websec
Internet-Draft
Intended status: Standards Track
Expires: May 30, 2011

A. Barth
Google, Inc.

November 26, 2010

The Web Origin Concept
draft-abarth-origin-09

Abstract

This document defines the concept of an "origin", which represents a web principal. Typically, user agents isolate content retrieved from different origins to prevent a malicious web site operator from interfering with the operation of benign web sites. In particular, this document defines how to compute an origin from a URI, how to serialize an origin to a string, and an HTTP header, named "Origin", for indicating which origin caused the user agent to issue a particular HTTP request.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on May 30, 2011.

Copyright Notice

Table of Contents

1. Introduction

User agents interact with content created by a large number of authors. Although many of those authors are well-meaning, some authors might be malicious. To the extent that user agents undertake actions based on content they process, user agent implementors might wish to restrict the ability of malicious authors to disrupt the confidentiality or integrity of other content or servers.

As an example, consider an HTTP user agent that renders HTML content retrieved from various servers. If the user agent executes scripts contained in those documents, the user agent implementor might wish to prevent scripts retrieved from a malicious server from reading documents stored on an honest server, which might, for example, be behind a firewall.

Traditionally, user agents have divided content according to its "origin". More specifically, user agents allow content retrieved from one origin to interact freely with other content retrieved from that origin, but user agents restrict how that content can interact with content from another origin.

This document does not describe the restrictions user agents ought to impose on cross-origin interaction. Instead, this document defines the origin concept itself in such a way that other specifications, such for HTTP [cite] or for HTML [cite], can refer to this document for a precise, common definition of the web origin concept.

Specifically, a user agent can compute the origin of a piece of content based on the URI from which the user agent retrieved the content. Given two origins computed in this way, the user agent can compare the origins to determine if they are "the same", which is useful for performing some security checks. Finally, given an origin, the user agent can serialize that origin into either an ASCII or a Unicode representation.

This document also defines one use of the ASCII serialization: the HTTP Origin header. An Origin header attached to an HTTP request contains the ASCII serializations of the origins that caused the user agent to issue the HTTP request. The Origin header has a number of uses, including for cross-origin resource sharing [cite].

2. Conventions

2.1. Conformance Criteria

The keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Requirements phrased in the imperative as part of algorithms (such as "strip any leading space characters" or "return false and abort these steps") are to be interpreted with the meaning of the key word ("MUST", "SHOULD", "MAY", etc) used in introducing the algorithm.

Conformance requirements phrased as algorithms or specific steps can be implemented in any manner, so long as the end result is equivalent. In particular, the algorithms defined in this specification are intended to be easy to understand and are not intended to be performant.

2.2. Syntax Notation

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

The following core rules are included by reference, as defined in [RFC5234], Appendix B.1: ALPHA (letters), CR (carriage return), CRLF (CR LF), CTL (controls), DIGIT (decimal 0-9), DQUOTE (double quote), HEXDIG (hexadecimal 0-9/A-F/a-f), LF (line feed), OCTET (any 8-bit sequence of data), SP (space), HTAB (horizontal tab), CHAR (any US-

ASCII character), VCHAR (any visible US-ASCII character), and WSP (whitespace).

The OWS (optional whitespace) rule is used where zero or more linear whitespace characters MAY appear:

```
OWS            = *( [ obs-fold ] WSP )
                   ; "optional" whitespace
obs-fold       = CRLF
```

OWS SHOULD either not be produced or be produced as a single SP character.

2.3. Terminology

The terms user agent, client, server, proxy, and origin server have the same meaning as in the HTTP/1.1 specification ([RFC2616], Section 1.3).

A globally unique identifier is a value which is different from all other previously existing values. For example, a sufficiently long random string is likely to be a globally unique identifier.

A idna-canonicalization host name is the string generated by the following algorithm:

1. Convert the host name to a sequence of NR-LDH labels (see Section 2.3.2.2 of [RFC5890]) and/or A-labels according to the appropriate IDNA specification [RFC5891] or [RFC3490] (see Section 10.1 of this specification)

2. Convert the labels to lower case.

3. Concatenate the labels, separating each label from the next with a %x2E (".") character.

3. Origin

An origin represents a web principal. Typically, user agents determine the origin of a piece of content from the URI from which they retrieved the content. In this section, we define how to compute an origin from a URI.

The origin of a URI is the value computed by the following algorithm:

1. If the URI does not use a server-based naming authority, or if the URI is not an absolute URI, then return a globally unique identifier.

2. Let uri-scheme be the scheme component of the URI, converted to lowercase.

3. If the implementation doesn't support the protocol given by uri-scheme, then return a globally unique identifier.

4. If uri-scheme is "file", the implementation MAY return an implementation-defined value.

    1. NOTE: Historically, user agents have granted content from the file scheme a tremendous number of privileges. However, granting all local files such wide privileges can lead to privilege escalation attacks. Some user agents have had success granting local files directory-based privileges, but this approach has not been widely adopted. Other user agent use a globally unique identifier each file URI, which is the most secure option.

5. Let uri-host be the idna-canonicalization of the host component of the URI.

6. If there is no port component of the URI:

    1. Let uri-port be the default port for the protocol given by uri-scheme.

    Otherwise:

    2. Let uri-port be the port component of the URI.

7. Return the triple (uri-scheme, uri-host, uri-port).

Implementations MAY define other types of origins in addition to the scheme/host/port triple type defined above. For example, an implementation might define an origin based on a public key or an

implementation might append addition "sandbox" bits to a scheme/host/port triple.

4. Comparing Origins

To origins are "the same" if, and only if, they are identical. In particular:

- If the two origins are scheme/host/port triple, the two origins are the same if, and only if, they have identical schemes, hosts, and ports.

- An origin that is globally unique identifier cannot be the same as an origin that is a scheme/host/port triple.

- Two origins that are globally unique identifiers cannot be the same if they were created at different times, even if they were created for the same URI.

Two URIs are the same-origin if their origins are the same.

> NOTE: A URI is not necessarily same-origin with itself. For example, a data URI is not same-origin with itself because data URIs do not use a server-based naming authority and therefore have globally unique identifiers as origins.

5. Serializing Origins

This section defines how to serialize an origin to a unicode string and to an ASCII string.

5.1. Unicode Serialization of an Origin

The unicode-serialization of an origin is the value returned by the following algorithm:

1. If the origin is not a scheme/host/port triple, then return the string

   null

   (i.e., the code point sequence U+006E, U+0075, U+006C, U+006C) and abort these steps.

2. Otherwise, let result be the scheme part of the origin triple.

3. Append the string "://" to result.

4. Append the [TODO: IDNA ToUnicode] algorithm to each component of the host part of the origin triple, and append the results of each component, in the same order, separated by U+002E FULL STOP code points (".") to result.

5. If the port part of the origin triple is different than the default port for the protocol given by the scheme part of the origin triple:

   1. Append a U+003A COLON code point (":") and the given port, in base ten, to result.

6. Return result.

[TODO: Check that we handle IPv6 literals correctly.]

5.2. ASCII Serialization of an Origin

The ascii-serialization of an origin is the value returned by the following algorithm:

1. If the origin is not a scheme/host/port triple, then return the string

null

(i.e., the code point sequence U+006E, U+0075, U+006C, U+006C) and abort these steps.

2. Otherwise, let result be the scheme part of the origin triple.

3. Append the string "://" to result.

4. Append the host port of the origin triple to result.

5. If the port part of the origin triple is different than the default port for the protocol given by the scheme part of the origin triple:

    1. Append a U+003A COLON code points (":") and the given port, in base ten, to result.

6. Return result.

6. The HTTP Origin header

This section defines the HTTP Origin header.

6.1. Syntax

The Origin header has the following syntax:

```
origin              = "Origin:" OWS origin-list-or-null OWS
origin-list-or-null = "null" / origin-list
origin-list         = serialized-origin *( SP serialized-origin )
serialized-origin   = scheme "://" host [ ":" port ]
                    ; <scheme>, <host>, <port> productions from RFC3986
```

6.2. Semantics

When included in an HTTP request, the Origin header indicates the origin(s) that caused the user agent to issue the request.

For example, consider a user agent that executes scripts on behalf of origins. If one of those scripts causes the user agent to issue an HTTP request, the user agent might wish to use the Origin header to inform the server that the request was issued by the script.

In some cases, a number of origins contribute to causing the user agents to issue an HTTP request. In those cases, the user agent can list all the origins in the Origin header. For example, if the HTTP request was initially issued by one origin but then later redirected by another origin, the user agent might wish to inform the server that two origins were involved in causing the user agent to issue the

request.

6.3. User Agent Requirements

The user agent MAY include an Origin header in any HTTP request.

The user agent MUST NOT include more than one Origin header field in any HTTP request.

Whenever a user agent issues an HTTP request from a "privacy-sensitive" context, the user agent MUST send the value "null" in the Origin header.

> NOTE: This document does not define the notion of a privacy-sensitive context. Applications that generate HTTP requests can designate contexts as privacy-sensitive to impose restrictions on

> how user agents generate Origin headers.

When generating an Origin header, the user agent MUST meet the following requirements:

- Each of the serialized-origin productions in the grammar MUST be the ascii-serialization of an origin.
- No two consecutive serialized-origin productions in the grammar can be identical. In particular, if the user agent would generate two consecutive serialized-origins, the user agent MUST NOT generate the second one.

If the user agent issued an HTTP request current-request because the user agent received 3xx Status Code response to another HTTP request previous-request for URI previous-uri:

- The HTTP request current-request MUST include an Origin header.
- The value of the Origin header MUST be either:
  - The string "null" (i.e., the byte sequence %x6E, %x75, %x6C, %x6C).
  - The value of the Origin header in the previous-request. The

user agent MUST NOT choose this option if the ascii-serialization of previous-uri is not identical to the last serialized-origin in the Origin header of the previous request.

* The value of the Origin header in previous header extended with a space and the ascii-serialization of the origin of previous-uri. The user agent MUST NOT choose this option if the ascii-serialization of the origin of previous-uri is "null".

The user agent SHOULD include the Origin header in an HTTP request if the user agent issues the HTTP request on behalf of an origin (e.g., not by the user operating a trusted user interface surface). In this case, the user agent SHOULD set the value of the Origin header to the ascii-serialization of that origin.

NOTE: This behavior differs from the usual user agent behavior for the HTTP Referer header, which user agents often suppress when an origin with an "https" scheme issues a request for a URI with an "http" scheme.

7. Privacy Considerations

[TODO: Privacy considerations.]

8. Security Considerations

[TODO: Security considerations.]

9. IANA Considerations

[TODO: Register the Origin header.]

10. Implementation Considerations

10.1. IDNA dependency and migration

IDNA2008 [RFC5890] supersedes IDNA2003 [RFC3490] but is not backwards-compatible. For this reason, there will be a transition period (possibly of a number of years). User agents SHOULD implement IDNA2008 [RFC5890] and MAY implement [Unicode Technical Standard #46 <http://unicode.org/reports/tr46/>] in order to facilitate a smoother IDNA transition. If a user agent does not implement IDNA2008, the user agent MUST implement IDNA2003 [RFC3490].

11. Normative References

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

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

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

See Section 10.1 for an explanation why the normative reference to an obsoleted specification is needed.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

Appendix A. Acknowledgements

Author's Address

Adam Barth
Google, Inc.

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