HTTPbis
Internet-Draft
Updates: 6265 (if approved)
Intended status: Standards Track
Expires: March 11, 2016

M. West
Google, Inc
M. Goodwin
Mozilla

September 8, 2015

First-Party-Only Cookies
draft-west-first-party-cookies-04

Abstract

This document updates RFC6265 by defining a "First-Party-Only" attribute which allows servers to assert that a cookie ought to be sent only in a "first-party" context. This assertion allows user agents to mitigate the risk of cross-origin information leakage, and provides some minimal protection against cross-site request forgery 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 March 11, 2016.

Table of Contents

1. Introduction 2
1.1. Goals 3
1.2. Limitations 3
1.3. Examples 4
2. Terminology and notation 4
2.1. First-party and Third-party Requests 5
2.1.1. Document-based requests 5
2.1.2. Worker-based requests 7
3. Server Requirements 8
3.1. Grammar 8
3.2. Semantics of the "First-Party-Only" Attribute (Non-Normative) 9
4. User Agent Requirements 9
4.1. The "First-Party-Only" attribute 9
4.2. Monkey-patching the Storage Model 9
4.3. Monkey-patching the "Cookie" header 10
5. Authoring Considerations 10
5.1. Mashups and Widgets 10
6. Privacy Considerations 10
7. References 11
7.1. Normative References 11
7.2. Informative References 12
Appendix A. Acknowledgements 12
Authors' Addresses 12

1. Introduction

Section 8.2 of [RFC6265] eloquently notes that cookies are a form of ambient authority, attached by default to requests the user agent sends on a user's behalf. Even when an attacker doesn't know the contents of a user's cookies, she can still execute commands on the user's behalf (and with the user's authority) by asking the user agent to send HTTP requests to unwary servers.

Here, we update [RFC6265] with a simple mitigation strategy that allows servers to declare certain cookies as "First-party-only", meaning they should be attached to requests if and only if those requests occur in a first-party context (as defined in section 2.1).

Note that the mechanism outlined here is backwards compatible with the existing cookie syntax. Servers may serve first-party cookies to all user agents; those that do not support the "First-Party-Only"

attribute will simply store a cookie which is returned in all applicable contexts, just as they do today.

1.1. Goals

These first-party-only cookies are intended to provide a solid layer of defense-in-depth against attacks which require embedding an authenticated request into an attacker-controlled context:

1. Timing attacks which yield cross-origin information leakage (such as those detailed in [pixel-perfect]) can be substantially mitigated by setting the "First-Party-Only" attribute on authentication cookies. The attacker will only be able to embed unauthenticated resources, as embedding mechanisms such as "<iframe>" will not create first-party contexts.

2. Cross-site script inclusion (XSSI) attacks are likewise mitigated by setting the "First-Party-Only" attribute on authentication cookies. The attacker will not be able to include authenticated resources via "<script>" or "<link>", as these embedding mechanisms will not create first-party contexts.

First-party-only cookies also mitigate one specific kind of cross-site request forgery (CSRF) attack by treating cross-origin requests with unsafe methods (including top-level navigations) as as third-party requests.

Aside from these attack mitigations, first-party-only cookies can also be useful for policy or regulatory purposes. That is, it may be valuable for an origin to assert that its cookies should not be sent along with third-party requests in order to limit its exposure to non-technical risk.

1.2. Limitations

First-party-only cookies provide reasonable defense in depth against CSRF attacks that rely on unsafe HTTP methods (like "POST"). They do not offer a robust defense against CSRF as a general category of attack:

1. Attackers can still pop up new windows or trigger top-level navigations in order to create a first-party context (as described in section 2.1), which is only a speedbump along the road to exploitation.

2. Features like "<link rel='prerender'>" [prerendering] can be exploited to create first-party contexts without the risk of user detection.

In addition to the usual server-side defenses (CSRF tokens, ensuring that "safe" HTTP methods are idempotent, etc), client-side techniques such as those described in [app-isolation] may prove effective against CSRF, and are certainly worth exploring in combination with first-party-only cookies. First-party-only cookies on their own, however, are not a barrier to CSRF attacks as a general category.

1.3. Examples

First-party-only cookies are set via the "First-Party-Only" attribute in the "Set-Cookie" header field. That is, given a server's response to a user agent which contains the following header field:

```
Set-Cookie: SID=31d4d96e407aad42; First-Party-Only
```

Subsequent requests from that user agent can be expected to contain the following header field if and only if both the requested resource and the resource in the top-level browsing context match the cookie.

2. Terminology and notation

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].

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234].

Two sequences of octets are said to case-insensitively match each other if and only if they are equivalent under the "i;ascii-casemap" collation defined in [RFC4790].

The terms "active document", "ancestor browsing context", "browsing context", "document", "iframe srcdoc document", "parent browsing context", and "top-level browsing context" are defined in the HTML Living Standard [HTML].

"Web Workers", "dedicated workers", "owner document", "list of relevant documents", and "shared workers" are defined in the Web Workers specification [WORKERS].

"Service Workers" are defined in the Service Workers specification [SERVICE-WORKERS].

The term "origin", the mechanism of deriving an origin from a URI, and the "the same" matching algorithm for origins are defined in [RFC6454].

"Safe" HTTP methods include "GET", "HEAD", "OPTIONS", and "TRACE", as defined in Section 4.2.1 of [RFC7231].

The term "public suffix" is defined in a note in Section 5.3 of [RFC6265] as "a domain that is controlled by a public registry". For example, "example.com"'s public suffix is "com". User agents SHOULD use an up-to-date public suffix list, such as the one maintained by Mozilla at [PSL].

An origin's "registerable domain" is the origin's host's public suffix plus the label to its left. That is, "https://www.example.com"'s registerable domain is "example.com". This concept is defined more rigorously in [PSL].

2.1. First-party and Third-party Requests

2.1.1. Document-based requests

When considering a request generated while parsing a document, or executing script in its context, we need to consider the document's origin as well as the origin of each of it's ancestors, in order to determine whether the request should be considered "first-party".

For this kind of request, the URI displayed in a user agent's address bar is the only security context directly exposed to users, and therefore the only signal users can reasonably rely upon to determine whether or not they trust a particular website. The origin of that URI is, therefore, the "first-party origin".

In order to prevent the kinds of "multiple-nested scenarios" described in Section 4 of [RFC7034], we must check the first-party origin against the origins of each of a document's ancestor browsing contexts' active documents. A document is considered a "first-party context" if and only if the registerable domain of the origin of its URI is the same as the registerable domain of the first-party origin, *and* if each of the active documents in its ancestors' browsing contexts' is a first-party context.

This definition has a few implications:

- New windows create new first-party contexts (as the active document is rendered into a top-level browsing context).

- Full-page navigations create new first-party contexts. Notably, this includes both HTTP and "<meta>"-driven redirects.

- "<iframe>"s do not create new first-party contexts; their requests MUST be considered in the context of the origin of the URL the user actually sees in the user agent's address bar.

To be more precise, given an HTTP request "request", the following algorithm returns "First-Party" if "request" is a first-party request, and "Third-Party" otherwise:

1. Let "document" be the document responsible for "request".

2. If "document" is a first-party context, and "request"'s URI's origin's registerable domain is the same the registerable domain of the origin of the URI of the active document in the top-level browsing context of "document", then return "First-Party".

3. Return "Third-Party".

Given a Document "document", the following algorithm returns "First-Party" if "document" is a first-party context, and "Third-Party" otherwise:

1. Let "top-origin" be the origin of the URI of the active document in the top-level browsing context of the document responsible for "request".

2. If "document"'s URI's origin is not "top-origin", return "Third-Party".

3. While "document" has a parent browsing context:

    1. Let "document" be "document"'s parent browsing context's active document.

    2. If "document"'s URI's origin does not have the same registerable domain as "top-origin" and "document" is not an iframe srcdoc document, return "Third-Party".

4. Return "First-Party".

Note that we deal with the document's location in steps 2, 3, and 4.2 above, not with the document's origin. For example, a top-level document from "https://example.com" which has been sandboxed into a unique origin still creates a non-unique first-party context for subsequent requests.

2.1.2. Worker-based requests

Worker-driven requests aren't as clear-cut as document-driven requests, as there isn't a clear link between a top-level browsing context and a worker. This is especially true for Service Workers [SERVICE-WORKERS], which may execute code in the background, without any document visible at all.

Note: The descriptions below assume that workers must be same-origin with the documents that instantiate them. If this invariant changes, we'll need to take the worker's script's URI into account when determining their status.

2.1.2.1. Dedicated Workers

Dedicated workers are fairly straightforward to categorize, as each dedicated worker is bound to one and only one Document. Requests generated from a dedicated worker (via "importScripts", "XMLHttpRequest", "fetch()", and so on) are first-party requests if and only if the worker's owner document is a first-party context.

To be more precise, given an HTTP request "request", the following algorithm returns "First-Party" if "request" is a first-party request, and "Third-Party" otherwise:

1. Let "worker" be the dedicated worker responsible for "request".

2. Let "document" be "worker"'s owner document.

3. If "document" is a first-party context, and "request"'s URI's origin's registerable domain is the same as the registerable domain of the origin of the URI of the active document in the top-level browsing context of "document", then return "First-Party".

4. Return "Third-Party".

2.1.2.2. Shared Workers

Shared Workers introduce the complexity of bindings to multiple Documents. As it is quite possible for a shared worker to be bound at the same time to one Document that is a first-party context, and another that isn't, we'll need to walk through all the documents associated with the worker to determine its status. If and only if all associated documents are first-party contexts, then the worker is a first-party context.

To be more precise, given an HTTP request "request", the following algorithm returns "First-Party" if "request" is a first-party request, and "Third-Party" otherwise:

1. Let "worker" be the dedicated worker responsible for "request".

2. For each "document" in "worker"'s list of relevant Documents:

    1. Return "Third-Party" if "document" is not a first-party context (as defined in section 2.1.1).

    2. Return "Third-Party" if "request"'s URI's origin's registerable domain is not the same as the registerable domain of the origin of the URI of the active document in the top-level browsing context of "document".

3. Return "First-Party".

2.1.2.3. Service Workers

Service Workers are more complex still, as they act as a completely separate execution context, with very little relationship to the Document which registered them.

Until we have more implementation experience, we will consider Service Workers as third-party contexts in all cases.

Note: Requests which simply pass through a service worker will be handled as described above; the only requests which will be effected by this categorization are those which the service worker itself initiates (via "fetch()", for instance).

3. Server Requirements

This section describes extensions to [RFC6265] necessary to implement the server-side requirements of the "First-Party-Only" attribute.

3.1. Grammar

Add "First-Party-Only" to the list of accepted attributes in the "Set-Cookie" header field's value by replacing the "cookie-av" token definition in Section 4.1.1 of [RFC6265] with the following ABNF grammar:

```
cookie-av           = expires-av / max-age-av / domain-av /
                      path-av / secure-av / httponly-av /
                      first-party-only-av / extension-av
first-party-only-av = "First-Party-Only"
```

3.2. Semantics of the "First-Party-Only" Attribute (Non-Normative)

The "First-Party-Only" attribute limits the scope of the cookie such that it will only be attached to requests if those requests are "first-party", as described in Section 2.1. For example, requests for "https://example.com/sekrit-image" will attach first-party-only cookies if and only if the top-level browsing context is currently displaying a document from "https://example.com".

The changes to the "Cookie" header field suggested in Section 4.3 provide additional detail.

4. User Agent Requirements

This section describes extensions to [RFC6265] necessary in order to implement the client-side requirements of the "First-Party-Only" attribute.

4.1. The "First-Party-Only" attribute

The following attribute definition should be considered part of the the "Set-Cookie" algorithm as described in Section 5.2 of [RFC6265]:

If the attribute-name case-insensitively matches the string "First-Party-Only", the user agent MUST append an attribute to the "cookie-attribute-list" with an "attribute-name" of "First-Party-Only" and an empty "attribute-value".

4.2. Monkey-patching the Storage Model

Note: There's got to be a better way to specify this. Until I figure out what that is, monkey-patching!

Alter Section 5.3 of [RFC6265] as follows:

1. Add "first-party-only-flag" to the list of fields stored for each cookie.

2. Before step 11 of the current algorithm, add the following:

    1. If the "cookie-attribute-list" contains an attribute with an "attribute-name" of "First-Party-Only", set the cookie's "first-party-only-flag" to true. Otherwise, set the cookie's "first-party-only-flag" to false.

    2. If the cookie's "first-party-only-flag" is set to true, and the request which generated the cookie is not a first-party

request (as defined in Section 2.1), then abort these steps and ignore the newly created cookie entirely.

4.3. Monkey-patching the "Cookie" header

Note: There's got to be a better way to specify this. Until I figure out what that is, monkey-patching!

Alter Section 5.4 of [RFC6265] as follows:

1. Add the following requirements to the list in step 1:

    * If the cookie's "first-party-only-flag" is true, then exclude the cookie if the HTTP request is a third-party request (see Section 2.1).

    * If the cookie's "first-party-only-flag" is true, then exclude the cookie if the HTTP request's method is not safe and the registerable domain of the origin of the URI of the document which originated the request is not the same as the registerable domain of the origin of the HTTP request's URI.

Note that the modifications suggested here concern themselves only with the origins of ancestor browsing contexts and the origin of the resource being requested. The cookie's "domain", "path", and "secure" attributes do not come into play for these comparisons.

5. Authoring Considerations

5.1. Mashups and Widgets

The "First-Party-Only" attribute is inappropriate for some important use-cases. In particular, note that content intended for embedding in a third-party context (social networking widgets or commenting services, for instance) will not have access to first-party-only cookies. Non-first-party cookies may be required in order to provide seamless functionality that relies on a user's state.

Likewise, some forms of Single-Sign-On might require authentication in a third-party context; these mechanisms will not function as intended with first-party-only cookies.

6. Privacy Considerations

First-party-only cookies in and of themselves don't do anything to address the general privacy concerns outlined in Section 7.1 of [RFC6265]. The attribute is set by the server, and serves to mitigate the risk of certain kinds of attacks that the server is

worried about. The user is not involved in this decision. Moreover, a number of side-channels exist which could allow a server to link distinct requests even in the absence of cookies. Connection and/or socket pooling, Token Binding, and Channel ID all offer explicit methods of identification that servers could take advantage of.

7. References

7.1. Normative References

[HTML] Hickson, I., "HTML Living Standard", n.d., <https://html.spec.whatwg.org/>.

[PSL] "Public Suffix List", n.d., <https://publicsuffix.org/list/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4790] Newman, C., Duerst, M., and A. Gulbrandsen, "Internet Application Protocol Collation Registry", RFC 4790, DOI 10.17487/RFC4790, March 2007, <http://www.rfc-editor.org/info/rfc4790>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <http://www.rfc-editor.org/info/rfc6265>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc6454>.

[RFC7034] Ross, D. and T. Gondrom, "HTTP Header Field X-Frame-Options", RFC 7034, DOI 10.17487/RFC7034, October 2013, <http://www.rfc-editor.org/info/rfc7034>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[SERVICE-WORKERS]
Russell, A., Song, J., and J. Archibald, "Service Workers", n.d., <http://www.w3.org/TR/service-workers/>.

[WORKERS] Hickson, I., "Web Workers", n.d., <http://www.w3.org/TR/workers/>.

7.2. Informative References

[app-isolation]
Chen, E., Bau, J., Reis, C., Barth, A., and C. Jackson, "App Isolation - Get the Security of Multiple Browsers with Just One", n.d., <http://www.collinjackson.com/research/papers/appisolation.pdf>.

[pixel-perfect]
Stone, P., "Pixel Perfect Timing Attacks with HTML5", n.d., <http://www.contextis.com/documents/2/Browser_Timing_Attacks.pdf>.

[prerendering]
Bentzel, C., "Chrome Prerendering", n.d., <https://www.chromium.org/developers/design-documents/prerender>.

[samedomain-cookies]
Goodwin, M. and J. Walker, "SameDomain Cookie Flag", 2011, <http://people.mozilla.org/~mgoodwin/SameDomain/samedomain-latest.txt>.

Appendix A. Acknowledgements

The first-party cookie concept documented here is indebited to Mark Goodwin's and Joe Walker's [samedomain-cookies]. Michal Zalewski, Artur Janc, and Ryan Sleevi provided particularly valuable feedback on this document.

Authors' Addresses

Mike West
Google, Inc

Email: mkwst@google.com
URI: https://mikewest.org/

Mark Goodwin
Mozilla

Email: mgoodwin@mozilla.com
URI: https://www.computerist.org/