

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2013

J. Hodges
PayPal
Jul 2012

Web Security Framework: Problem Statement and Requirements
draft-hodges-websec-framework-reqs-02

Abstract

Web-based malware and attacks are proliferating rapidly on the Internet. New web security mechanisms are also rapidly growing in number, although in an incoherent fashion. This document provides a brief overview of the present situation and the various seemingly piece-wise approaches being taken to mitigate the threats. It then provides an overview of requirements as presently being expressed by the community in various online and face-to-face discussions.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 2, 2013.

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	3
1.1. Where to Discuss This Draft	4
2. Document Conventions	4
3. Overall Constraints	5
4. Overall Requirements	6
5. Attacks and Threats to Address	8
5.1. Attacks	8
5.2. Threats	8
6. Use Cases	9
7. Detailed Functional Requirements	10
8. Extant Policies to Coalesce	18
9. Example Concrete Approaches	19
10. Security Considerations	19
11. References	19
12. Informative References	23
Author's Address	23

1. Introduction

Over the past few years, we have seen a proliferation of AJAX-based web applications (AJAX being shorthand for asynchronous JavaScript and XML), as well as Rich Internet Applications (RIAs), based on so-called Web 2.0 technologies. These applications bring both luscious eye-candy and convenient functionality--e.g. social networking--to their users, making them quite compelling. At the same time, we are seeing an increase in attacks against these applications and their underlying technologies [1]. The latter include (but aren't limited to) Cross-Site-Request Forgery (CSRF) -based attacks [2], content-sniffing cross-site-scripting (XSS) attacks [3], attacks against browsers supporting anti-XSS policies [4], clickjacking attacks [5], malvertising attacks [6], as well as man-in-the-middle (MITM) attacks against "secure" (e.g. Transport Layer Security (TLS/SSL)-based [7]) web sites along with distribution of the tools to carry out such attacks (e.g. sslstrip) [8].

During the same time period we have also witnessed the introduction of new web security indicators, techniques, and policy communication mechanisms sprinkled throughout the various layers of the Web and HTTP. We have a new cookie security flag called HTTPOnly [9]. We have the anti-clickjacking X-Frame-Options HTTP header [10], the Strict-Transport-Security HTTP header [11], anti-CSRF headers (e.g. Origin) [12], an anti-sniffing header (X-Content-Type-Options: nosniff) [13], various approaches to content restrictions [14] [15] and notably Mozilla Content Security Policy (CSP; conveyed via a HTTP header) [16], the W3C's Cross-Origin Resource Sharing (CORS; also conveyed via a HTTP header) [17], as well as RIA security controls such as the crossdomain.xml file used to express a site's Adobe Flash security policy [18]. There's also the Application Boundaries Enforcer (ABE) [19], included as a part of NoScript [20], a popular Mozilla Firefox security extension. Sites can express their ABE rule-set at a well-known web address for downloading by individual clients [21], similarly to Flash's crossdomain.xml. Amidst this haphazard collage of new security mechanisms at least one browser vendor has even devised a new HTTP header that disables one of their newly created security features: witness the X-XSS-Protection header that disables the new anti-XSS features [22] in Microsoft's Internet Explorer 8 (IE8).

Additionally, there are various proposals aimed at addressing other facets of inherent web vulnerabilities, for example: JavaScript postMessage-based mashup communications [23], hypertext isolation techniques [24], and service security policies advertised via the Domain Name System (DNS) [25]. Going even further, there are efforts to redesign web browser architectures [26], of which Google Chrome and IE8 are deployed examples. An even more radical approach is

exhibited in the Gazelle Web Browser [27], which features a browser kernel embodied in a multi-principal OS construction providing cross-principal protection and fair sharing of all system resources.

Not to be overlooked is the fact that even though there is a plethora of "standard" browser security features--e.g. the Same Origin Policy (SOP), network-related restrictions, rules for third-party cookies, content-handling mechanisms, etc. [28]--they are not implemented uniformly in today's various popular browsers and RIA frameworks [29]. This makes life even harder for web site administrators in that allowances must be made in site security posture and approaches in consideration of which browser a user may be wielding at any particular time.

Although industry and researchers collectively are aware of all the above issues, we observe that the responses to date have been issue-specific and uncoordinated. What we are ending up with looks perhaps similar to Frankenstein's monster [30]--a design with noble intents but whose final execution is an almost-random amalgamation of parts that do not work well together. It can even cause destruction on its own [31].

Thus, the goal of this document is to define the requirements for a common framework expressing security constraints on HTTP interactions. Functionally, this framework should be general enough that it can be used to unite the various individual solutions above, and specific enough that it can address vulnerabilities not addressed by current solutions, and guide the development of future mechanisms.

Overall, such a framework would provide web site administrators the tools for managing, in a least privilege [33] manner, the overall security characteristics of their web site/applications when realized in the context of user agents.

1.1. Where to Discuss This Draft

Please discuss this draft on the websec@ietf.org mailing list [WebSec].

2. Document Conventions

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

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

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

We will also be making use of the WebSec WG issue tracker, so use of the above two issue & TODO marks will evolve accordingly.

3. Overall Constraints

Regardless of the overall approaches chosen for conveying site security policies, we believe that to be deployed at Internet-scale, and to be as widely usable as possible for both novice and expert alike, the overall solution approach will need to address these three points of tension:

Granularity:

There has been much debate during the discussion of some policy mechanisms (e.g. CSP) as to how fine-grained such mechanisms should be. The argument against fine-grained mechanisms is that site administrators will cause themselves pain by instantiating policies that do not yield the intended results. E.g. simply copying the expressed policies of a similar site. The claim is that this would occur for various reasons stemming from the mechanisms' complexity [34].

Configurability:

Not infrequently, the complexity of underlying facilities, e.g. in server software, is not well-packaged and thus administrators are obliged to learn more about the intricacies of these systems than otherwise might be necessary. This is sometimes used as an argument for "dumbing down" the capabilities of policy expression mechanisms [34].

Usability:

Research shows that when security warnings are displayed, users are often given too much information as well as being allowed to relatively easily bypass the warnings and continue with their potentially compromising activity [35] [36] [37] [38] [39]. Thus users have become trained to "click through" security notifications "in order to get work done", though not infrequently rendering themselves insecure and perhaps compromised [40].

In the next section we discuss various high-level requirements derived with the guidance of the latter tension points.

4. Overall Requirements

1. Policy conveyance:

in-band:

We believe that a regime based on HTTP header(s) is appropriate. However we must devise a generalized, extensible HTTP security header(s) such that the on-going "bloat" of the number of disjoint HTTP security headers is mitigated and there is a documented framework that we can leverage as new approaches and/or threats emerge.

Note: The distinction between in-band and out-of-band signaling is difficult to characterize because some seemingly out-of-band mechanisms rely on the same protocols (HTTP/HTTPS) and infrastructure (transparent proxy servers) as the protocols they ostensibly protect.

It may be reasonable to devise a small set of headers to convey different classes of policies, e.g. web application content policies versus web application network capabilities policies.

out-of-band:

This policy communication mechanism must be secure and should have two facets, one for communicating securely out-of-band of the HTTP protocol to allow for secure client policy store bootstrapping. potential approaches are factory-installed web browser configuration, site security policy download a la Flash's crossdomain.xml and Maone's ABE for Web Authors [21], and DNS-based policy advertisement leveraging the security of DNS Security (DNSSEC) [32].

2. Granularity:

In terms of granularity, vast arrays of stand-alone blog, wiki, hosted web account, and other "simple" web sites could ostensibly benefit from relatively simple, pre-determined policies. However, complex sites--e.g. payment, ecommerce, software-as-a-service, mashup sites, etc.--often differ in various ways, as well as being inherently complex implementation-wise. One-size-fits-all policies will generally not work well for them. Thus, we believe that to be effective for a broad array of web site and application types,

the policy expression mechanism must fundamentally facilitate fine-grained control. For example, CSP offers such control. In order to address the less complex needs of the more simple classes of web sites, the policy expression mechanism could have a "macro"-like feature enabling "canned policy profiles". Or, the configuration facilities of various components of the web infrastructure can be enhanced to provide an appropriately simple veneer over the complexity.

3. Configurability:

With respect to configurability, development effort should be applied to creating easy-to-use administrative interfaces addressing the simple cases, like those mentioned above, while providing advanced administrators the tools to craft and manage fine-grained multi-faceted policies. Thus more casual or novice administrators can be aided in readily choosing, or be provided with, safe default policies while other classes of sites have the tools to craft the detailed policies they require. Examples of such an approach are Microsoft's "Packaging Wizard" [41] that easily auto-generates a quite complicated service deployment descriptor on behalf of less experienced administrators, and Firefox's simple Preferences dialog [42] as compared to its detailed about:config configuration editor page [43]. In both cases, simple usage by inexperienced users is anticipated and provided for on one hand, while complex tuning of the myriad underlying preferences is provided for on the other.

4. Usability:

Much has been learned over the last few years about what does and does not work with respect to security indicators in web browsers and web pages, as noted above, these lessons should be applied to the security indicators rendered by new proposed security mechanisms. We believe that in cases of user agents venturing into insecure situations, their response should be to fail the connections by default without user recourse, rather than displaying warnings along with bypass mechanisms, as is current practice. For example, the Strict Transport Security specification [I-D.draft-ietf-websec-strict-transport-sec-11] suggests the former hard-fail behavior.

5. Attacks and Threats to Address

This section enumerates various attacks and threats that ought to be mitigated by a web security policy framework. In terms of defining threats in contrast to attacks, Lucas supplied this:

```
<"Re: More on XSS mitigation (was Re: XSS mitigation in browsers)"
(Lucas Adamski).  http://lists.w3.org/Archives/Public/
public-web-security/2011Jan/0089.html>
```

```
"... There's a fundamental question about whether we should be
looking at these problems from an attack vs threat standpoint.  An
attack is XSS [or CSRF, or Response Splitting, etc.].  A threat is
that an attacker could compromise a site via content injection to
trick the user to disclosing confidential information (by
injecting a plugin or CSS to steal data or fool the user into
sending their password to the attacker's site).  ..."
```

5.1. Attacks

The below is an enumeration of attacks which are desirable to mitigate via a web application security framework (see [WASC-THREAT] for a definition and taxonomy of attacks):

1. cross-site-scripting (XSS) [2] [WASC-THREAT]
2. Man-in-the-middle (MITM) attacks against "secure" (e.g. Transport Layer Security (TLS/SSL)-based [7] [8] [WASC-THREAT]) web sites. For example, be able to subsume the HSTS header [11].
3. User Interface Redressing [UIRedress], aka Clickjacking [Clickjacking].
4. Cross-Site-Request Forgery (CSRF) [3] [WASC-THREAT] (?)
5. Response Splitting [WASC-THREAT]
6. more (ie eg from [WASC-THREAT] ?) ?

5.2. Threats

Via the attacks above, an attacker can..

1. Obtain a victim's confidential web application credentials (e.g., via cookie theft), and use the credentials to impersonate the victim and enter into transactions, often with the aim of monetizing the transaction results to the attacker's benefit.

2. Insert themselves as a Man-in-the-Middle (MITM) between victim and various services, thus is able to instigate, control, intercept, and attempt to monetize various transactions and interactions with web applications, to the benefit of the attacker.
3. Enumerate various user agent information stores, e.g. browser history, facilitating views of the otherwise confidential habits of the victim. This information could possibly be used in various offline attacks against the victim directly. E.g.: blackmail, denial of services, law enforcement actions, etc.
4. Use gathered information and credentials to construct and present a falsified persona of the victim (e.g. for character assassination).

There is a risk of exfiltration of otherwise confidential victim information with all the threats listed above.

6. Use Cases

This section outlines various concrete use cases. Where applicable, source email messages are cited.

1. I'm a web application site administrator. My web app includes static user-supplied content (e.g. submitted from user agents via HTML FORM + HTTP POST), but either my developers don't properly sanitize user-supplied content in all cases or/and content injection vulnerabilities exist or materialize (for various reasons).

This leaves my web app vulnerable to cross-site scripting. I wish I could set overall web app-wide policies that prevent user-supplied content from injecting malicious content (e.g. JavaScript) into my web app.

2. I'm a web application site administrator. My web application is intended, and configured, to be uniformly served over HTTPS, but my developers mistakenly keep including content via insecure channels (e.g. via insecure HTTP; resulting in so-called "mixed content").

I wish I could set a policy for my web app that prevents user agents from loading content insecurely even if my web app is otherwise telling them to do so.

3. I'm a web application site administrator. My site has a policy that we can only include content from certain trusted providers (e.g., our CDN, Amazon S3), but my developers keep adding dependencies on origins I don't trust. I wish I could set a policy for my site that prevents my web app from accidentally loading resources outside my whitelist.
4. I'm a web application site administrator. I want to ensure that my web app is never framed by other web apps.
5. I'm a developer of a web application which will be included (i.e. framed) by third parties within their own web apps. I would like to ensure that my web app directs user agents to only load resources from URIs I expect it to (possibly even down to specific URI paths), without affecting the containing web app or any other web apps it also includes.
6. I'm a web application site administrator. My web app frames other web apps whose behavior, properties, and policies are not 100% known or predictable.

I need to be able to apply policies that both protect my web app from potential vulnerabilities or attacks introduced by the framed web apps, and that work to ensure that the desired interactions between my web app and the framed apps are securely realized.

7. Detailed Functional Requirements

Many of the below functional requirements are extracted from a recent discussion on the [public-web-security] list. Particular messages are cited inline and appropriate quotes extracted and reproduced here. Inline citations are provided for definitions of various terms.

1. Policy expression syntax:

- * Declarative.

<"declarative languages". <http://www.encyclopedia.com/doc/1011-declarativelanguages.html>>

- * Extensible.

<"Extensibility". <https://secure.wikimedia.org/wikipedia/en/wiki/Extensible>>

<"Re: XSS mitigation in browsers" (Lucas Adamski). <http://lists.w3.org/Archives/Public/public-web-security/2011Jan/0066.html>>

"On a conceptual level, I am not really a believer in the current proliferation of orthogonal atomic mechanisms intended to solve very specific problems. Security is a holistic discipline, and so I'm a big supporter of investing in an extensible declarative security policy mechanism that could evolve as the web and the threats that it faces do. Web developers have a hard enough time with security already without being expected to master a potentially large number of different security mechanisms, each with their own unique threat model, implementation and syntax. Not to mention trying to figure out how they're expected to interact with each other... how to manage the gaps and intersections between the models."

<"Re: Scope and complexity (was Re: More on XSS mitigation)" (Adam Barth). <http://lists.w3.org/Archives/Public/public-web-security/2011Jan/0108.html>>

"I guess I wish we had an extensibility model more like HTML where we could grow the security protections over time. For example, we can probably agree that both <canvas> and <video> are great additions to HTML that might not have made sense when folks were designing HTML 1.0.

As long as you're not relying on the security policy as a first line of defense, the extensibility story for security policies is even better than it is with HTML tags. With an HTML tag, you need a fall-back for browsers that don't support the tag, whereas with a security policy, you'll always have your first line of defense.

Ideally, we could come up with a policy mechanism that let us nail XSS today and that fostered innovation in security for years to come. In the short term, you could view the existing CSP features (e.g., clickjacking protection) as the first wave of innovation. If those pieces are popular, then it should be easy for other folks to adopt them."

2. Tooling:

- * We will need tools to (ideally) analyze a web application and generate a starting point security policy.

<"Re: More on XSS mitigation" (John Wilander). <http://>

lists.w3.org/Archives/Public/public-web-security/2011Jan/0082.html>

"*Developers Will Want a Policy Generator* A key issue for in-the-field success of CSP is how to write, generate and maintain the policies. Just look at the epic failure of Java security policies. The Java policy framework was designed for static releases shipped on CDs, not for moving code, added frameworks, new framework versions etc. The world of web apps is so dynamic I'm still amazed. If anything, for instance messy security policies, gets in the way of daily releases it's a no go. At least until there's an exploit. Where am I going with this? Well, we should implement a PoC *policy generator* and run it on some fairly large websites before we nail the standard. There will be subtleties found which we can address and we can bring the PoC to production level while the standard is being finalized and shipped in browsers. Then we release the policy generator along with policy enforcement -- success! "

3. Performance:

* Minimizing performance impact is a first-order concern.

<"Re: More on XSS mitigation" (John Wilander). <http://lists.w3.org/Archives/Public/public-web-security/2011Jan/0082.html>>

"*We Mustn't Spoil Performance* Web developers (and browser developers) are so hung up on performance that we really need to look at what they're up to and make sure we don't spoil things. Especially load performance now that it's part of Google's rating."

4. Granularity:

* For example, discriminate between:

+ "inline" script in <head> versus <body>, or not.

+ "inline" script and "src=" loaded script.

+ Classes of "content", e.g. scriptable content, passive multimedia, nested documents, etc.

<"Proposal to move the debate forward" (Daniel Veditz). <http://lists.w3.org/Archives/Public/public-web-security/2011Jan/0122.html>>

"We oscillated several times between lumpy and granular. Fewer classes (simpler) is always more attractive, easier to explain and understand. The danger is that future features then end up being added to the existing lumps, possibly enabling things that the site isn't aware they need to now filter. It's a constant problem as we expand the capabilities of browsers -- sites that used to be perfectly secure are suddenly hackable because all the new browsers added feature-X."

5. Notifications and reporting:

- * Convey to the user agent an identifier (e.g. a URI) denoting where to send policy violation reports. Could also specify a DOM event to be dedicated for this purpose.
- * An ability to specify that a origin's policies are to be enforced in a "report only" mode will be useful for debugging policies as well as site-policy interactions. E.g. for answering the question: "does my policy 'break' my site?".

<"[Content Security Policy] Proposal to move the debate forward" (Brandon Sterne). <http://lists.w3.org/Archives/Public/public-web-security/2011Jan/0118.html>>

"...

3. Violation Reporting

- a. report-uri: URI to which a report will be sent upon policy violation
- b. SecurityViolation event: DOM event fired upon policy violations

..."

6. Facilitating Separation of Duties:

- * Specifically, allowing for Web Site operations/deployment personnel to apply site policy, rather than having it being encoded in the site implementation code by site developers/ implementors.

<"RE: Content Security Policy and iframe@sandbox" (Andrew Steingruebl). <http://lists.w3.org/Archives/Public/public-web-security/2011Feb/0050.html>>

"... 2. SiteC is also totally in control of all HTTP headers it emits. It could just as easily indicate policy choices for all frames via CSP. It could advertise a blanket policy

(No JS, No ActiveX). Advertising a page-specific, or frame/target specific policy is substantially more difficult and probably unwieldy. But, depending on how SiteC is configured, setting a global site policy via headers offers a potential separation of duties that #1 does not, it allows website admin to specific things that each web developer might not be able to. ..."

7. Hierarchical Policy Application:

- * The notion that policy emitted by the application's source origin is able to constrain behavior and policies of contained origins.

<"RE: Content Security Policy and iframe@sandbox" (Andrew Steingruebl). <http://lists.w3.org/Archives/Public/public-web-security/2011Feb/0048.html>>

"... I could imagine a tweak to CSP wherein CSP would control all contents hierarchically. I already spoke to Brandon about it, but it was just a quick brainstorm.

You could imagine revoking permissions in the frame hierarchy and not granting them back. This does start to get awfully ugly, but just as CSP controls loading policy for itself, it could also control loading policy for children, ..."

8. Framing Policy Hierarchy, cross-origin, granularity:

<"Re: Content Security Policy and iframe@sandbox") (Andy Steingruebl, Adam Barth) <http://lists.w3.org/Archives/Public/public-web-security/2011Feb/0051.html>>

On Sat, Feb 12, 2011 at 9:01 PM, Steingruebl, Andy
<asteingruebl@paypal-inc.com> wrote:

```
>> -----Original Message-----
>> From: Adam Barth [mailto:w3c@adambarth.com]
>
>> That all sounds very abstract. If you have some concrete examples,
>> that might be more productive to discuss. When enforcing policy
>> supplied by one origin on another origin, we need to be careful to
>> consider the case where the policy providing origin is the attacker
>> and the origin on which the policy is being enforced is the victim.
>
> SiteA wants to make sure it cannot ever be framed. It deploys
X-Frame-Options headers and framebusting JS, and maybe even CSP
```

frame-ancestors.

>

> SiteB wants to make sure it never loads data from anything other than SiteB (no non-origin loads). It outputs CSP headers to this effect

>

> SiteC wants to make sure that any content it frames cannot run ActiveX controls, nor do a 401 authentication. It can't really do this with current iframe sandboxing, but pretend it could...

>

> SiteC wants to control the behavior of children that it frames. It needs to advertise this policy to a web browser. It has two choices:

>

> 1. It can do it inline in the HTML it outputs with extra attributes of the iframe it creates. SiteC is in complete control of the HTML that creates the iframe. I can impose any policy via sandbox attributes. Currently for example, it can disable JS in the frame. If it frames SiteA, SiteA's framebusting JS will never run, but the browser will respect its X-Frame-Options headers.

>

> 2. SiteC is also totally in control of all HTTP headers it emits. It could just as easily indicate policy choices for all frames via CSP. It could advertise a blanket policy (No JS, No ActiveX). Advertising a page-specific, or frame/target specific policy is substantially more difficult and probably unwieldy. But, depending on how SiteC is configured, setting a global site policy via headers offers a potential separation of duties that #1 does not, it allows website admin to specific things that each web developer might not be able to.

>

> 3. Because all of Site A,B,C are in different origins, they don't really have to worry about polluting other origins, but they do have to worry about problematic behavior such as top-nav, 401-auth popups, etc. Parents need to constrain certain behavior of things they embed, according to certain rules of whether the child allows itself to be framed.

>

> I totally get how existing iframe sandboxing that turns off JS is problematic for sites [due to] older browsers that don't support X-Frame-Options. We already have a complicated interaction between these multiple security controls.

>

> Can you give me an example of why my #1/#2 are actually that different? Whether we control behavior with headers or inline content, each site is totally responsible for what it emits, and can already control in some interesting ways the behavior of content it frames/includes.

In this example, the trade-off for Site C seems to boil down to the granularity of the policy. Using attributes on a frame is more

fine-grained because Site C can make these decisions on an iframe-by-iframe basis whereas using a document-wide policy is more coarse-grained.

Of course, there's a trade-off between different granularities. On the one hand, fine-grained gives the site more control over how different iframes behavior. On the other hand, it's much easier to audit and understand the effects of a coarse-grained policy.

Adam

9. Policy Delivery:

- * The web application policy must be communicated by the web application to the user agent. There are various approaches and they have tradeoffs between security, audience, and practicality.

<"[Content Security Policy] Proposal to move the debate forward" (Brandon Sterne). <http://lists.w3.org/Archives/Public/public-web-security/2011Jan/0118.html>>

"...

6. Policy delivery

- a. HTTP header
- b. <meta> (or <link>) tag, to be superseded by header if present
- c. policy-uri: a URI from which the policy will be fetched; can be specified in either header or tag

..."

<"Re: [Content Security Policy] Proposal to move the debate forward" (gaz Heyes). <http://lists.w3.org/Archives/Public/public-web-security/2011Jan/0148.html>>

"...

- a) Policy shouldn't be defined in a http header it's too messy and what happens when there's a mistake?
- b) As discussed on the list there is no need to have a separate method as it can be generated by an attacker. If a policy doesn't exist then an attacker can now DOS the web site via meta.
- c) We have a winner, a http header specifying a link to the policy file is the way to go IMO, my only problem with it is

devs implementing it. Yes facebook would and probably twitter would but Dave's tea shop wouldn't pay enough money to hire a web dev who knew how to implement a custom http header yet they would know how to validate HTML. So the question is are we bothered about little sites that are likely to have nice tea and XSS holes? If so I suggest updating the HTML W3C validator to require a security policy to pass validation if not I suggest a policy file delivered by http header.
..."

10. Policy Conflict Resolution:

*

<"RE: Content Security Policy and iframe@sandbox" (Andrew Steingruebl). <http://lists.w3.org/Archives/Public/public-web-security/2011Feb/0048.html>>

> -----Original Message-----
> From: public-web-security-request@w3.org [mailto:public-web-security-
> request@w3.org] On Behalf Of Adam Barth
>
> @sandbox and CSP are very different. The primary difference is who
> choses the policy. In the case of @sandbox, the embedder chooses
> the policy. In CSP, the provider of the resource chooses the policy.

While this is true today, I could imagine a tweak to CSP wherein CSP would control all contents hierarchically. I already spoke to Brandon about it, but it was just a quick brainstorm.

You could imagine revoking permissions in the frame hierarchy and not granting them back. This does start to get awfully ugly, but just as CSP controls loading policy for itself, it could also control loading policy for children, right?

Fundamentally, since the existing security model doesn't really provide for strict separation of parent/child (popups, 401's, top-nav) CSP and iframe sandbox both try to control the behavior of resources we pull from other parties.

Do we think that these are both special cases of a general security policy (my intuition says yes) or that they have some quite orthogonal types of security controls that cannot be mixed into a single policy declaration?

One clear problem that comes to mind is that there are policies that come from the "child" such as X-Frame-Options that must break the ordinary parent/child relationship from a precedence standpoint.

8. Extant Policies to Coalesce

Presently, this section lists a grab-bag of individually-expressed web app security policies which a more general substrate could ostensibly encompass (in order to, for example, reduce "header bloat" and bytes-on-the-wire issues), as well as reduce functional policy duplication/overlap.

CORS

XDomainRequest

toStaticHtml

innerSafeHtml
X-Frame-Options
CSP frame-ancestors
more?

9. Example Concrete Approaches

An overall, broad approach (from [0]):

As for an overall policy mechanism, we observe that leveraging a combination of CSP [16] and ABE [19], or their employment in tandem, as a starting point for a multi-vendor approach may be reasonable. For a near-term policy delivery mechanism, we advocate use of both HTTP headers and a policy file at a well-known location. Leveraging DNSSEC is attractive in the intermediate term, i.e. as it becomes more widely deployed.

10. Security Considerations

Security considerations go here.

11. References

[[TODO1: re-code refs into xml and place in proper refs section.
--JeffH]]

[0] J. Hodges, A. Steingruebl, "The Need for Coherent Web Security Policy Framework(s)", Web 2.0 Security & Privacy, Oakland CA, 20 May 2010. <http://w2spconf.com/2010/papers/pl1.pdf>

[1] Breach Security, "THE WEB HACKING INCIDENTS DATABASE 2009," Aug. 2009. http://www.breach.com/resources/whitepapers/downloads/WP_TheWebHackingIncidents-2009.pdf

[2] R. Auger, The Cross-Site Request Forgery (CSRF/XSRF) FAQ, 2007. <http://www.cgisecurity.com/articles/csrf-faq.shtml>

[3] A. Barth, J. Caballero, and D. Song, "Secure Content Sniffing for Web Browsers--or How to Stop Papers from Reviewing Themselves," Proceedings of the 30th IEEE Symposium on Security & Privacy, Oakland, CA: 2009.

- [4] D. Goodin, "Major IE8 flaw makes 'safe' sites unsafe - Microsoft's XSS buster busted," The Register, Nov. 2009. http://www.theregister.co.uk/2009/11/20/internet_explorer_security_flaw/
- [5] J. Grossman, "Clickjacking: Web pages can see and hear you," Oct. 2008. <http://jeremiahgrossman.blogspot.com/2008/10/clickjacking-web-pages-can-see-and-hear.html>
- [6] W. Salusky, Malvertising, 2007. <http://isc.sans.org/diary.html?storyid=3727>
- [7] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC5246, Internet Engineering Task Force, Aug. 2008. <http://www.ietf.org/rfc/rfc5246.txt>
- [8] M. Marlinspike, SSLSTRIP, 2009. <http://www.thoughtcrime.org/software/sslstrip/>
- [9] Scope of HTTPOnly Cookies. http://docs.google.com/View?docid=dxxxqgkd_0cvcqhsdw
- [10] E. Lawrence, IE8 Security Part VII: ClickJacking Defenses, 2009. <http://blogs.msdn.com/ie/archive/2009/01/27/ie8-security-part-vii-clickjacking-defenses.aspx>
- [11] J. Hodges, C. Jackson, and A. Barth, "Strict Transport Security," Work-in-progress, Internet-Draft, Jul. 2010. <http://tools.ietf.org/html/draft-hodges-strict-transport-sec>
- [12] A. Barth, C. Jackson, and I. Hickson, "The Web Origin Concept," Internet-Draft, work in progress, Internet Engineering Task Force, 2009. <http://tools.ietf.org/html/draft-abarth-origin>
- [13] E. Lawrence, IE8 Security Part VI: Beta 2 Update, 2008. <http://blogs.msdn.com/ie/archive/2008/09/02/ie8-security-part-vi-beta-2-update.aspx>
- [14] G. Markham, Content restrictions, 2007. <http://www.gerv.net/security/content-restrictions/>
- [15] T. Jim, N. Swamy, and M. Hicks, "BEEP: Browser-Enforced Embedded Policies," Proceedings of the 16th International World Wide Web Conference, Banff, Alberta, Canada, 2007.
- [16] B. Sterne, "Content Security Policy (CSP)," 2011. <https://dvcs.w3.org/hg/content-security-policy/raw-file/bcflc45f312f/csp-unofficial-draft-20110303.html>

- [17] A.V. Kesteren, "Cross-Origin Resource Sharing (CORS)," Mar. 2009. <http://www.w3.org/TR/2009/WD-cors-20090317/>
- [18] Adobe Systems, "Cross-domain policy file specification." http://learn.adobe.com/wiki/download/attachments/64389123/CrossDomain_PolicyFile_Specification.pdf?version=1
- [19] G. Maone, ABE - Application Boundaries Enforcer, 2009. <http://noscript.net/abe/>
- [20] G. Maone, NoScript. <http://noscript.net/>
- [21] G. Maone, ABE for Web Authors, 2009. <http://noscript.net/abe/web-authors.html>
- [22] Microsoft, "Event 1046 - Cross-Site Scripting Filter," MSDN Library, undated. <http://msdn.microsoft.com/en-us/library/dd565647%28VS.85%29.aspx>
- [23] A. Barth, C. Jackson, and W. Li, "Attacks on JavaScript Mashup Communication," Proceedings of the Web 2.0 Security and Privacy Workshop, 2009.
- [24] M. Ter Louw, P. Bisht, and V. Venkatakrisnan, "Analysis of Hypertext Isolation Techniques for XSS Prevention," Proceedings of the Web 2.0 Security and Privacy Workshop, 2008 .
- [25] A. Ozment, S.E. Schechter, and R. Dhamija, "Web Sites Should Not Need to Rely on Users to Secure Communications," W3C Workshop on Transparency and Usability of Web Authentication, 2006.
- [26] C. Reis, A. Barth, and C. Pizano, "Browser Security: Lessons from Google Chrome," ACM Queue, 2009, pp. 1-8.
- [27] H.J. Wang, C. Grier, A. Moshchuk, S.T. King, P. Choudhury, and H. Venter, "The Multi-Principal OS Construction of the Gazelle Web Browser," USENIX Security Symposium, 2009.
- [28] M. Zalewski, Browser Security Handbook. <http://code.google.com/p/browsersec/>
- [29] A. Stamos, D. Thiel, and J. Osborne, Living in the RIA World: Blurring the Line between Web and Desktop Security, BlackHat presentation, iSecPartners, 2008. https://www.isecpartners.com/files/RIA_World_BH_2008.pdf
- [30] Mary Shelley, "Frankenstein, or The Modern Prometheus," ca. 1831. http://en.wikipedia.org/wiki/Frankenstein%27s_monster

- [31] D. Goodin, "cPanel, Netgear and Linksys susceptible to nasty attack - Unholy Trinity," The Register, 2009.
http://www.theregister.co.uk/2009/08/02/unholy_trinity_csrf/
- [32] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," RFC4033, Internet Engineering Task Force, Mar. 2005.
<http://www.ietf.org/rfc/rfc4033.txt>
- [33] J.H. Saltzer and M.D. Schroeder, "The Protection of Information in Computer Systems," Communications of the ACM, vol. 17, Jul. 1974.
- [34] I. Hickson and many others, "Comments on the Content Security Policy specification," discussion on mozilla.dev.security newsgroup.
http://groups.google.com/group/mozilla.dev.security/browse_frm/thread/87ebe5cb9735d8ca?tvc=1&q=Comments+on+the+Content+Security+Policy+specification
- [35] S. Egelman, L.F. Cranor, and J. Hong, "You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings," CHI 2008, April 5 - 10, 2008, Florence, Italy, 2008.
- [36] S.E. Schechter, R. Dhamija, A. Ozment, and I. Fischer, "The Emperor's New Security Indicators," Proceedings of the 2007 IEEE Symposium on Security and Privacy.
- [37] R. Dhamija and J.D. Tygar, "The Battle Against Phishing: Dynamic Security Skins," Proceedings of the 2005 Symposium on Usable Privacy and Security (SOUPS).
- [38] J. Sobey, T. Whalen, R. Biddle, P.V. Oorschot, and A.S. Patrick, Browser Interfaces and Extended Validation SSL Certificates: An Empirical Study, Ottawa, Canada: School of Computer Science, Carleton University, 2009.
- [39] J. Sunshine, S. Egelman, H. Almuhiemedi, N. Atri, and L.F. Cranor, "Crying Wolf: An Empirical Study of SSL Warning Effectiveness," USENIX Security Symposium, 2009.
- [40] C. Jackson and A. Barth, "ForceHTTPS: Protecting High-Security Web Sites from Network Attacks," Proceedings of the 17th International World Wide Web Conference (WWW), 2008.
- [41] Microsoft, "Packaging Wizard."
[http://msdn.microsoft.com/en-us/library/aal57732\(office.10\).aspx](http://msdn.microsoft.com/en-us/library/aal57732(office.10).aspx)
- [42] Mozilla, "Options window."

<http://support.mozilla.com/en-US/kb/Options+window>

[43] S. Yegulalp, "Hacking Firefox: The secrets of about:config," ComputerWorld, May. 2007. http://www.computerworld.com/s/article/9020880/Hacking_Firefox_The_secrets_of_about_config

12. Informative References

[Clickjacking]

"Clickjacking", Sep 2008,
<<http://www.sectheory.com/clickjacking.htm>>.

[I-D.ietf-websec-strict-transport-sec]

Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)",
draft-ietf-websec-strict-transport-sec-11 (work in progress), July 2012.

[UIRedress]

"Dealing with UI redress vulnerabilities inherent to the current web", Sep 2008, <<http://lists.whatwg.org/htdig.cgi/whatwg-whatwg.org/2008-September/016284.html>>.

[WASC-THREAT]

Web Application Security Consortium, "The WASC Threat Classification v2.0", January 2010,
<http://projects.webappsec.org/f/WASC-TC-v2_0.pdf>.

[WebSec]

"Web HTTP Application Security Minus Authentication and Transport",
<<https://www.ietf.org/mailman/listinfo/websec>>.

[public-web-security]

"public-web-security@w3.org: Improving standards and implementations to advance the security of the Web.",
<<http://lists.w3.org/Archives/Public/public-web-security/>>.

Author's Address

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

Email: Jeff.Hodges@PayPal.com

WEBSEC
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2013

D. Ross
Microsoft
T. Gondrom
July 6, 2012

HTTP Header Frame Options
draft-ietf-websec-frame-options-00

Abstract

To improve the protection of web applications against Clickjacking this standards defines a http response header that declares a policy communicated from a host to the client browser whether the transmitted content MUST NOT be displayed in frames of other pages from different origins which are allowed to frame the content.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 7, 2013.

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	3
1.1. Requirements Language	3
2. Frame-Options Header	3
2.1. Syntax	3
2.2. Backus-Naur Form (BNF)	4
2.3. Design Issues	5
2.3.1. Enable HTML content from other domains	5
2.3.2. Browser Behaviour and Processing	5
2.4. Examples of Frame-Options Headers	6
2.4.1. Example scenario for the ALLOW-FROM parameter	6
3. Acknowledgements	6
4. IANA Considerations	6
4.1. Registration Template	7
5. Security Considerations	7
6. References	7
6.1. Normative References	7
6.2. Informative References	7
Appendix A. Description of a Clickjacking attack	8
A.1. Shop	8
A.2. Confirm Purchase Page	9
A.3. Flash Configuration	9
Authors' Addresses	9

1. Introduction

In 2009 and 2010 many browser vendors introduced the use of a non-standard http header RFC 2616 [RFC2616] "X-Frame-Options" to protect against Clickjacking [Clickjacking]. This standard is to replace the non-standard header.

Existing anti-ClickJacking measures, e.g. Frame-breaking Javascript, have weaknesses so that their protection can be circumvented as a study [FRAME-BUSTING] demonstrated.

Short of configuring the browser to disable frames and script entirely, which massively impairs browser utility, browser users are vulnerable to this type of attack.

"Frame-Options" allows a secure web page from host B to declare that its content (for example a button, links, text, etc.) must not be displayed in a frame of another page (e.g. from host A). In principle this is done by a policy declared in the HTTP header and obeyed by conform browser implementations.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

2. Frame-Options Header

The Frame-Options HTTP response header indicates a policy whether a browser MUST NOT allow to render a page in a <frame> or <iframe> . Hosts can declare this policy in the header of their HTTP responses to prevent clickjacking attacks, by ensuring that their content is not embedded into other pages or frames.

2.1. Syntax

The header field name is:
Frame-Options

There are three different values for the header field. These values are exclusive, that is NOT more than one of the three values MUST be set.

DENY

A browser receiving content with this header MUST NOT display this content in any frame.

SAMEORIGIN

A browser receiving content with this header MUST NOT display this content in any frame from a page of different origin than the content itself.

If a browser or plugin can not reliably determine whether the origin of the content and the frame have the same origin, this MUST be treated as "DENY".

[TBD]current implementations do not display if the origin of the top-level-browsing-context is different than the origin of the page containing the FRAME-OPTIONS header.

ALLOW-FROM (followed by a URI of a trusted origin)

A browser receiving content with this header MUST NOT display this content in any frame from a page of different origin than the listed origin. While this can expose the page to risks by the trusted origin, in some cases it may be necessary to use content from other domains.

For example: FRAME-OPTIONS: ALLOW-FROM https://www.domain.com/

In the case of SAMEORIGIN and ALLOW-FROM, there is also an optional flag "AllAncestors". If this flag is set, it means that browsers MUST validate the URL of each hosting frame up to the top level and only allow the framing if all ancestor frames' origins are either the same as in SAMEORIGIN or included in the ALLOW-FROM list.

The URIs listed for ALLOW-FROM must be valid.

Any data beyond the domain address (i.e. any data after the "/" separator) is to be ignored and to verify a referring page is of the same origin as the content or that the referring page is listed in the ALLOW-FROM list of URI, the algorithm to compare origins from [ORIGIN] should be used.

Wildcards to declare multiple domains in one statement are not permitted.

[TBD] Current Implementations do not consider the port a component of the origin - conflicting with [ORIGIN].

2.2. Backus-Naur Form (BNF)

The RFC 822 [RFC0822] EBNF of the Frame-Options header is:

```
Frame-Options = "Frame-Options" ":" "DENY"/ "SAMEORIGIN" /  
                ("ALLOW-FROM" ":" URI) : flags
```

```
flags      = token [ "=" ( token | quoted-string ) ]
```

[TBD] with URI as defined in the websec-origin draft
[TBD] Or should we use the ABNF (RFC 2234) alternatively or in addition?

2.3. Design Issues

2.3.1. Enable HTML content from other domains

There are three main direct vectors that enable HTML content from other domains:

- o IFRAME Tag
- o Frame tag
- o The Object tag (requires a redirect)

Besides these other ways to host HTML content can be possible. For example some plugins may host HTML views directly. If these plugins appear essentially as frames (as opposed to top-level windows), the plugins MUST conform to the FRAME-OPTIONS directive as specified in this draft as well.

2.3.2. Browser Behaviour and Processing

To allow secure implementations browser implementations MUST behave in a consistent and reliable way.

If a HTTP Header prohibits framing, the user-agent of the browser MAY immediately abort downloading or parsing of the document.

When a browser discovers loaded content with the FRAME-OPTIONS header would be displayed in a frame against the specified origin orders of the header, the browser SHOULD redirect as soon as possible to a "No-Frame" page.

"No-Frame" Page

If the display of content is denied by the FRAME-OPTIONS header an error page SHOULD be displayed. For example this can be a noframe.html page also stating the full URL of the protected page and the hostname of the protected page.

The NoFrame page MAY provide the user with an option to open the target URL in a new window.

2.4. Examples of Frame-Options Headers

2.4.1. Example scenario for the ALLOW-FROM parameter

1. Inner IFRAME suggests via a querystring parameter what site it wants to be hosted by. This can obviously be specified by an attacker, but that's OK.
2. Server verifies the hostname meets whatever criteria. For example, for a Facebook "Like" button, the server can check to see that the supplied hostname matches the hostname expected for that Like button.
3. Server serves up the hostname in FRAME-OPTIONS: ALLOW-FROM if the proper criteria was met in step #2.
4. Browser enforces the FRAME-OPTIONS: ALLOW-FROM domain.com header.

3. Acknowledgements

This document was derived from input from specifications published by various browser vendors like Microsoft (Eric Lawrence, David Ross), Mozilla, Google, Opera and Apple.

4. IANA Considerations

This memo a request to IANA to include the specified HTTP header in registry as outlined in Registration Procedures for Message Header Fields [RFC3864]

4.1. Registration Template

PERMANENT MESSAGE HEADER FIELD REGISTRATION TEMPLATE:

Header field name: Frame-Option

Applicable protocol: http [RFC2616]

Status: Standard

Author/Change controller: IETF

Specification document(s): draft-ietf-websec-frame-options

Related information:

Figure 1

5. Security Considerations

The introduction of the http header FRAME-OPTIONS does improve the protection against Clickjacking, however it is not self-sufficient on its own but MUST be used in conjunction with other security measures like secure coding and Content Security Policy (CSP)

The parameter ALLOW-FROM allows a page possibilities to guess who is framing it. This is by design, but may lead to data leakage or data protection concerns.

6. References

6.1. Normative References

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

6.2. Informative References

[CLICK-DEFENSE-BLOG]
Microsoft, "Clickjacking Defense", 2009, <<http://blogs.msdn.com/b/ie/archive/2009/01/27/ie8-security-part-vii-clickjacking-defenses.aspx>>.

[CSRF] OWASP (Open Web Application Security Project), "OWASP Top-10: Cross-Site Request Forgery (CSRF)", 2010,

<http://www.owasp.org/index.php/Top_10_2010-A5>.

[Clickjacking]

OWASP (Open Web Application Security Project),
"Clickjacking", 2010,
<<http://www.owasp.org/index.php/Clickjacking>>.

[FRAME-BUSTING]

Stanford Web Security Research, "Busting frame busting: a study of clickjacking vulnerabilities at popular sites", 2010, <<http://seclab.stanford.edu/websec/framebusting/>>.

[ORIGIN]

IETF, "The Web Origin Concept", December 2010,
<<http://tools.ietf.org/id/draft-ietf-websec-origin-00.txt>>.

[RFC0822]

Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[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.

[RFC3864]

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

[RFC6454]

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

Appendix A. Description of a Clickjacking attack

More detailed explanation of Clickjacking scenarios

A.1. Shop

An Internet Marketplace/Shop offering a feature with a link/button to "Buy this" Gadget

The marketplace wants their affiliates (who could be bad guys) to be able to stick the "Buy such-and-such from XYZ" IFRAMES into their pages. There is a ClickJack possibility here, which is why the marketplace/onlineshop needs to then immediately navigate the main browsing context (or a new window) to a confirmation page which is protected by anti-CJ protections.

A.2. Confirm Purchase Page

Onlineshop "Confirm purchase" anti-CSRF page

The Confirm Purchase page must be shown to the end user without possibility of overlay or misuse by an attacker. For that reason, the confirmation page uses anti-CSRF tokens and contains the FRAME-OPTIONS directive, mitigating ClickJack attacks.

A.3. Flash Configuration

Macromedia Flash configuration page

Macromedia Flash configuration settings are set by a Flash object which can run only from a specific configuration page on Macromedia's site. The object runs inside the page and thus can be subject to a ClickJacking attack. In order to prevent ClickJacking attacks against the security settings, the configuration page uses the FRAME-OPTIONS directive.

Authors' Addresses

David Ross
Microsoft
U.S.

Phone:
Email:

Tobias Gondrom
Kruegerstr. 5A
Unterschleissheim,
Germany

Phone: +44 7521003005
Email: tobias.gondrom@gondrom.org

Web Security
Internet-Draft
Intended status: Standards Track
Expires: December 6, 2012

C. Evans
C. Palmer
Google, Inc.
June 4, 2012

Public Key Pinning Extension for HTTP
draft-ietf-websec-key-pinning-02

Abstract

This memo describes an extension to the HTTP protocol allowing web host operators to instruct user agents (UAs) to remember ("pin") the hosts' cryptographic identities for a given period of time. During that time, UAs will require that the host present a certificate chain including at least one Subject Public Key Info structure whose fingerprint matches one or more of the pinned fingerprints for that host. By effectively reducing the scope of authorities who can authenticate the domain during the lifetime of the pin, pinning may reduce the incidence of man-in-the-middle attacks due to compromised Certification Authorities and other authentication errors and attacks.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 6, 2012.

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.

1. Introduction

We propose a new HTTP header to enable a web host to express to user agents (UAs) which Subject Public Key Info (SPKI) structure(s) UAs MUST expect to be present in the host's certificate chain in future connections using TLS (see [rfc-5246]). We call this "public key pinning". At least one user agent (Google Chrome) has experimented with shipping with a user-extensible embedded set of pins. Although effective, this does not scale. This proposal addresses the scale problem.

Deploying public key pinning safely will require operational and organizational maturity due to the risk that hosts may make themselves unavailable by pinning to a SPKI that becomes invalid. (See Section 3.) We believe that, with care, host operators can greatly reduce the risk of MITM attacks and other false-authentication problems for their users without incurring undue risk.

We intend for hosts to use public key pinning together with HSTS (as defined in [hsts-draft]), but is possible to pin keys without requiring HSTS.

This draft is being discussed on the WebSec Working Group mailing list, websec@ietf.org.

1.1. Requirements Language

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 RFC 2119 [rfc-2119].

2. Server and Client Behavior

2.1. Response Header Field Syntax

To set a pin, hosts use a new HTTP header field, `Public-Key-Pins`, in their HTTP responses. Figure 1 describes the syntax of the header field.

```
Public-Key-Pins = "Public-Key-Pins" ":" LWS directives

directives      = max-age LWS ";" LWS pins
                  / pins LWS ";" LWS max-age

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

pins            = pin
                  / pin LWS ";" LWS pins

pin             = "pin-" token LWS "=" LWS quoted-string
```

Figure 1

In the pin rule, the token is the name of a cryptographic hash algorithm, and MUST be either "sha1" or "sha256". (Future versions of this specification may change the hash functions.) The quoted-string is a sequence of base64 digits: a base64-encoded hash. See Section 2.2.

Figure 2 shows some example response header fields using the pins extension (folded for clarity).

```
Public-Key-Pins: max-age=500;
  pin-sha1="4n972HfV354KP560yw4uqe/baXc=";
  pin-sha1="IvGeLsbqzPxdI0b0wuj2xVTdXgc="

Public-Key-Pins: max-age=31536000;
  pin-sha1="4n972HfV354KP560yw4uqe/baXc=";
  pin-sha256="LPJNul+wow4m6DsqxnbnnhsWHlwfp0JecwQzYpOLmCQ="

Public-Key-Pins: pin-sha1="4n972HfV354KP560yw4uqe/baXc=";
  pin-sha1="qvTGHdzF6KLavt4PO0gs2a6pQ00=";
  pin-sha256="LPJNul+wow4m6DsqxnbnnhsWHlwfp0JecwQzYpOLmCQ=";
  max-age=2592000
```

Figure 2

2.2. Semantics of Pins

The fingerprint is the SHA-1 or SHA-256 hash of the DER-encoded ASN.1 representation of the SubjectPublicKeyInfo (SPKI) field of the X.509 certificate. Figure 3 reproduces the definition of the SubjectPublicKeyInfo structure in [rfc-5280].

```
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm      AlgorithmIdentifier,
    subjectPublicKey BIT STRING }

AlgorithmIdentifier ::= SEQUENCE {
    algorithm      OBJECT IDENTIFIER,
    parameters    ANY DEFINED BY algorithm OPTIONAL }
```

Figure 3

The SPKI hash is then encoded in base-64 for use in an HTTP header. (See [rfc-4648].)

We pin public keys, rather than entire certificates, to enable operators to generate new certificates containing old public keys (see [why-pin-key]).

See Appendix A for an example non-normative program that generates public key fingerprints from SubjectPublicKeyInfo fields in certificates.

2.3. Noting Pins

Upon receipt of the Public-Key-Pins response header field, the UA notes the host as a Pinned Host, storing the pins and their associated max-age in non-volatile storage (for example, along with the HSTS metadata). The pins and their associated max-age are collectively known as Pinning Metadata.

The UA MUST observe these conditions when noting a host:

- o The UA MUST note the pins if and only if it received the Public-Key-Pins response header field over an error-free TLS connection. The UAs MUST ignore Public-Key-Pins response header fields received on HTTP (non-HTTPS) connections.
- o The UA MUST note the pins if and only if the TLS connection was authenticated with a certificate chain containing at least one of the SPKI structures indicated by at least one of the given fingerprints. (See Section 2.4.)

- o The UA MUST note the pins if and only if the given set of pins contains at least one pin that does NOT refer to an SPKI in the certificate chain. (That is, the host must set a Backup Pin; see Section 3.1.)

If the Public-Key-Pins response header field does not meet all three of these criteria, the UA MUST NOT note the host as a Pinned Host, and MUST discard any previously set Pinning Metadata for that host in its non-volatile store. Public-Key-Pins response header fields that meet all these criteria are known as Valid Pinning Headers.

Whenever a UA receives a Valid Pinning Header, it MUST set its Pinning Metadata to the exact pins and max-age given in the most recently received Valid Pinning Header.

2.3.1. max-age

max-age specifies the number of seconds, after the reception of the Public-Key-Pins HTTP Response Header, during which the UA regards the host as a Pinned Host (up to a maximum of 30 days; see Section 2.5). The delta-seconds production is specified in [rfc-2616].

Note that by setting a low or 0 value for max-age, hosts effectively instruct UAs to cease regarding them as Pinned Hosts.

2.4. Validating Pinned Connections

When a UA connects to a Pinned Host, if the TLS connection has errors, the UA MUST terminate the connection without allowing the user to proceed anyway. (This behavior is the same as that required by [hsts-draft].)

If the connection has no errors, the UA will then apply a new, additional correctness check: Pin Validation. To perform Pin Validation, the UA will compute the fingerprints of the SPKI structures in each certificate in the host's validated certificate chain. (For the purposes of Pin Validation, the UA MUST ignore superfluous certificates in the chain that do not form part of the validating chain.) The UA will then check that the set of these fingerprints intersects the set of fingerprints in that host's Pinning Metadata. If there is set intersection, the UA continues with the connection as normal. Otherwise, the UA MUST treat this Pin Failure as a non-recoverable error.

Note that, although the UA has previously received public key pins at the HTTP layer, it can and MUST perform Pin Validation at the TLS layer, before beginning an HTTP conversation over the TLS channel. The TLS layer thus evaluates TLS connections with pinning information

the UA received previously, regardless of mechanism: statically preloaded, via HTTP header, or some other means (possibly in the TLS layer itself, such as specified in [tack-draft]).

2.5. Pin Validity Times

In harmony with section 5.3.4 "Create and activate pins" of [tack-draft], clients MUST enforce a maximum age for pins that is no longer than the least of (a) 30 days (30 * 24 * 60 * 60 seconds) after the most recent time that the client noted the pin; (b) the amount of time the pin has been noted; or (c) the most recent time the pin was noted + max-age:

```
active_period_end = MIN(current + current - initial,
                        time_pin_noted + max-age,
                        current + 30 days)
```

Figure 4

2.6. Interactions With Preloaded Pin Lists

UAs MAY choose to implement built-in public key pins, alongside any built-in HSTS opt-in list. UAs MUST allow users to override a built-in pin list, including turning it off.

UAs MUST use the newest information -- built-in or set via Valid Pinning Header -- when performing Pin Validation for the host.

2.7. Pinning Self-Signed End Entities

If UAs accept hosts that authenticate themselves with self-signed end entity certificates, they MAY also allow hosts to pin the public keys in such certificates. The usability and security implications of this practice are outside the scope of this specification.

3. Security Considerations

Pinning public keys helps hosts assert their cryptographic identity, but there is some risk that a host operator could lose or lose control of their host's private key. In this case, the operator would not be able to serve their web site or application in a way that UAs would trust for the duration of their pin's max-age. (Recall that UAs MUST close the connection to a host upon Pin Failure.)

3.1. Backup Pins

The primary way to cope with the risk of inadvertant Pin Failure is to keep a Backup Pin. A Backup Pin is a fingerprint for the public key of a secondary, not-yet-deployed key pair. The operator keeps the backup key pair offline, and sets a pin for it in the Public-Key-Pins header. Then, in case the operator loses control of their primary private key, they can deploy the backup key pair. UAs, who have had the backup key pair pinned (when it was set in previous Valid Pinning Headers), can connect to the host without error.

Because having a backup key pair is so important to recovery, UAs MUST require that hosts set a Backup Pin. (See Section 2.3.)

4. IANA Considerations

This document has no actions for IANA.

5. Usability Considerations

When pinning works to detect impostor Pinned Hosts, users will experience denial of service. UAs MUST explain the reason why, i.e. that it was impossible to verify the confirmed cryptographic identity of the host.

UAs MUST have a way for users to clear current pins for Pinned Hosts. UAs SHOULD have a way for users to query the current state of Pinned Hosts.

6. Acknowledgements

Thanks to Tobias Gondrom, Jeff Hodges, Adam Langley, Nicolas Lidzborski, SM, James Manger, and Yoav Nir for suggestions and edits that clarified the text. Thanks to Trevor Perrin for suggesting a mechanism to affirmatively break pins ([pin-break-codes]). Adam Langley provided the SPKI fingerprint generation code.

7. What's Changed

Removed the section on pin break codes and verifiers, in favor the of most-recently-received policy (Section 2.3).

Now using a new header field, Public-Key-Pins, separate from HSTS. This allows hosts to use pinning separately from Strict Transport Security.

Explicitly requiring that UAs perform Pin Validation before the HTTP conversation begins.

Backup Pins are now required.

Separated normative from non-normative material. Removed tangential and out-of-scope non-normative discussion.

8. References

8.1. Normative References

[hsts-draft]

Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", October 2011, <<http://tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-03>>.

[rfc-2119]

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

[rfc-2616]

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

[rfc-4648]

Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", October 2006, <<http://www.ietf.org/rfc/rfc4648.txt>>.

[rfc-5246]

Rescorla, E. and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2", August 2008, <<http://www.ietf.org/rfc/rfc5246.txt>>.

[rfc-5280]

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", May 2008, <<http://www.ietf.org/rfc/rfc5280.txt>>.

8.2. Informative References

[why-pin-key]

Langley, A., "Public Key Pinning", May 2011, <<http://www.imperialviolet.org/2011/05/04/pinning.html>>.

[pin-break-codes]

Perrin, T., "Self-Asserted Key Pinning", September 2011, <<http://trevp.net/SAKP/>>.

[tack-draft]

Perrin, T. and M. Marlinspike, "Trust Assertions for
Certificate Keys", May 2012,
<<http://tools.ietf.org/html/draft-perrin-tls-tack>>.

Appendix A. Fingerprint Generation

This Go program generates public key fingerprints, suitable for use in pinning, from PEM-encoded certificates. It is non-normative.

```
package main

import (
    "io/ioutil"
    "os"
    "crypto/sha1"
    "crypto/x509"
    "encoding/base64"
    "encoding/pem"
    "fmt"
)

func main() {
    if len(os.Args) < 2 {
        fmt.Printf("Usage: %s PEM-filename\n", os.Args[0])
        os.Exit(1)
    }
    pemBytes, err := ioutil.ReadFile(os.Args[1])
    if err != nil {
        panic(err.String())
    }
    block, _ := pem.Decode(pemBytes)
    if block == nil {
        panic("No PEM structure found")
    }
    derBytes := block.Bytes
    certs, err := x509.ParseCertificates(derBytes)
    if err != nil {
        panic(err.String())
    }
    cert := certs[0]
    h := sha1.New()
    h.Write(cert.RawSubjectPublicKeyInfo)
    digest := h.Sum()

    fmt.Printf("Hex: %x\nBase64: %s\n", digest,
        base64.StdEncoding.EncodeToString(digest))
}
```

Figure 5

Appendix B. Deployment Guidance

This section is non-normative guidance which may smooth the adoption of public key pinning.

- o Operators SHOULD get the backup public key signed by a different (root and/or intermediary) CA than their primary certificate, and store the backup key pair safely offline.
- o It is most economical to have the backup certificate signed by a completely different signature chain than the live certificate, to maximize recoverability in the event of either root or intermediary signer compromise.
- o Operators SHOULD periodically exercise their Backup Pin plan -- an untested backup is no backup at all.
- o Operators SHOULD start small. Operators SHOULD first deploy public key pinning by setting a max-age of minutes or a few hours, and gradually increase max-age as they gain confidence in their operational capability.

Authors' Addresses

Chris Evans
Google, Inc.
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: cevans@google.com

Chris Palmer
Google, Inc.
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: palmer@google.com

WEBSEC Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2013

J. Hodges
PayPal
C. Jackson
Carnegie Mellon University
A. Barth
Google, Inc.
Jul 10, 2012

HTTP Strict Transport Security (HSTS)
draft-ietf-websec-strict-transport-sec-11

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 11, 2013.

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	4
1.1. Organization of this specification	5
1.2. Document Conventions	5
2. Overview	5
2.1. Use Cases	5
2.2. HTTP Strict Transport Security Policy Effects	6
2.3. Threat Model	6
2.3.1. Threats Addressed	6
2.3.1.1. Passive Network Attackers	6
2.3.1.2. Active Network Attackers	7
2.3.1.3. Web Site Development and Deployment Bugs	7
2.3.2. Threats Not Addressed	8
2.3.2.1. Phishing	8
2.3.2.2. Malware and Browser Vulnerabilities	8
2.4. Requirements	8
2.4.1. Overall Requirement	8
2.4.1.1. Detailed Core Requirements	8
2.4.1.2. Detailed Ancillary Requirements	9
3. Conformance Criteria	10
4. Terminology	10
5. HSTS Mechanism Overview	12
6. Syntax	13
6.1. Strict-Transport-Security HTTP Response Header Field	13
6.1.1. The max-age Directive	14
6.1.2. The includeSubDomains Directive	14
6.2. Examples	14
7. Server Processing Model	15
7.1. HTTP-over-Secure-Transport Request Type	15
7.2. HTTP Request Type	15
8. User Agent Processing Model	16
8.1. Strict-Transport-Security Response Header Field Processing	17
8.1.1. Noting an HSTS Host	17
8.2. Known HSTS Host Domain Name Matching	18
8.3. URI Loading and Port Mapping	19
8.4. Errors in Secure Transport Establishment	20
8.5. HTTP-Equiv <Meta> Element Attribute	20
8.6. Missing Strict-Transport-Security Response Header Field	20
9. Domain Name IDNA-Canonicalization	20

10. Server Implementation and Deployment Advice	21
10.1. HSTS Policy expiration time considerations	21
10.2. Using HSTS in conjunction with self-signed public-key certificates	22
10.3. Implications of includeSubDomains	23
11. User Agent Implementation Advice	24
11.1. No User Recourse	24
11.2. User-declared HSTS Policy	24
11.3. HSTS Pre-Loaded List	25
11.4. Disallow Mixed Security Context Loads	25
11.5. HSTS Policy Deletion	25
12. Constructing an Effective Request URI	26
12.1. ERU Fundamental Definitions	26
12.2. Determining the Effective Request URI	26
12.2.1. Effective Request URI Examples	27
13. Internationalized Domain Names for Applications (IDNA): Dependency and Migration	27
14. Security Considerations	28
14.1. Ramifications of HSTS Policy Establishment only over Error-free Secure Transport	28
14.2. The Need for includeSubDomains	29
14.3. Denial of Service	30
14.4. Bootstrap MITM Vulnerability	31
14.5. Network Time Attacks	31
14.6. Bogus Root CA Certificate Phish plus DNS Cache Poisoning Attack	32
14.7. Creative Manipulation of HSTS Policy Store	32
14.8. Internationalized Domain Names	32
15. IANA Considerations	33
16. References	34
16.1. Normative References	34
16.2. Informative References	35
Appendix A. Design Decision Notes	38
Appendix B. Differences between HSTS Policy and Same-Origin Policy	38
Appendix C. Acknowledgments	39
Appendix D. Change Log	40
D.1. For draft-ietf-websec-strict-transport-sec	40
D.2. For draft-hodges-strict-transport-sec	47
Authors' Addresses	48

1. Introduction

The HTTP protocol [RFC2616] may be used over various transports, typically the Transmission Control Protocol (TCP). However, TCP does not provide channel integrity protection, confidentiality, nor 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-a-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], and thus can be described as "click-through insecurity".

A key vulnerability enabled by click-through insecurity is the leaking of any cookies the web 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 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 below 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. The HSTS mechanism itself is formally specified in Section 4 through Section 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, and can contain normative requirements.

2. Overview

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

2.1. Use Cases

The high-level use case is a combination of:

- o Web browser user wishes to interact with various web sites (some arbitrary, some known) in a secure fashion.
- o Web site deployer wishes to offer their site in an explicitly secure fashion for both their own, as well as their users', benefit.

2.2. HTTP Strict Transport Security Policy Effects

The effects of the HTTP Strict Transport Security (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. Addressed and not addressed threats are briefly discussed below. Readers may wish 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], which enable their wielder to obtain other local users' session cookies for various web applications.

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

permit interoperability with versions of the service offered over insecure transport ("Secure 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 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

- 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 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.

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 11.1 "No User Recourse".

Note: A means for uniformly securely meeting the first core requirement above is not specifically addressed by this specification (see Section 14.4 "Bootstrap MITM Vulnerability"). It may be addressed by a future revision of this specification or some other specification. Note also that there are means by which UA implementations may more fully meet the first core requirement; see Section 11 "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 (UAs).

[[IESG Note: the next two paragraphs are for readers with a background in W3C specification style, of which we expect many. RFC Editor, please remove this note before publication.]]

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: 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 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 12 "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.

Local policy: comprises 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. See also "Effective Request URI".

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] .

unknown HSTS Host: is an HSTS Host that the user agent in question has not yet 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 Section 6 through Section 15.

An HSTS Host conveys its HSTS Policy to UAs via the Strict-Transport-Security HTTP response header field over secure transport (e.g., TLS). Receipt of this header field signals to UAs to enforce the HSTS Policy for all subsequent connections made to the HSTS Host, for a specified time duration. Application of the HSTS Policy to subdomains of the HSTS Host name may optionally be specified.

HSTS Hosts manage their advertised HSTS Policies by sending Strict-Transport-Security HTTP response header fields to UAs with new values for policy time duration and application to subdomains. UAs cache the "freshest" HSTS Policy information on behalf of an HSTS Host. Specifying a zero time duration signals to the UA to delete the HSTS Policy for that HSTS Host.

Section 6.2 presents examples of Strict-Transport-Security HTTP response header fields.

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 fields 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 defined for them at such time.

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". 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.

6.1.2. The includeSubDomains Directive

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

6.2. Examples

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

Strict-Transport-Security: max-age=31536000

The below HSTS header field stipulates that the HSTS Policy is to remain in effect for approximately six months and the policy applies 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"
```

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], 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, 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.

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.

Establishing a given host as a Known HSTS Host, in the context of a given UA, MAY be accomplished over the HTTP protocol by correctly returning, per this specification, at least one valid 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 11 "User Agent Implementation Advice".

7.2. HTTP Request Type

If an HSTS Host receives a HTTP request message over a non-secure transport, it SHOULD send a 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 12 "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 [owaspTLSSGuide].
- * 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 a 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 9 "Domain Name IDNA-Canonicalization" prior to the processing specified in this section.

The above assumptions mean that this processing model also specifically assumes that appropriate IDNA and Unicode validations and character list testing have 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 14.8 "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"),

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 if the HSTS Host is known, or, MUST NOT note this HSTS Host if it is not yet known.

If a UA receives more than one STS header field in a 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

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 flag is asserted or not. See also Section 10.1 "HSTS Policy expiration time considerations".

Note: The UA MUST NOT modify the expiry time nor the includeSubDomains flag of any superdomain matched Known HSTS Host.

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. Or, there may be no match.

The below steps 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 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: 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".

If a superdomain match with an asserted includeSubDomains flag is found, or if a congruent match is found -- without any found superdomain matches having asserted includeSubDomains flags -- 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: The the above steps ensure that the HSTS Policy applies to HTTP over any TCP port of an HSTS Host.

8.4. Errors in Secure Transport Establishment

When connecting to a Known HSTS Host, the UA MUST terminate the connection (see also Section 11 "User Agent Implementation Advice") if there are any errors (e.g., certificate errors), whether "warning" or "fatal" or any other error level, with the underlying secure transport. This includes any issues with certificate revocation checking whether via the Certificate Revocation List (CRL) [RFC5280], or via the Online Certificate Status Protocol (OCSP) [RFC2560].

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 they 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. 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 a 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 the latter specification).

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 a 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 Section 13 "Internationalized Domain Names for Applications (IDNA): Dependency and Migration" and Section 14.8 "Internationalized Domain Names" of this specification for further details and considerations.

10. Server Implementation and Deployment Advice

This section is non-normative.

10.1. 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 on 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 wilfully set max-age in order to have UAs enforce the HSTS Policy for their host, and protects them from inadvertently enabling HSTS with some arbitrary non-zero duration.

10.2. 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 and this certificate does not match a usable TLS certificate association (as defined by Section 4 of the TLSA protocol specification [I-D.ietf-dane-protocol]),

...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 14.6 "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.

10.3. Implications of includeSubDomains

The includeSubDomains directive has some practical implications. For example, if an HSTS Host offers HTTP-based services on various ports or at various subdomains of its host domain name, then they will all have to be available over secure transport in order to work properly.

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 which 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 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. 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:

11.1. 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 re-try.

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 advertises HSTS, then it is opting into this scheme, whereby all certificate errors or warnings cause a connection termination, with no chance to "fool" the user into making the wrong decision and compromising themselves.

11.2. 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 Section 14.4 "Bootstrap MITM Vulnerability".

Note: Such a feature is difficult to get right on a per-site basis. See the discussion of "rewrite rules" in Section 5.5 of [ForceHTTPS]. For example, arbitrary web sites may not materialize all their URIs using the "https" scheme, and thus could "break" if a UA were to attempt to access the site exclusively using such URIs. Also note that this feature would complement, but is independent of, an "HSTS Pre-Loaded List" feature (see Section 11.3).

11.3. 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 Section 14.4 "Bootstrap MITM Vulnerability".

Note: Such a facility would complement a "User-declared HSTS Policy" feature.

11.4. Disallow Mixed Security Context Loads

"Mixed security context" loads happen when an 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.

11.5. 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 users get used to 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.

12. Constructing an Effective Request URI

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

HTTP requests often do not carry an 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.

12.1. ERU Fundamental Definitions

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

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

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

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

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

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

12.2. Determining the Effective 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 `/`.

12.2.1. 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`.

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 below sections 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. Ramifications of HSTS Policy Establishment only over Error-free Secure Transport

The User Agent Processing Model defined in Section 8, 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 14.4 "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 14.3 "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 an 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 proxy -- within a corporate intranet, for example -- and interacts with an unknown HSTS Host beyond the proxy, the user will 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 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 11 "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.

14.2. The Need for includeSubDomains

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

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

Now, if an attacker causes the UA to request a subdomain name that is unlikely to already exist in the web application, such as "https://uxdhbpahpdsf.example.com/", but 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", and,
2. The HTTP request sent to uxdhbpahpdsf.example.com will include the Secure-flagged domain cookie.
3. If "uxdhbpahpdsf.example.com" returns a certificate during TLS establishment, and the user clicks through any warning that might be 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.

14.3. 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 that are -- or critical portions of them 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 for critical portions of its web application, then the web application will be rendered unusable for the UA's user.

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

- * If the web application has a 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, then the attacker can 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 their UAs offer such a capability (see Section 11 "User Agent Implementation Advice"). Or, 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.

14.4. Bootstrap MITM Vulnerability

The bootstrap MITM (Man-In-The-Middle) vulnerability is a vulnerability users and HSTS Hosts encounter in the situation where the user manually enters, or follows a link, to an unknown HSTS Host using a "http" URI rather than a "https" URI. Because the UA uses an insecure channel in the initial attempt to interact with the specified 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 11 "User Agent Implementation Advice".

14.5. Network Time Attacks

Active network attacks can subvert network time protocols (such as 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].

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

An attacker could conceivably obtain a victim HSTS-protected web application's users' login credentials 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 attackers' 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 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.

14.7. 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 an HSTS Host(s) 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 clever loaded page construction causing the UA to send queries to (variations of) the encoded domain names. Such queries can reveal whether the UA had prior visited the original HSTS Host (and subdomains).

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

14.8. 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 9 "Domain Name IDNA-Canonicalization"). These steps are necessary in order to avoid various potentially compromising situations.

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

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

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

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

[I-D.ietf-dane-protocol]

Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", draft-ietf-dane-protocol-23 (work in progress), June 2012.

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

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

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

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

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

This specification is referenced due to its ongoing relevance to actual deployments for the foreseeable future.

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

This specification is referenced due to its ongoing relevance to actual deployments for the foreseeable future.

[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.

- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.
- [RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.
- [RFC5895] Resnick, P. and P. Hoffman, "Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008", RFC 5895, September 2010.
- [UTS46] Davis, M. and M. Suignard, "Unicode IDNA Compatibility Processing", Unicode Technical Standards # 46, <<http://unicode.org/reports/tr46/>>.
- [Unicode] The Unicode Consortium, "The Unicode Standard", <<http://www.unicode.org/versions/latest/>>.
- [W3C.REC-html401-19991224] Hors, A., Raggett, D., 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://wirelesscenter.dk/Crypt/wifi-security-attacks/Practical%20Attacks%20Against%20WEP%20and%20WPA.pdf>>.
- [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, on-going,
<[https://secure.wikimedia.org/wikipedia/en/wiki/
Firesheep](https://secure.wikimedia.org/wikipedia/en/wiki/Firesheep)>.

[ForceHTTPS]

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

[GoodDhamijaEtAl05]

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

[I-D.ietf-tls-ssl-version3]

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

This is the canonical reference for SSLv3.0.

[JacksonBarth2008]

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

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

[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.

- [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.
- [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, April 2011.
- [RFC6376] Crocker, D., Hansen, T., and M. Kucherawy, "DomainKeys Identified Mail (DKIM) Signatures", RFC 6376, September 2011.
- [RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.
- [SunshineEgelmanEtAl09]
Sunshine, J., Egelman, S., Almuhiemedi, 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.REC-wsc-ui-20100812]
Saldhana, A. and T. Roessler, "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>.
- [owaspTLSTLSGuide]
Coates, M., Wichers, d., Boberski, M., and T. Reguly, "Transport Layer Protection Cheat Sheet", Accessed: 11-Jul-2010, <<http://www.owasp.org/index.php/>>

Transport_Layer_Protection_Cheat_Sheet>.

Appendix A. Design Decision Notes

This appendix documents various design decisions.

1. Cookies aren't appropriate for HSTS Policy expression as they are potentially mutable (while stored in the UA), therefore an HTTP header field is employed.
2. We chose to not attempt to specify how "mixed security context loads" (AMA "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 flag. If the HSTS Host couldn't correct such errors over protocol, it would require some form of annunciation to users and manual intervention on 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.

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 a 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, 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, Tom Ritter, Peter Saint-Andre, Sid Stamm, Maciej Stachowiak, Andy Steingrubl, Brandon Sterne, Martin Thomson, Daniel Veditz, 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 re-writing of the effective request URI text, which he did when incorporating the ERU notion into the HTTPbis work. Subsequently, the ERU text in this spec was lifted from Julian's work in the updated HTTP/1.1 (part 1) specification and adapted to the [RFC2616] ABNF.

Appendix D. Change Log

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

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

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

Changes from -10 to -11:

1. Various minor editorial fixes based on Barry Leiba's AD review, as well as ID-Nits warnings.
2. Clarification addition of directive-name and directive-value to Strict-Transport-Security ABNF in Section 6.1, from Barry's AD review. <<https://www.ietf.org/mail-archive/web/websec/current/msg01265.html>>
3. Moved ref to RFC5894 to Informational since it is a truly informational reference.

Changes from -09 to -10:

1. Added "(including when following HTTP redirects [RFC2616])" to section 8.3. This addresses issue ticket #47. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/47>>
2. Fixed max-age value in section 10.1. Substituted 7776000 (actually 90 days) for 778000 (only 9 days). This addresses issue ticket #48. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/48>>
3. Added mention of "Certificate Status Request" TLS extension [RFC6066] aka "OCSP stapling" to example in section 10.3. This addresses issue ticket #49. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/49>>

Changes from -08 to -09:

1. Added IESG Note to Section 3 "Conformance Criteria" per Barry Leiba's suggestion on the mailing list. <<https://www.ietf.org/mail-archive/web/websec/current/msg01200.html>>
2. Added additional requirement #5 to requirements for STS header field directives in Section 6.1 per Alexey's review. This completes the addressing of issue ticket #45. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/45>>
3. Addressed editorial feedback in Murray's AppsDir review of -06.

Most all of these changes were addressing detailed/small editorial items, however note the addition of a couple of introductory paragraphs in the Security Considerations section, as well as a re-written and expanded Section 14.6 "Bogus Root CA Certificate Phish plus DNS Cache Poisoning Attack", as well the new item #5 to Appendix A "Design Decision Notes".

This addresses issue ticket #46.

<<http://trac.tools.ietf.org/wg/websec/trac/ticket/46>>

Changes from -07 to -08:

1. Clarified requirement #4 for STS header field directives in Section 6.1, and removed "(which "update" this specification)". Also added explicit "max-age=0" to Section 6.1.1. Reworked final sentence in 2nd para of Section 13. This addresses issue ticket #45. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/45>>

Changes from -06 to -07:

1. Various minor/modest editorial tweaks throughout as I went through it pursuing the below issue tickets. Viewing a visual diff against -06 revision recommended.
2. fixed some minor editorial issues noted in review by Alexey, fixes noted in here: <<https://www.ietf.org/mail-archive/web/websec/current/msg01163.html>>
3. Addressed ABNF exposition issues, specifically inclusion of quoted-string syntax for directive values. Fix STS header ABNF such that a leading ";" isn't required. Add example of quoted-string-encoded max-age-value. This addresses (re-opened) issue ticket #33. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/33>>

4. Reworked sections 8.1 through 8.3 to ensure matching algorithm and resultant HSTS Policy application is more clear, and that it is explicitly stipulated to not muck with attributes of superdomain matching Known HSTS Hosts. This addresses issue ticket #37.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/37>>
5. Added reference to [I-D.ietf-dane-protocol], pared back extraneous discussion in section 2.2, and updated discussion in 10.2 to accomodate TLSA (nee DANE). This addresses issue ticket #39.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/39>>
6. Addressed various editorial items from issue ticket #40.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/40>>
7. Loosened up the language regarding redirecting "http" requests to "https" in section 7.2 such that future flavors of permanent redirects are accommodated. This addresses issue ticket #43.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/43>>
8. Reworked the terminology and language in Section 9, in particular defining the term "putative domain name string" to replace "valid Unicode-encoded string-serialized domain name". This addresses issue ticket #44.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/44>>

Changes from -05 to -06:

1. Addressed various editorial comments provided by Tobias G. This addresses issue ticket #38.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/38>>

Changes from -04 to -05:

1. Fixed up references to move certain ones back to the normative section -- as requested by Alexey M. Added explanation for referencing obsoleted [RFC3490] and [RFC3492]. This addresses issue ticket #36.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/36>>
2. Made minor change to Strict-Transport-Security header field ABNF in order to address further feedback as appended to ticket #33. This addresses issue ticket #33.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/33>>

Changes from -03 to -04:

1. Clarified that max-age=0 will cause UA to forget a known HSTS Host, and more generally clarified that the "freshest" info from the HSTS Host is cached, and thus HSTS Hosts are able to alter the cached max-age in UAs. This addresses issue ticket #13. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/13>>
2. Updated section on "Constructing an Effective Request URI" to remove remaining reference to RFC3986 and reference RFC2616 instead. Further addresses issue ticket #14. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/14>>
3. Addresses further ABNF issues noted in comment:1 of issue ticket #27. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/27#comment:1>>
4. Reworked the introduction to clarify the denotation of "HSTS Policy" and added the new Appendix B summarizing the primary characteristics of HSTS Policy and Same-Origin Policy, and identifying their differences. Added ref to [RFC4732]. This addresses issue ticket #28. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/28>>
5. Reworked language in Section 2.3.1.3. wrt "mixed content", more clearly explain such vulnerability, disambiguate "mixed content" in web security context from its usage in markup language context. This addresses issue ticket #29. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/29>>
6. Expanded Denial of Service discussion in Security Considerations. Added refs to [RFC4732] and [CWE-113]. This addresses issue ticket #30. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/30>>
7. Mentioned in prose the case-insensitivity of directive names. This addresses issue ticket #31. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/31>>
8. Added Section 10.3 "Implications of includeSubDomains". This addresses issue ticket #32. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/32>>
9. Further refines text and ABNF definitions of STS header field directives. Retains use of quoted-string in directive grammar. This addresses issue ticket #33. <<http://trac.tools.ietf.org/wg/websec/trac/ticket/33>>

10. Added Section 14.7 "Creative Manipulation of HSTS Policy Store", including reference to [WebTracking]. This addresses issue ticket #34.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/34>>
11. Added Section 14.1 "Ramifications of HSTS Policy Establishment only over Error-free Secure Transport" and made some accompanying editorial fixes in some other sections. This addresses issue ticket #35.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/35>>
12. Refined references. Cleaned out un-used ones, updated to latest RFCs for others, consigned many to Informational. This addresses issue ticket #36.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/36>>
13. Fixed-up some inaccuracies in the "Changes from -02 to -03" section.

Changes from -02 to -03:

1. Updated section on "Constructing an Effective Request URI" to remove references to RFC3986. Addresses issue ticket #14.
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/14>>
2. Reference RFC5890 for IDNA, retaining subordinate refs to RFC3490. Updated IDNA-specific language, e.g. domain name canonicalization and IDNA dependencies. Addresses issue ticket #26
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/26>>.
3. Completely re-wrote the STS header ABNF to be fully based on RFC2616, rather than a hybrid of RFC2616 and httpbis. Addresses issue ticket #27
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/27>>.

Changes from -01 to -02:

1. Updated Section 8.3 "URI Loading and Port Mapping" fairly thoroughly in terms of refining the presentation of the steps, and to ensure the various aspects of port mapping are clear. Nominally fixes issue ticket #1
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/1>>
2. Removed dependencies on [I-D.draft-ietf-httpbis-pl-messaging-15]. Thus updated STS ABNF in Section 6.1 "Strict-Transport-Security HTTP Response Header Field" by lifting some productions entirely from

[I-D.draft-ietf-httpbis-pl-messaging-15] and leveraging [RFC2616]. Addresses issue ticket #2
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/2>>.

3. Updated Effective Request URI section and definition to use language from [I-D.draft-ietf-httpbis-pl-messaging-15] and ABNF from [RFC2616]. Fixes issue ticket #3
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/3>>.
4. Added explicit mention that the HSTS Policy applies to all TCP ports of a host advertising the HSTS Policy. Nominally fixes issue ticket #4
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/4>>
5. Clarified the need for the "includeSubDomains" directive, e.g. to protect Secure-flagged domain cookies. In Section 14.2 "The Need for includeSubDomains". Nominally fixes issue ticket #5
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/5>>
6. Cited Firesheep as real-live threat in Section 2.3.1.1 "Passive Network Attackers". Nominally fixes issue ticket #6
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/6>>.
7. Added text to Section 11 "User Agent Implementation Advice" justifying connection termination due to tls warnings/errors. Nominally fixes issue ticket #7
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/7>>.
8. Added new subsection Section 8.6 "Missing Strict-Transport-Security Response Header Field". Nominally fixes issue ticket #8
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/8>>.
9. Added text to Section 8.4 "Errors in Secure Transport Establishment" explicitly note revocation check failures as errors causing connection termination. Added references to [RFC5280] and [RFC2560]. Nominally fixes issue ticket #9
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/9>>.
10. Added a sentence, noting that distributing specific end-entity certificates to browsers will also work for self-signed/private-CA cases, to Section 10 "Server Implementation and Deployment Advice" Nominally fixes issue ticket #10
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/10>>.
11. Moved "with no user recourse" language from Section 8.4 "Errors in Secure Transport Establishment" to Section 11

"User Agent Implementation Advice". This nominally fixes issue ticket #11

<<http://trac.tools.ietf.org/wg/websec/trac/ticket/11>>.

12. Removed any and all dependencies on [I-D.draft-ietf-httpbis-pl-messaging-15], instead depending on [RFC2616] only. Fixes issue ticket #12
<<http://trac.tools.ietf.org/wg/websec/trac/ticket/12>>.
13. Removed the inline "XXX1" issue because no one had commented on it and it seems reasonable to suggest as a SHOULD that web apps should redirect incoming insecure connections to secure connections.
14. Removed the inline "XXX2" issue because it was simply for raising consciousness about having some means for distributing secure web application metadata.
15. Removed "TODO1" because description prose for "max-age" in the Note following the ABNF in Section 6 seems to be fine.
16. Decided for "TODO2" that "the first STS header field wins". TODO2 had read: "Decide UA behavior in face of encountering multiple HSTS headers in a message. Use first header? Last?". Removed TODO2.
17. Added Section 1.1 "Organization of this specification" for readers' convenience.
18. Moved design decision notes to be a proper appendix Appendix A.

Changes from -00 to -01:

1. Changed the "URI Loading" section to be "URI Loading and Port Mapping".
2. (HASMAT) reference changed to (WEBSEC).
3. Changed "server" -> "host" where applicable, notably when discussing "HSTS Hosts". Left as "server" when discussing e.g. "http server"s.
4. Fixed minor editorial nits.

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

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

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

Changes from -01 to -02:

1. updated abstract such that means for expressing HSTS Policy other than via HSTS header field is noted.
2. Changed spec title to "HTTP Strict Transport Security (HSTS)" from "Strict Transport Security". Updated use of "STS" acronym throughout spec to HSTS (except for when specifically discussing syntax of Strict-Transport-Security HTTP Response Header field), updated "Terminology" appropriately.
3. Updated the discussion of "Passive Network Attackers" to be more precise and offered references.
4. Removed para on normative/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 explanatory "Note:" and reference to "HTTP Request Type" section. Added "XXX1" issue.
9. Added conformance clause to "URI Loading".
10. Moved "Notes for STS Server implementers:" from "UA Implementation Advice" to "HSTS Policy expiration time considerations:" in "Server Implementation Advice", and also noted another option.
11. Added cautionary "Note:" to "Ability to delete UA's cached HSTS Policy on a per HSTS Server basis".
12. Added some informative references.
13. Various minor editorial fixes.

Changes from -00 to -01:

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

Authors' Addresses

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

Email: Jeff.Hodges@PayPal.com

Collin Jackson
Carnegie Mellon University

Email: collin.jackson@sv.cmu.edu

Adam Barth
Google, Inc.

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

WEBSEC
Internet-Draft
Intended status: Informational
Expires: January 4, 2013

D. Ross
Microsoft
T. Gondrom
July 3, 2012

HTTP Header X-Frame-Options
draft-ietf-websec-x-frame-options-00

Abstract

To improve the protection of web applications against Clickjacking this standards defines a http response header that declares a policy communicated from a host to the client browser whether the transmitted content MUST NOT be displayed in frames of other pages from different origins which are allowed to frame the content. This drafts serves to document the existing use and specification of X-Frame-Options.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2013.

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	3
1.1. Requirements Language	3
2. X-Frame-Options Header	3
2.1. Syntax	3
2.2. Backus-Naur Form (BNF)	4
2.3. Design Issues	5
2.3.1. Enable HTML content from other domains	5
2.3.2. Browser Behaviour and Processing	5
2.4. Examples of X-Frame-Options Headers	5
2.4.1. Example scenario for the ALLOW-FROM parameter	6
3. Acknowledgements	6
4. IANA Considerations	6
4.1. Registration Template	7
5. Security Considerations	7
6. References	7
6.1. Normative References	7
6.2. Informative References	7
Appendix A. Description of a Clickjacking attack	8
A.1. Shop	8
A.2. Confirm Purchase Page	8
A.3. Flash Configuration	9
Authors' Addresses	9

1. Introduction

In 2009 and 2010 many browser vendors introduced the use of a non-standard http header RFC 2616 [RFC2616] "X-Frame-Options" to protect against Clickjacking [Clickjacking]. This draft is to document the current use of X-Frame-Options header and shall in the future be replaced by the Frame-Options (CSRF) [FRAME-OPTIONS] standard.

Existing anti-ClickJacking measures, e.g. Frame-breaking Javascript, have weaknesses so that their protection can be circumvented as a study [FRAME-BUSTING] demonstrated.

Short of configuring the browser to disable frames and script entirely, which massively impairs browser utility, browser users are vulnerable to this type of attack.

The "X-Frame-Options" allows a secure web page from host B to declare that its content (for example a button, links, text, etc.) must not be displayed in a frame of another page (e.g. from host A). In principle this is done by a policy declared in the HTTP header and obeyed by conform browser implementations.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

2. X-Frame-Options Header

The X-Frame-Options HTTP response header indicates a policy whether a browser MUST NOT allow to render a page in a <frame> or <iframe> . Hosts can declare this policy in the header of their HTTP responses to prevent clickjacking attacks, by ensuring that their content is not embedded into other pages or frames.

2.1. Syntax

The header field name is:
X-Frame-Options

There are three different values for the header field. These values are exclusive, that is NOT more than one of the three values MUST be set.

DENY

A browser receiving content with this header MUST NOT display this content in any frame.

SAMEORIGIN

A browser receiving content with this header MUST NOT display this content in any frame from a page of different origin than the content itself.

If a browser or plugin can not reliably determine whether the origin of the content and the frame have the same origin, this MUST be treated as "DENY".

[TBD]current implementations do not display if the origin of the top-level-browsing-context is different than the origin of the page containing the X-FRAME-OPTIONS header.

ALLOW-FROM (followed by a URI of trusted origins)

A browser receiving content with this header MUST NOT display this content in any frame from a page of different origin than the listed origin. While this can expose the page to risks by the trusted origin, in some cases it may be necessary to use content from other domains.

For example: X-FRAME-OPTIONS: ALLOW-FROM
https://www.domain.com/

The URIs listed for ALLOW-FROM must be valid.

Any data beyond the domain address (i.e. any data after the "/" separator) is to be ignored and to verify a referring page is of the same origin as the content or that the referring page is listed in the ALLOW-FROM list of URI, the algorithm to compare origins from [RFC6454] should be used.

Wildcards to declare multiple domains in one statement are not permitted.

[TBD] Current Implementations do not consider the port a component of the origin - conflicting with [RFC6454].

2.2. Backus-Naur Form (BNF)

The RFC 822 [RFC0822] EBNF of the X-Frame-Options header is:

```
X-Frame-Options = "Frame-Options" ":" "DENY"/ "SAMEORIGIN" /  
                  ("ALLOW-FROM" ":" URI)
```

[TBD] with URI as defined in the websec-origin draft

[TBD] Or should we use the ABNF (RFC 2234) alternatively or in

addition?

2.3. Design Issues

2.3.1. Enable HTML content from other domains

There are three main direct vectors that enable HTML content from other domains:

- o IFRAME Tag
- o Frame tag
- o The Object tag (requires a redirect)

Besides these other ways to host HTML content can be possible. For example some plugins may host HTML views directly. If these plugins appear essentially as frames (as opposed to top-level windows), the plugins MUST conform to the X-FRAME-OPTIONS directive as specified in this draft as well.

2.3.2. Browser Behaviour and Processing

To allow secure implementations browser implementations MUST behave in a consistent and reliable way.

If a HTTP Header prohibits framing, the user-agent of the browser MAY immediately abort downloading or parsing of the document.

When a browser discovers loaded content with the X-FRAME-OPTIONS header would be displayed in a frame against the specified origin orders of the header, the browser SHOULD redirect as soon as possible to a "No-Frame" page.

"No-Frame" Page

If the display of content is denied by the X-FRAME-OPTIONS header an error page SHOULD be displayed. For example this can be a noframe.html page also stating the full URL of the protected page and the hostname of the protected page.

The NoFrame page MAY provide the user with an option to open the target URL in a new window.

2.4. Examples of X-Frame-Options Headers

2.4.1. Example scenario for the ALLOW-FROM parameter

1. Inner IFRAME suggests via a querystring parameter what site it wants to be hosted by. This can obviously be specified by an attacker, but that's OK.
2. Server verifies the hostname meets whatever criteria. For example, for a Facebook "Like" button, the server can check to see that the supplied hostname matches the hostname expected for that Like button.
3. Server serves up the hostname in X-FRAME-OPTIONS: ALLOW-FROM if the proper criteria was met in step #2.
4. Browser enforces the X-FRAME-OPTIONS: ALLOW-FROM domain.com header.

3. Acknowledgements

This document was derived from input from specifications published by various browser vendors like Microsoft (Eric Lawrence, David Ross), Mozilla, Google, Opera and Apple.

4. IANA Considerations

This memo a request to IANA to include the specified HTTP header in registry as outlined in Registration Procedures for Message Header Fields [RFC3864]

4.1. Registration Template

PERMANENT MESSAGE HEADER FIELD REGISTRATION TEMPLATE:

Header field name: X-Frame-Option

Applicable protocol: http [RFC2616]

Status: Standard

Author/Change controller: IETF

Specification document(s): draft-ietf-websec-x-frame-options

Related information:

Figure 1

5. Security Considerations

The introduction of the http header X-FRAME-OPTIONS does improve the protection against Clickjacking, however it is not self-sufficient on its own but MUST be used in conjunction with other security measures like secure coding and Content Security Policy (CSP)

The parameter ALLOW-FROM allows a page to guess who is framing it. This is by design, but may lead to data leakage or data protection concerns.

6. References

6.1. Normative References

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

6.2. Informative References

[CLICK-DEFENSE-BLOG]
Microsoft, "Clickjacking Defense", 2009, <<http://blogs.msdn.com/b/ie/archive/2009/01/27/ie8-security-part-vii-clickjacking-defenses.aspx>>.

[Clickjacking]
OWASP (Open Web Application Security Project),

"Clickjacking", 2010,
<<http://www.owasp.org/index.php/Clickjacking>>.

[FRAME-BUSTING]

Stanford Web Security Research, "Busting frame busting: a study of clickjacking vulnerabilities at popular sites", 2010, <<http://seclab.stanford.edu/websec/framebusting/>>.

[FRAME-OPTIONS]

IETF, "The Web Origin Concept", December 2010, <<http://tools.ietf.org/id/draft-gondrom-frame-options-02.txt>>.

[RFC0822] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[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.

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

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

Appendix A. Description of a Clickjacking attack

More detailed explanation of Clickjacking scenarios

A.1. Shop

An Internet Marketplace/Shop offering a feature with a link/button to "Buy this" Gadget

The marketplace wants their affiliates (who could be bad guys) to be able to stick the "Buy such-and-such from XYZ" IFRAMES into their pages. There is a ClickJack possibility here, which is why the marketplace/onlineshop needs to then immediately navigate the main browsing context (or a new window) to a confirmation page which is protected by anti-CJ protections.

A.2. Confirm Purchase Page

Onlineshop "Confirm purchase" anti-CSRF page

The Confirm Purchase page must be shown to the end user without possibility of overlay or misuse by an attacker. For that reason, the confirmation page uses anti-CSRF tokens and contains the X-FRAME-

OPTIONS directive, mitigating ClickJack attacks.

A.3. Flash Configuration

Macromedia Flash configuration page
Macromedia Flash configuration settings are set by a Flash object which can run only from a specific configuration page on Macromedia's site. The object runs inside the page and thus can be subject to a ClickJacking attack. In order to prevent ClickJacking attacks against the security settings, the configuration page uses the X-FRAME-OPTIONS directive.

Authors' Addresses

David Ross
Microsoft
U.S.

Phone:
Email:

Tobias Gondrom
Kruegerstr. 5A
Unterschleissheim,
Germany

Phone: +44 7521003005
Email: tobias.gondrom@gondrom.org

