSTS Host, or that updates are made to a Known HSTS Host's
cached information, only if the UA receives the STS header field over
a secure transport connection having no underlying secure transport
errors or warnings.

The rationale behind this is that if there is a "man in the middle"
(MITM) -- whether a legitimately deployed proxy or an illegitimate
entity -- it could cause various mischief (see also Appendix A
("Design Decision Notes") item 3, as well as Section 14.6 ("Bootstrap
MITM Vulnerability")); for example:

o Unauthorized notation of the host as a Known HSTS Host,
   potentially leading to a denial-of-service situation if the host
   does not uniformly offer its services over secure transport (see
   also Section 14.5 ("Denial of Service")).

o Resetting the time to live for the host's designation as a Known
   HSTS Host by manipulating the max-age header field parameter value
   that is returned to the UA. If max-age is returned as zero, this
   will cause the host to cease being regarded as a Known HSTS Host
   by the UA, leading to either insecure connections to the host or
   possibly denial of service if the host delivers its services only
   over secure transport.

However, this means that if a UA is "behind" a MITM non-transparent
TLS proxy -- within a corporate intranet, for example -- and
interacts with an unknown HSTS Host beyond the proxy, the user could
possibly be presented with the legacy secure connection error
dialogs. Even if the risk is accepted and the user "clicks through",
the host will not be noted as an HSTS Host. Thus, as long as the UA
is behind such a proxy, the user will be vulnerable and will possibly
be presented with the legacy secure connection error dialogs for
as-yet unknown HSTS Hosts.

Once the UA successfully connects to an unknown HSTS Host over error-
free secure transport, the host will be noted as a Known HSTS Host.
This will result in the failure of subsequent connection attempts
from behind interfering proxies.

The above discussion relates to the recommendation in Section 12
("User Agent Implementation Advice") that the secure connection be
terminated with "no user recourse" whenever there are warnings and
errors and the host is a Known HSTS Host. Such a posture protects
users from "clicking through" security warnings and putting
themselves at risk.

14.4. 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 directive 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 that the attacker has
managed to register in the DNS and point at an HTTP server under the
attacker's control, then:

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

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 presented (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.

14.5. Denial of Service

HSTS could be used to mount certain forms of Denial-of-Service (DoS)
attacks against web sites. A DoS attack is an attack in which one or
more network entities target a victim entity and attempt to prevent
the victim from doing useful work. This section discusses such
scenarios in terms of HSTS, though this list is not exhaustive. See
also [RFC4732] for a discussion of overall Internet DoS
considerations.

o Web applications available over HTTP

   There is an opportunity for perpetrating DoS attacks with web
   applications (or critical portions of them) that are available
   only over HTTP without secure transport, if attackers can cause
   UAs to set HSTS Policy for such web applications' host(s).

   This is because once the HSTS Policy is set for a web
   application's host in a UA, the UA will only use secure transport
   to communicate with the host. If the host is not using secure

transport or is not using it for critical portions of its web
application, then the web application will be rendered unusable
for the UA's user.

NOTE: This is a use case for UAs to offer an "HSTS Policy
        deletion" feature as noted in Section 12.5 ("HSTS Policy
        Deletion").

An HSTS Policy can be set for a victim host in various ways:

* If the web application has an HTTP response splitting
   vulnerability [CWE-113] (which can be abused in order to
   facilitate "HTTP header injection").

* If an attacker can spoof a redirect from an insecure victim
   site, e.g., <http://example.com/> to <https://example.com/>,
   where the latter is attacker-controlled and has an apparently
   valid certificate. In this situation, the attacker can then
   set an HSTS Policy for example.com and also for all subdomains
   of example.com.

* If an attacker can convince users to manually configure HSTS
   Policy for a victim host. This assumes that their UAs offer
   such a capability (see Section 12 ("User Agent Implementation
   Advice")). Alternatively, if such UA configuration is
   scriptable, then an attacker can cause UAs to execute his
   script and set HSTS Policies for whichever desired domains.

o Inadvertent use of includeSubDomains

   The includeSubDomains directive instructs UAs to automatically
   regard all subdomains of the given HSTS Host as Known HSTS Hosts.
   If any such subdomains do not support properly configured secure
   transport, then they will be rendered unreachable from such UAs.

14.6. Bootstrap MITM Vulnerability

   Bootstrap MITM (man-in-the-middle) vulnerability is a vulnerability
   that users and HSTS Hosts encounter in the situation where the user
   manually enters, or follows a link, to an unknown HSTS Host using an
   "http" URI rather than an "https" URI. Because the UA uses an
   insecure channel in the initial attempt to interact with the
   specified server, such an initial interaction is vulnerable to
   various attacks (see Section 5.3 of [ForceHTTPS]).

   NOTE: There are various features/facilities that UA implementations
           may employ in order to mitigate this vulnerability. Please
           see Section 12 ("User Agent Implementation Advice").

14.7. Network Time Attacks

   Active network attacks can subvert network time protocols (such as
   the Network Time Protocol (NTP) [RFC5905]) -- making HSTS less
   effective against clients that trust NTP or lack a real time clock.
   Network time attacks are beyond the scope of this specification.
   Note that modern operating systems use NTP by default. See also
   Section 2.10 of [RFC4732].

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

   An attacker could conceivably obtain users' login credentials
   belonging to a victim HSTS-protected web application via a bogus root
   CA certificate phish plus DNS cache poisoning attack.

   For example, the attacker could first convince users of a victim web
   application (which is protected by HSTS Policy) to install the
   attacker's version of a root CA certificate purporting (falsely) to
   represent the CA of the victim web application. This might be
   accomplished by sending the users a phishing email message with a
   link to such a certificate, which their browsers may offer to install
   if clicked on.

   Then, if the attacker can perform an attack on the users' DNS
   servers, (e.g., via cache poisoning) and turn on HSTS Policy for
   their fake web application, the affected users' browsers would access
   the attacker's web application rather than the legitimate web
   application.

   This type of attack leverages vectors that are outside of the scope
   of HSTS. However, the feasibility of such threats can be mitigated
   by including in a web application's overall deployment approach
   appropriate employment, in addition to HSTS, of security facilities
   such as DNS Security Extensions [RFC4033], plus techniques to block
   email phishing and fake certificate injection.

14.9. Creative Manipulation of HSTS Policy Store

   Since an HSTS Host may select its own host name and subdomains
   thereof, and this information is cached in the HSTS Policy store of
   conforming UAs, it is possible for those who control one or more HSTS
   Hosts to encode information into domain names they control and cause
   such UAs to cache this information as a matter of course in the
   process of noting the HSTS Host. This information can be retrieved
   by other hosts through cleverly constructed and loaded web resources,
   causing the UA to send queries to (variations of) the encoded domain
   names. Such queries can reveal whether the UA had previously visited
   the original HSTS Host (and subdomains).

Such a technique could potentially be abused as yet another form of "web tracking" [WebTracking].

14.10. 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 10 ("Domain Name IDNA-Canonicalization")). These steps are necessary in order to avoid various potentially compromising situations.

In brief, examples of issues that could stem from lack of careful and consistent Unicode and IDNA validations include 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.

15. IANA Considerations

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

      Header field name: Strict-Transport-Security
      Applicable protocol: http
      Status: standard
      Author/Change controller: IETF
      Specification document(s):   this one

16. References

16.1. Normative References

   [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.

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

   [RFC5246]   Dierks, T. and E. Rescorla, "The Transport Layer Security
                (TLS) Protocol Version 1.2", RFC 5246, August 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.

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

   [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication
              of Named Entities (DANE) Transport Layer Security (TLS)
              Protocol: TLSA", RFC 6698, August 2012.

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

   [Unicode] The Unicode Consortium, "The Unicode Standard",
              <http://www.unicode.org/versions/latest/>.

   [W3C.REC-html401-19991224]
              Raggett, D., Le Hors, A., and I. Jacobs, "HTML 4.01
              Specification", World Wide Web Consortium Recommendation
              REC-html401-19991224, December 1999,
              <http://www.w3.org/TR/1999/REC-html401-19991224/>.

16.2. 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://dl.acm.org/citation.cfm?id=1514286>.

   [CWE-113] "CWE-113: Improper Neutralization of CRLF Sequences in
              HTTP Headers ('HTTP Response Splitting')", Common Weakness
              Enumeration <http://cwe.mitre.org/>, The Mitre
              Corporation <http://www.mitre.org/>,
              <http://cwe.mitre.org/data/definitions/113.html>.

   [Firesheep]
              Various, "Firesheep", Wikipedia Online, ongoing, <https://
              secure.wikimedia.org/wikipedia/en/w/
              index.php?title=Firesheep&oldid=517474182>.

   [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://www.law.berkeley.edu/files/
             Spyware_at_the_Gate.pdf>.

   [HTTP1_1-UPD]
             Fielding, R., Ed., and J. Reschke, Ed., "Hypertext
             Transfer Protocol (HTTP/1.1): Message Syntax and Routing",
             Work in Progress, October 2012.

   [JacksonBarth2008]
             Jackson, C. and A. Barth, "Beware of Finer-Grained
             Origins", Web 2.0 Security and Privacy Workshop, Oakland,
             CA, USA, 2008,
             <http://seclab.stanford.edu/websec/origins/fgo.pdf>.

   [OWASP-TLSGuide]
             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>.

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

   [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.

   [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S.
             Rose, "DNS Security Introduction and Requirements",
             RFC 4033, March 2005.

   [RFC4732] Handley, M., Rescorla, E., and IAB, "Internet Denial-of-
             Service Considerations", RFC 4732, December 2006.

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

   [RFC5226]   Narten, T. and H. Alvestrand, "Guidelines for Writing an
               IANA Considerations Section in RFCs", BCP 26, RFC 5226,
               May 2008.

   [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.

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

   [RFC5905]   Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network
               Time Protocol Version 4: Protocol and Algorithms
               Specification", RFC 5905, June 2010.

   [RFC6066]   Eastlake, D., "Transport Layer Security (TLS) Extensions:
               Extension Definitions", RFC 6066, January 2011.

   [RFC6101]   Freier, A., Karlton, P., and P. Kocher, "The Secure
               Sockets Layer (SSL) Protocol Version 3.0", RFC 6101,
               August 2011.

   [RFC6125]   Saint-Andre, P. and J. Hodges, "Representation and
               Verification of Domain-Based Application Service Identity
               within Internet Public Key Infrastructure Using X.509
               (PKIX) Certificates in the Context of Transport Layer
               Security (TLS)", RFC 6125, March 2011.

   [RFC6265]   Barth, A., "HTTP State Management Mechanism", RFC 6265,
               April 2011.

   [RFC6454]   Barth, A., "The Web Origin Concept", RFC 6454,
               December 2011.

   [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, August 2009, <http://
               www.usenix.org/events/sec09/tech/full_papers/
               sunshine.pdf>.

   [W3C.REC-wsc-ui-20100812]
              Roessler, T. and A. Saldhana, "Web Security Context: User
              Interface Guidelines", World Wide Web Consortium
              Recommendation REC-wsc-ui-20100812, August 2010,
              <http://www.w3.org/TR/2010/REC-wsc-ui-20100812>.

   [WebTracking]
              Schmucker, N., "Web Tracking", SNET2 Seminar Paper
              - Summer Term, 2011, <http://www.snet.tu-berlin.de/
              fileadmin/fg220/courses/SS11/snet-project/
              web-tracking_schmuecker.pdf>.

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" (also known as "mixed content loads") are handled, due to
        UA implementation considerations as well as classification
        difficulties.

    3. An HSTS Host may update UA notions of HSTS Policy via new HSTS
        header field parameter 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 erroneous HSTS Policy,
        such as a multi-year max-age value, and/or an incorrect
        includeSubDomains directive. If the HSTS Host couldn't correct
        such errors over protocol, it would require some form of
        annunciation to users and manual intervention on the users' part,
        which could be a non-trivial problem for both web application
        providers and their users.

    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.

    5. The max-age approach of having the HSTS Host provide a simple
        integer number of seconds for a cached HSTS Policy time-to-live
        value, as opposed to an approach of stating an expiration time in
        the future, was chosen for various reasons. Amongst the reasons
        are no need for clock synchronization, no need to define date and
        time value syntaxes (specification simplicity), and
        implementation simplicity.

    6. In determining whether port mapping was to be employed, the
        option of merely refusing to dereference any URL with an explicit
        port was considered. It was felt, though, that the possibility
        that the URI to be dereferenced is incorrect (and there is indeed
        a valid HTTPS server at that port) is worth the small cost of
        possibly wasted HTTPS fetches to HTTP servers.

Appendix B. Differences between HSTS Policy and Same-Origin Policy

   HSTS Policy has the following primary characteristics:

      HSTS Policy stipulates requirements for the security
      characteristics of UA-to-host connection establishment, on a
      per-host basis.

      Hosts explicitly declare HSTS Policy to UAs. Conformant UAs are
      obliged to implement hosts' declared HSTS Policies.

      HSTS Policy is conveyed over protocol from the host to the UA.

      The UA maintains a cache of Known HSTS Hosts.

      UAs apply HSTS Policy whenever making an HTTP connection to a
      Known HSTS Host, regardless of host port number; i.e., it applies
      to all ports on a Known HSTS Host. Hosts are unable to affect
      this aspect of HSTS Policy.

      Hosts may optionally declare that their HSTS Policy applies to all
      subdomains of their host domain name.

   In contrast, the Same-Origin Policy (SOP) [RFC6454] has the following
   primary characteristics:

      An origin is the scheme, host, and port of a URI identifying a
      resource.

      A UA may dereference a URI, thus loading a representation of the
      resource the URI identifies. UAs label resource representations
      with their origins, which are derived from their URIs.

      The SOP refers to a collection of principles, implemented within
      UAs, governing the isolation of and communication between resource
      representations within the UA, as well as resource
      representations' access to network resources.

   In summary, although both HSTS Policy and SOP are enforced by UAs,
   HSTS Policy is optionally declared by hosts and is not origin-based,
   while the SOP applies to all resource representations loaded from all
   hosts by conformant UAs.

Appendix C. Acknowledgments

   The authors thank Devdatta Akhawe, Michael Barrett, Ben Campbell,
   Tobias Gondrom, Paul Hoffman, Murray Kucherawy, Barry Leiba, James
   Manger, Alexey Melnikov, Haevard Molland, Yoav Nir, Yngve N.
   Pettersen, Laksh Raghavan, Marsh Ray, Julian Reschke, Eric Rescorla,
   Tom Ritter, Peter Saint-Andre, Brian Smith, Robert Sparks, Maciej
   Stachowiak, Sid Stamm, Andy Steingrubl, Brandon Sterne, Martin
   Thomson, Daniel Veditz, and Jan Wrobel, as well as all the websec
   working group participants and others for their various reviews and
   helpful contributions.

   Thanks to Julian Reschke for his elegant rewriting of the effective
   request URI text, which he did when incorporating the ERU notion into
   the updates to HTTP/1.1 [HTTP1_1-UPD]. Subsequently, the ERU text in
   this spec was lifted from Julian's work in the updated HTTP/1.1
   (part 1) specification and adapted to the [RFC2616] ABNF.

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/