Security/Origin

From MozillaWiki
< Security

The HTTP Request header Sec-From, has changed to "Origin" because it is interoperable with the Origin header in Cross-Origin Resource Sharing (http://www.w3.org/TR/2009/WD-cors-20090317/).

Contents

- 1 Origin header proposal for CSRF and clickjacking mitigation
  - 1.1 Design Path
    - 1.1.1 Advantage of more than one bit of data
    - 1.1.2 Selection of "null" token
    - 1.1.3 Why not include a frame list?
    - 1.1.4 Firewall-based Origin header scrubbing
- 2 Origin header format
  - 2.1 When Origin is served (and when it is "null")
    - 2.1.1 Privacy-Sensitive Contexts
- 3 Implementation
  - 3.1 Sample Web Application Use

Origin header proposal for CSRF and clickjacking mitigation

This page contains collected thoughts generated in discussion and deep thinking about implementing some type of Origin-like header (http://people.mozilla.org/~bsterne/content-security-policy/origin-header-proposal.html).

The Origin header is considered helpful against JSON data theft and CSRF attacks. The information provided by Origin--a bit of contextual request-creation information--should provide hints to web servers about trustworthiness of requests in all three of these situations.

JSON data theft
: Data served via JSON (and imported using a script tag) can be stolen if the origin of a JSON request is not authenticated. If the origin of a script request were provided, web servers could decide whether or not to serve JSON data.

CSRF
: Cross-site request forgeries are often GET requests assembled and sent through the use of an automatic load (like an img or script tag). In many scenarios, like the two mentioned, state changing transactions should not be allowed. In other scenarios, like form submissions, state-changing

transactions should be accepted but should be authenticated so the server knows what site generated the request.

Generally, the Origin header aims to provide a bit of context with HTTP requests so that servers may make educated decisions on whether or not to serve data, accept request data for state-changing transactions, or continue with a persistent session. This is accomplished by specifying a list of sites that indirectly caused a request (the redirect chain) and the immediate "Origin" of a request, or the entity that most recently caused the request to happen. This Origin may be a host name or the string "null" in the cases where a request may have been falsely or deceptively generated.

Design Path

Much discussion and debate went on when considering how to design this feature. This section attempts to describe some of the decisions made and the reasoning behind them.

Advantage of more than one bit of data

One similar approach to solving CSRF is to send a "Same-Origin" header with requests, setting its value to "YES" if the referrer is of the same origin as the requested content and "NO" otherwise. While this one-bit approach can be effective against CSRF, it doesn't seem robust enough for deployment.

Some sites post data cross-site when they own multiple domains. Additionally, a state-changing request might not be actually *intended*: an open redirect might be exploited, thus spoofing the content submission. Finally, a small amount of extra complexity (and request size) provides flexibility so this feature can be used not only as CSRF protection but also in other scenarios where knowing the origin of a request (and its redirect chain) might be handy.

Selection of "null" token

In some scenarios, the string "null" is sent in lieu of origin information. This is done to indicate that the cause of the request is not trustworthy, even though it may come from the same origin. Certain requests are not generally useful as state-changing triggers (like requests for stylesheets, images or window navigation) and probably should not be trusted even if sent same-origin.

We chose the string "null" because of its neutral connotations. The Origin header must *always* be sent to indicate support from the User Agent; "null" seems to indicate that, though Origin is supported, the User Agent didn't think the request should be trusted to trigger state change. Other tokens could be used that more aptly describe the meaning of an "empty but present" header value: "redacted", "private" or "unsafe". "null" is fairly standard across HTTP, though, and for now we have opted to use it.

Why not include a frame list?

There were a number of factors that caused this proposal to change from a model that helps prevent clickjacking to what is proposed here. An earlier proposal suggested providing the chain of frames as well as the origin of the request. However, the **chain of requests** (i.e., redirects and referrer) that cause a document to load and the **layout context** in which a document will be rendered seem to be useful in different cases; the data points for "how you get something" versus "what you do with it" solve pretty orthogonal problems and we don't want to add complexity to Origin if it means a significant delay in adoption.

Knowing the frame chain is indeed useful, but it seems helpful when solving problems different from those originally targeted by a feature like Origin. For example, clickjacking prevention (one of the uses for the layout context or frame tree) seems most appropriate on the client side where the framing takes place, so it seems to me that sending this data to the server might not be a best course of action.

There are other features in the works that will hopefully fill the need for clickjacking prevention (CSP for example).

Firewall-based Origin header scrubbing

TODO: Explain why and how admins of intranets may want to manipulate requests (setting to "null" instead of erasing) when forwarding outbound requests. Explain why removing is bad.

Origin header format

In order to provide enough information that makes this Origin header useful for more server-side protections (other than just CSRF), the origin of a request may be sent (or the string "null") as well as a list of any redirects that led to the final request.

The Origin header is described in an internet draft by Adam Barth, Collin Jackson and Ian Hickson (http://tools.ietf.org/html/draft-abarth-origin). The general format of the Origin header will be:

```
```

An   is a combination of scheme, host and port. Unlike HTTP Referer, no path data or query string will be provided in the origin. The first origin value will be the initial source of the request, and any remaining values will be origins of any redirects that changed the target of the request.

When Origin is served (and when it is "null")

This table explains when origin values for Origin are served and when "null" is served as its value instead.

API	Send origins? (no means "null")	Workaround to Get Origin Value	Notes
Anchor tag	NO	Convert to GET FORM	Many sites allow users to post links, so we don't want to send Origin with links
Window navigation	NO	Convert to GET FORM	Refers to anchor.href, window.location, window.open, ...? These are often used as equivalents to user generated links, making them susceptible to CSRF.
IMG	NO		Images are easy to inject into a site.
iframe, embed, applet	YES	None	Embedding information useful to address clickjacking. *If ancestor tree is more than 1 deep, send "null"
Form (GET and POST)	YES	Rewrite as simple GET upon submission	Remote sites need to authenticate source of request
<SCRIPT>	YES	None	Additional mitigation for JSON data theft

stylesheets	NO	N/A	CSS is generally session-dependent and requests for such should not be state modifying.
dependent loads from stylesheets	NO	N/A	These are calls within CSS and are mainly images.
Redirects	YES	None	Before honoring redirect, append current origin to end of Origin value (unless the last origin in the header is equal to the current origin, then do not modify its value). Set entire header value to "null" if redirect crosses FQDN boundaries or if initial value is "null".
XHR	YES	None	

Privacy-Sensitive Contexts

To elaborate on the table above, in the [1] (http://tools.ietf.org/html/draft-abarth-origin-05), it is stated that "null" must be sent as the value of Origin instead of origin data when the request is initiated from a privacy-sensitive context. Following are a list of privacy sensitive contexts:

Anchor Tag/hyperlink click
: hyperlinks are common ways to jump from one site to another without trust. They should not be used to initiate state-changing procedures.

Window navigation
: changing the location of a window is a common way to mimic a hyperlink click.

Image load (<img> tag)
: third-party images are commonly embedded across origins and can be used as "web bugs"

Stylesheet
: third-party stylesheets should not initiate state changing requests.

Dependent load in stylesheet
: usually an image, protected for reasons like the image load mentioned above.

Remaining contexts are not privacy sensitive and origin information should be transmitted in the Origin header.

Implementation

... (discuss confusion with CORS Origin, Referer, etc. Programmatic details about serving the origins)

Sample Web Application Use

... (show examples of how to use Origin to protect content from CSRF/JSON-theft)