Fenced Frame Reporter Header Changes

Liam Brady May 24, 2024

Reviewer	Status	Date
Shivani Sharma	lgtm with nits	10/21/2024
Alex Moshchuk	lgtm	Oct 22, 2024
Christian Dullweber	lgtm	Oct 24, 2024

Objective

Include information about the reportEvent() initiator that will be needed to help support cross-origin reportEvent() beacon support as well as cross-origin automatic beacon support which was introduced in this I2S.

Background

This document refers to the following variants of beacon reporting:

- reportEvent() with destination enum
- reportEvent() with destination URL
- Automatic beacons

We plan on introducing cross-origin support for reportEvent() beacons. When this happens, event-level reports will be able to be sent from origins other than the root ad frame's origin. However, a server currently can't tell where an event-level report originates, since the referer is unset and the Origin header is set to the worklet origin. Having this capability will be useful for the server to make a more informed decision on how to process a beacon originating from a cross-origin source. However, we need to be careful to not give a server any extra information it would otherwise not be able to obtain.

Design

"Referer" Header

The "Referer" header, which is currently unset, will be populated with the origin of the frame that triggered the event-level report. For each of the beacon types specifically:

- reportEvent() with destination enum: The origin of the frame that invoked reportEvent()
- reportEvent() with destination URL: The origin of the frame that invoked reportEvent().

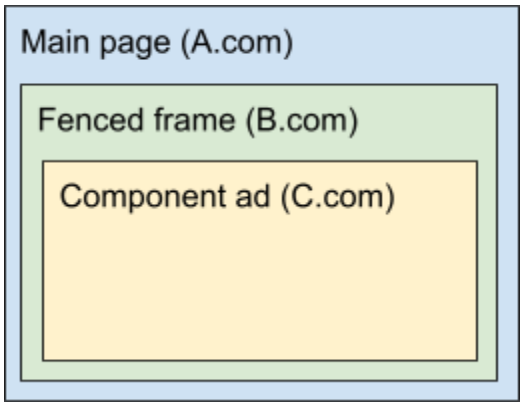
- Automatic beacons: The origin of the frame that initiated the navigation resulting in the beacon being sent out.
- Automatic beacons from component ad frames: The origin of the root ad frame of the component ad frame that initiated the navigation. This is done because information about the component ad's origin is not currently available to beacon reporting servers and should not be introduced.

Referrer Policy

The referrer policy will be set to the referrer policy of the frame that initiated the beacon request. For automatic beacons from component ads, the referrer policy will be set to the referrer policy of the ad frame root to not expose the ad component's referrer policy to reporting endpoints. With this, the referrer that is sent out with the beacon will be, at its least restrictive, the origin of the relevant frame. The referrer will never be the full URL, as that is information we don't want to leak.

Example

For example, consider the following page setup:



- For all beacons triggered from B.com, the referrer header will be set to B.com and the referrer policy will be set to B.com's referrer policy.
- For all reportEvent() beacons triggered from C.com, the referrer header will be set to C.com and the referrer policy will be set to C.com's referrer policy.
- For automatic beacons triggered from C.com, the referrer header will be set to B.com and the referrer policy will be set to B.com's referrer policy.
- A.com is not in a fenced frame tree, so it can't trigger reporting beacons and this does not apply.

Security Considerations

All of these changes are handled entirely in the browser, so no new information is being exposed to any renderer process.

The "Referer" header is not used for security checks the way that the "Origin" header is, so setting these values will not affect security.

Privacy Considerations

The "Referer" header is also introducing new information to the beacon, namely the origin of the frame that triggered the event. Although note that this information could also be included today in the reportEvent parameters. However, by also including the referrer policy, the frame sending out the request will have control over how much information is sent to reporting endpoints. To preserve privacy, we will never include the full path in the header; at most, the origin will be included in the referrer header.

The one exception is for component ad events, which is already not allowed to set their own data for privacy reasons. Setting the origin to the component ad frame's origin **would** introduce new information, so we guard against that by setting the origin to the origin of the root ad frame, as that frame would be the one setting the automatic beacon event on the ad component's behalf. This same logic applies to the referrer policy, as ad components will use the ad frame root's referrer policy in beacons instead of its own frame's policy.

What about the "Origin" header?

As part of these changes, we also looked at how the "Origin" header was being set. Right now, the "Origin" header is set to the worklet's origin for reportEvent() with destination enum and automatic beacon events. This is done because the beacon destination is determined by the worklet. The "Origin" header is set to the origin of the frame that invoked reportEvent() for destination URL events, since the URL is determined by the frame itself.

We considered standardizing this so that all beacon types would have the worklet origin as its header. This makes sense for the cases where the worklet is providing the reporting URL, but it does not make sense when the fenced frame itself is providing the reporting URL. If we did make this change, then a fenced frame could send a reportEvent() with destination URL event to the worklet's origin without the worklet's knowledge, and the server would think that the request came from the worklet origin. This is something we want to avoid, so the "Origin" header behavior will remain unchanged.