Incrementally Better Cookies
draft-west-cookie-incrementalism-00

Abstract

This document proposes two changes to cookies inspired by the properties of the HTTP State Tokens mechanism proposed in [I-D.west-http-state-tokens]. First, cookies should be treated as "SameSite=Lax" by default. Second, cookies that explicitly assert "SameSite=None" in order to enable cross-site delivery should also be marked as "Secure".

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 https://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 November 8, 2019.

Copyright Notice

Copyright (c) 2019 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 (https://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
1. Introduction

The HTTP State Tokens proposal ([I-D.west-http-state-tokens]) aims to replace cookies with a state management mechanism that has better security and privacy properties. That proposal is somewhat aspirational: it’s going to take a long time to come to agreement on the exact contours of a cookie replacement, and an even longer time to actually do so.

While we’re debating the details of a new state management primitive, it seems quite reasonable to reevaluate some aspects of the existing primitive: cookies. When we can find consensus on some aspect of HTTP State Tokens, we can apply those aspirations to cookies, driving incremental improvements to state management in the status quo.

Based on conversations at [HTTP-Workshop-2019] and elsewhere, I’d suggest that we have something like agreement on at least two principles:

1. HTTP requests should not carry state along with cross-site requests by default (see Section 8.2 of [RFC6265bis]).
2. HTTP requests should not carry state over non-secure channels (see Section 8.3 of [RFC6265bis], and [RFC7258]).

With those principles in mind, this document proposes two changes that seem possible to deploy in the near-term. User agents should:

1. Treat the lack of an explicit "SameSite" attribute as "SameSite=Lax". That is, the "Set-Cookie" value "key=value" will produce a cookie equivalent to "key=value; SameSite=Lax". Cookies that require cross-site delivery can explicitly opt-into such behavior by asserting "SameSite=None" when creating a cookie.

   This is spelled out in more detail in Section 3.1.

2. Require the "Secure" attribute to be set for any cookie which asserts "SameSite=None" (similar conceptually to the behavior for the "__Secure-" prefix). That is, the "Set-Cookie" value "key=value; SameSite=None; Secure" will be accepted, while "key=value; SameSite=None" will be rejected.

   This is spelled out in more detail in Section 3.2.

2. Conventions and Definitions

2.1. Conformance

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Syntax

This document adjusts some syntax from [RFC6265bis], and in doing so, relies upon the Augmented Backus-Naur Form (ABNF) notation of [RFC5234].

3. Monkey-Patches against RFC6265bis

3.1. "Lax" by Default

The processing algorithm in Section 5.3.7 of [RFC6265bis] treats the absence of a "SameSite" attribute in a "Set-Cookie" header as equivalent to the presence of "SameSite=None". Cookies are therefore available for cross-site delivery by default, and developers may opt-into more security by setting some other value explicitly. Ideally,
we'd invert that such that developers who accepted the risks of cross-site delivery (see Section 8.2 of [RFC6265bis]) could opt into them, while developers who didn’t make any explicit choice would be protected by default.

We could accomplish this goal by first altering the processing algorithm, replacing the current step 1:

1. Let "enforcement" be "None".

with the following two steps:

1. Let "enforcement" be "Lax".
2. If cookie-av’s attribute-value is a case-insensitive match for "None", set "enforcement" to "None".

And then by, altering step 13 of the cookie storage model (Section 5.4 of [RFC6265bis]) from:

13. If the cookie-attribute-list contains an attribute with an attribute-name of "SameSite", set the cookie’s same-site-flag to attribute-value (i.e. either "Strict", "Lax", or "None"). Otherwise, set the cookie’s same-site-flag to "None".

to:

13. If the cookie-attribute-list contains an attribute with an attribute-name of "SameSite", set the cookie’s same-site-flag to attribute-value. Otherwise, set the cookie’s same-site-flag to "Lax".

This would have the effect of mapping the default behavior in the absence of an explicit "SameSite" attribute, as well as the presence of any unknown "SameSite" value, to the "Lax" behavior, protecting developers by making cross-site delivery an explicit choice, as opposed to an implicit default.

3.2. Requiring "Secure" for "SameSite=None"

Cookies sent over plaintext HTTP are visible to anyone on the network. As section 8.3 of [RFC6265bis] points out, this visibility exposes substantial amounts of data to network attackers. We know, for example, that long-lived and stable cookies have enabled pervasive monitoring [RFC7258] in the past (see Google’s PREF cookie [pref-cookie]), and we know that a secure transport layer provides significant confidentiality protections against this kind of attack.
We can, to a reasonable extent, mitigate this threat by ensuring that cookies intended for cross-site delivery (and therefore likely to be more prevalent on the wire than cookies scoped down to same-site requests) require secure transport.

That is, we can require that any cookie which asserts "SameSite=None" must also assert the "Secure" attribute (Section 4.1.2.5 of [RFC6265bis]) by altering the storage model defined in Section 5.4 of [RFC6265bis], inserting the following step after the existing step 14:

15. If the cookie’s "same-site-flag" is "None", abort these steps and ignore the cookie entirely unless the cookie’s secure-only-flag is true.

This is conceptually similar to the requirements put into place for the "__Secure-" prefix (Section 4.1.3.1 of [RFC6265bis]).

4. Security and Privacy Considerations

4.1. CSRF

"SameSite" is a reasonably robust defense against some classes of cross-site request forgery attacks, as described in Section 8.8.1 of [RFC6265bis], but developers need to opt-into its protections in order for them to have any effect. That is, developers are vulnerable to CSRF attacks by default, and must do some work to shift themselves into a more defensible position.

The change proposed in Section 3.1 would invert that requirement, placing the burden on the small number of developers who are building services that require state in cross-site requests. Those developers would be empowered to opt-into the status quo’s less-secure model, while developers who don’t intend for their projects to be embedded in cross-site contexts are protected by default.

4.2. Secure Transport

As discussed in Section 8.3 of [RFC6265bis], cookies delivered over plaintext channels are exposed to intermediaries, and thereby enable pervasive monitoring [RFC7258]. The change proposed in Section 3.2 above would set secure transport as a baseline requirement for all stateful cross-site requests, thereby reducing the risk that these cookies can be cataloged or modified by network attackers.

Requiring secure transport for cookies intended for cross-site usage has the exciting secondary effect of increasing pressure on entities that produce embeddable content to migrate their products to HTTPS.
That has security benefits for those third-party products themselves, but also has the effect of removing the potential of mixed content ([mixed-content]) as a blocker to first-party migration to HTTPS.

Note that in the long term, it seems quite reasonable to take the additional step of requiring the "Secure" attribute for all cookies, regardless of their "SameSite" value. That would have more substantial impact on pervasive monitoring and network attackers generally. This document’s proposal limits itself to "SameSite=None" because that seems like a low-hanging, high-value change that’s deployable in the near term. User agents are encouraged to find additional subsets for which "Secure" can be required.

4.3. Tracking

The proposals in this document do not in themselves mitigate the privacy risks described in Section 7.1 of [RFC6265bis]. Entities who wish to use cookies to track user activity from cross-site contexts can continue to do so by setting cookies that declare themselves as "SameSite=None".

Requiring that explicit declaration, however, gives user agents the ability to easily distinguish cookies used for stateful cross-site requests from those with narrower scope. After the change proposed in Section 3.1, only those cookies that make an explicit "SameSite=None" declaration can be directly used for cross-site tracking. It may make sense for user agents to use that information to give users different controls for these cookies, or to apply different policies for expiration and delivery.

5. Implementation Considerations

5.1. Sequencing

The steps described in this document don’t need to be taken at the same time. It’s quite possible that it will be less disruptive to deploy "SameSite=Lax" as a default first, and then to require the "Secure" attribute for any explicitly "SameSite=None" cookie as a subsequent step.

User agents are encouraged to adopt these recommendations in whatever order they believe will lead to the widest, most expedient deployment.
5.2. Deployment

It’s possible that a middle-ground between "SameSite=Lax" and "SameSite=None" could be a better balance between doing what developers want by default, and mitigating CSRF by default. [I-D.west-cookie-samesite-firstparty] explores the possibility of integrating First-Party Sets [first-party-set] with the "SameSite" attribute in order to allow entities that shard themselves across multiple registrable domains to maintain stateful communication between them (to support single-sign on, for example).

It’s possible that user agents who support First-Party Sets could reduce the deployment overhead for developers, and increase the robustness of a site’s CSRF defense for cross-site-but-not-cross-party cookies by defaulting to something like that document’s "FirstPartyLax" instead of "Lax".

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. Normative References


7.2. Informative References

[first-party-set]

[HTTP-Workshop-2019]

[I-D.west-cookie-samesite-firstparty]

[I-D.west-http-state-tokens]
West, M., "HTTP State Tokens", draft-west-http-state-tokens-00 (work in progress), March 2019.

[mixed-content]

[pref-cookie]


Acknowledgments

Conversations with a number of folks at 2019’s HTTP Workshop helped me clarify my thinking around the incremental improvements we can make to cookies. In particular, Martin Thomson and Anne van Kesteren provided insightful feedback.

Author’s Address