HTTP
Internet-Draft
Intended status: Informational
Expires: 18 April 2026

A. Gill
Google LLC

15 October 2025

Origin-Bound Cookies
draft-amarjotgill-origin-bound-cookies-protocol-00

Abstract

This draft introduces Origin-Bound Cookies which updates [COOKIES] aiming to make cookies more secure by default through binding cookies by port and scheme.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://amarjotgill.github.io/origin-bound-cookies/draft-amarjotgill-origin-bound-cookies-protocol.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-amarjotgill-origin-bound-cookies-protocol/.

Source for this draft and an issue tracker can be found at https://github.com/amarjotgill/origin-bound-cookies.

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 18 April 2026.

Table of Contents

1. Introduction 2
1.1. Examples 3
2. Conventions and Definitions 3
2.1. Terminology 3
3. User Agent Requirements 3
3.1. Origin-Bound Behavior 3
3.1.1. Port Bound 4
3.1.2. Scheme Bound 4
3.2. Storage 5
3.3. Retrieve Cookies 5
3.4. Cookie Store Eviction 6
3.5. Requirements Specific to Non-Browser User Agents 7
3.5.1. The Cookie Header Field 7
4. Security Considerations 8
5. IANA Considerations 9
6. References 9
6.1. Normative References 9
6.2. Informative References 9
Acknowledgments 9
Author's Address 10

1. Introduction

Cookies are one of the few components of the web platform that are not scoped to the origin by default. This difference in scoping means that cookies have weak confidentiality (https://httpwg.org/http-extensions/draft-ietf-httpbis-layered-cookies.html#name-weak-confidentiality) compared with other storage APIs on the web platform.

1.1. Examples

1. https://somesite.com sets a simple cookie, secret=123456, which contains private information about a user. Information that an attacker wishes to learn. To do so the attacker man-in-the-middles the user, and then tricks them into visiting http://somesite.com (note the insecure scheme). When the user visits that page their browser will send the secret cookie and the attacker can see it.

2. Similarly, if the attacker has somehow compromised a service running on a different port on the same server, let’s say port 345, as https://somesite.com then they could trick the user into visiting https://somesite.com:345, the user’s browser will send the secret cookie, and once again the attacker can see it. Even more, through the same techniques, an attacker can also modify a user’s cookies, sending a Set-Cookie field instead of simply eavesdropping.

All of these examples are possible because cookies by default do not care about the scheme or port of their connection. As long as the host matches the cookie will be accessible.

This document proposes changes to [COOKIES] by binding cookies to port and scheme which will prevent against these attacks and greatly improve cookie security.

2. Conventions and Definitions

2.1. Terminology

An aliasing cookie refers to a cookie which shares the same cookie-name, cookie-value and sub domain with another cookie but differs via scheme or port.

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.

3. User Agent Requirements

3.1. Origin-Bound Behavior

3.1.1. Port Bound

First, alter Section 5.1.2 of COOKIES (https://httpwg.org/http-extensions/draft-ietf-httpbis-layered-cookies.html#name-cookie-struct) by adding port to the Cookie Struct, this would be necessary to keep track of the port the cookie was set on.

Below is the definition of the port attribute.

> A cookie’s port is either null or a 16-bit unsigned integer. It is initially null.

For port matching algorithms below will be updated to compare integers to ensure port values match. Pre-existing cookies with unspecified "port" will have a null value. This value will cause the cookie to be treated with legacy behavior (The cookie will not be bound to port).

Example:

The new behavior will behave as the following.

A cookie set by origin https://example.com will only ever be sent to https://example.com(:443). It will never be sent to a different port value such as https://example.com:8443.

3.1.2. Scheme Bound

First, alter Section 5.1.2 of COOKIES (https://httpwg.org/http-extensions/draft-ietf-httpbis-layered-cookies.html#name-cookie-struct) by adding scheme to the Cookie Struct, this would be necessary to keep track of the port the cookie was set on.

Below is the definition of the scheme attribute.

> A cookie’s scheme is null or a byte sequence. It is initially null.

Pre-existing cookies with unspecified "scheme" will have a null value. This value will cause the cookie to be treated with legacy behavior (The cookie will not be bound to scheme).

Example:

The new behavior will behave as the following.

A cookie set by origin https://example.com will only ever be sent to https://example.com. It will never be sent to a different scheme value such as http://example.com.

3.2. Storage

We update the storage model in Section 5.4.3 of COOKIES (https://httpwg.org/http-extensions/draft-ietf-httpbis-layered-cookies.html#name-store-a-cookie) to ensure that port and scheme are being considered when comparing with existing cookies.

Step number 18 of this section will need to be altered to:

> If the user agent's cookie store contains a cookie oldCookie whose name is cookie's name, host is host-equal to cookie's host, host-only is cookie's host-only, path is path-equal to cookie's path, and scheme is equal to cookie's scheme

A new substep should also be added to step number 18:

1. If cookie's host-only is set to true and port does not equal cookie's port then skip the remaining sub-steps.

Note the addition of checking port and scheme, this will prevent a cookie with differing port or scheme values from overwriting the oldCookie, instead this cookie would be stored as a separate cookie and the oldCookie will not be deleted. Also note if the domain attribute is set then we will ignore checking via port so it will overwite the oldCookie.

3.3. Retrieve Cookies

The Retrieve Cookies algorithm will need to be updated in Section 5.4.5 of COOKIES (https://httpwg.org/http-extensions/draft-ietf-httpbis-layered-cookies.html#name-retrieve-cookies).

Retrieve Cookies will now take a port (16-bit unsigned integer) and a scheme (string).

The following will need to be added to step 2:

1. cookies host only is true, cookie's port is not null, and port equals cookie's port.

2. cookie's scheme is not null and scheme equals cookie's scheme.

This will ensure that a cookie is only retrieved if the request origin is equal to the cookie’s origin, while minimizing disruption to users. Allowing current sites to continue working as-is, as old cookies are replaced with newer ones, the new cookies will be origin-bound.

3.4. Cookie Store Eviction

The last algorithm that will need to be updated is the Cookie Store Eviction algorithm outlined Section 5.2.2 of COOKIES (https://httpwg.org/http-extensions/draft-ietf-httpbis-layered-cookies.html#name-remove-excess-cookies-for-a).

First a new algorithm to sort eviction cookies will be added in Section 5.3 of COOKIES (https://httpwg.org/http-extensions/draft-ietf-httpbis-layered-cookies.html#name-subcomponent-algorithms)

To Sort Eviction Cookies, given a host, run the following steps.

1. Let insecureCookies be a list of cookies in the user agent’s cookie store whose host is host-equal to host and whose secure is false.

2. Let insecureDomainCookies be a list of all cookies in insecureCookies whose host-only is false.

3. Sort insecureDomainCookies by earliest last-access-time first.

4. Let insecureOriginCookies be a list of all cookies in insecureCookies whose host-only is true.

5. Sort insecureOriginCookies by earliest last-access-time first.

6. Let sortedInsecureCookies be the result of appending insecureOriginCookies to the end of insecureDomainCookies.

7. Let secureCookies be a list of cookies in the user agent’s cookie store whose host is host-equal to host and whose secure is true.

8. Let secureDomainCookies be a list of all cookies in secureCookies whose host-only is false.

9. Sort secureDomainCookies by earliest last-access-time first.

10. Let secureOriginCookies be a list of all cookies in secureCookies whose host-only is true.

11. Sort secureOriginCookies by earliest last-access-time first.

12. Let sortedSecureCookies be the result of appending secureOriginCookies to the end of secureDomainCookies.

13. Return sortedInsecureCookies and sortedSecureCookies.

Step 2 of Section 5.2.2 of COOKIES (https://httpwg.org/http-extensions/draft-ietf-httpbis-layered-cookies.html#name-remove-excess-cookies-for-a) need to be updated to replace:

| 1. Let insecureCookies be a list of references to all cookies in the user agent's cookie store whose host is host-equal to host and whose secure is false.
|
| 2. Sort insecureCookies by earliest last-access-time first.
|
| 3. Let secureCookies be a list of references to all cookies in the user agent's cookie store whose host is host-equal to host and whose secure is true.
|
| 4. Sort secureCookies by earliest last-access-time first.

With the following:

| 1. Let insecureCookies and secureCookies be the result of running Sort Eviction Cookies.

All remaining steps will stay the same. Updating these steps will ensure that cookies with the domain attribute set for each origin are deleted before any other cookie.

3.5. Requirements Specific to Non-Browser User Agents

3.5.1. The Cookie Header Field

The cookie header field outlined in Section 5.5.2 of COOKIES (https://httpwg.org/http-extensions/draft-ietf-httpbis-layered-cookies.html#name-the-cookie-header-field) will need to be altered.

It will be altered to the following:

| 1. Let isSecure be a boolean indicating whether request's URL's scheme is deemed secure, in an implementation-defined manner.
|
| 2. Let host be request's host.
|
| 3. Let path be request's URL's path.

```
|
|  4.  Let httpOnlyAllowed be true.
|
|  5.  Let sameSite be a string whose value is implementation-
|      defined, but has to be one of "strict-or-less", "lax-or-less",
|      "unset-or-less", or "none".
|
|  6.  Let port be request’s URL’s port.
|
|  7.  Let scheme be request’s URL’s scheme.
|
|  8.  Let cookies be the result of running Retrieve Cookies given
|      isSecure, host, path, httpOnlyAllowed, sameSite, port, and
|      scheme.
|
|  9.  Return the result of running Serialize Cookies given cookies.
```

Note the major change here is extracting the port and scheme from the request and running the new version of Retrieve Cookies based on that.

4. Security Considerations

Origin-Bound Cookies is considered net positive for security, the following would be mitigated by it.

Weak integrity: This occurs when an insecure site, controlled by a network attacker, sets a malicious cookie that is then sent to the secure version of that site. OBC binds cookies to their setting origin, preventing such malicious cookies from being accessed by a different origin.

Weak confidentiality: This refers to an attacker reading sensitive user data set by a secure site, which is unintentionally sent to an insecure site. OBC ensures that cookies are only accessible by the same origin that set them, preventing this type of data leakage.

One security concern is how to handle clients that require the use of cookies across origins, this is handled by the following, when a cookie has a valid Domain attribute specified (hereby called a “domain cookie”) that cookie has relaxed bindings. Namely, the cookie may be sent to:

1. any host which domain matches the Domain value (This is unchanged from legacy behavior)

2. any port value

Importantly a domain cookie is still bound to the scheme of its setting origin.

This behavior allows developers to opt-out of the stronger protections of an origin cookie which can help with compatibility for usages that need a particular cookie available across hosts and/or ports.

Origin binding cookies will obsolete the purpose of Secure but this proposal will not remove support for Secure. Removal for Secure could be a possible future consideration for the HTTP WG.

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. Normative References

[COOKIES] "Cookies HTTP State Management Mechanism", May 2025, <https://datatracker.ietf.org/doc/draft-ietf-httpbis-layered-cookies/>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

6.2. Informative References

[Cookies-Lack-Integrity] "Cookies Lack Integrity Real-World Implications", August 2015, <https://www.microsoft.com/en-us/research/wp-content/uploads/2017/01/sec15_cookies-lack-integrity-published.pdf>.

Acknowledgments

The editors would also like to thank the following individuals for feedback, insight, and implementation of this draft and its predecessors (in alphabetical order): Dylan Cutler, Johann Hofmann, Mike West, Steven Bingler,

Author’s Address

Amarjot Gill
Google LLC
Email: amarjotgill@google.com