Always Send Automatic Beacons To Registered URLs

Liam Brady / *Sep 13, 2023*

Background

Note that this feature is a variant of fenced frame event-level reporting, and therefore will only be available temporarily for the next several years.

With automatic beacons, we currently only send beacons to destinations explicitly mentioned when calling `setReportEventDataForAutomaticBeacons()`. While this allows more control over who gets automatic beacons, it does mean that destinations that have asked to receive automatic beacons will not receive them unless explicitly opted in through the `setReportEventDataForAutomaticBeacons()` call. This can get confusing, since that function call implies that it's gating the event data, and not the entire event itself.

After an internal discussion, we decided that any destination that asked to receive a beacon for a top-level navigation should be allowed to get one, since they have a vested interest in knowing if a purchase happened based on an ad click. The frame that calls `setReportEventDataForAutomaticBeacons()` should not be allowed to interfere with whether a destination gets that information. However, that frame should still have control over what destinations get the data it's setting for the automatic beacon.

From kleber@:

> "The reason this makes sense in the ad tech context is that in today's world, the SSP actually proxies the content from the DSP — and so they have the ability to literally modify the ad creative. In the Protected Audience case we have so far tried to clean this up a bit, and have the actual HTML of the ad itself come from just a single party, the DSP. But SSPs are losing necessary information that they used to get because they actually were the first party of the rendered ad iframe before."

[See Turtledove issue for more information.]

The Change

Any worklet that registers an automatic beacon will now always have an automatic beacon sent to its destination when triggered. However, it will not receive any data as part of the automatic beacon unless explicitly added as a destination when calling `setReportEventDataForAutomaticBeacons()`. If no data is specified, or if the data is not meant to go to a certain destination, that destination will receive a POST request with an empty body.

Interaction With “once”

The “once” parameter in `setReportEventDataForAutomaticBeacons()` is meant to only have automatic beacons be sent once. After a beacon is sent, the data is cleared out and subsequent top-level navigations will not result in a beacon being sent (since there is no more data to send). With the new changes, the “once” behavior will need to change. There are a couple ways we can do this:

1: “Once” only sends data once

Automatic beacons will be sent to every registered URL, but the data will only be sent as part of the beacon one time. After that, automatic beacons with no data will be sent out to every destination.

This will be a breaking change to how “once” works, and will most likely break current workflows. If we go this route, this change must be communicated to external parties that are currently using “once”. We will most likely need to reach out directly to them to give them a heads up.

2: “Once” stops beacons from sending more than once

“Once” will act as a way to halt automatic beacons from sending. If the automatic beacon data has “once” set to true, after a beacon is sent, no further beacons will send out until otherwise specified by a subsequent `setReportEventDataForAutomaticBeacons()` call.

This will probably be very confusing, as this will essentially disable automatic beacons in a frame after one event. This should probably not be done.

Decision

While option 1 is a breaking change, it makes the most sense in the scope of how automatic beacons currently work. It also will match how `setReportEventDataForAutomaticBeacons()` is now only controlling whether the data is sent, rather than whether the whole beacon is sent. We will need to make sure this change is communicated and that partners are given enough of a heads up to update their experiments.

Privacy Implications

This change allows the auction participants to know there was a user-activated top-level navigation from the creative. This is gated behind user activation and does not allow arbitrary data from the FF site to be joined with the data from the worklet site either via the event name (since that's fixed as reserved.top_navigation) or event data (no event data is sent unless opted in by the FF site).
Additionally, the site that registers to get the beacon, will need to enroll and attest for Protected Audience API usage.

Security Implications

This change alters what information/when that information is sent from a frame. Before, the fenced frame had the final say in whether an automatic beacon would be sent out. Now, it only has the power to send or not send additional data to destinations that receive automatic beacons. If a buyer or seller involved in an ad auction wants to know if a click resulted in a top-level navigation, it will now get that information.

While this is only 1 bit being sent out, it's still information going from a frame to cross-origin servers without the frame's consent. That being said, the information being sent out is specifically whether a click from a user resulted in a top-level navigation committing, and the recipients of that information are being determined by parties directly involved in the creation of that frame (i.e. the buyer and seller of an ad auction).

Ad platforms need this behavior so they can still get reporting data from potentially uncooperative ad frames. This capability does not currently exist for them in the web platform, so, feature-wise, this would be an improvement for them.

This would have the potential to become problematic if something that's not an ad platform ran an ad auction and embedded a frame that's not intended in being an ad, giving the platform information on whether a click happened in the frame that led to a top-level navigation (no additional click information is sent with this event unless explicitly specified by the frame using `setReportEventDataForAutomaticBeacons()`). But note that the entity that invokes `registerAdBeacon()` for event-level reports are enrolled/attested under the Privacy Sandbox enrollment/attestation framework. Therefore, we have transparency into which business entity is using this information, and we know that the business entity attests to not use this information for cross-site user tracking. Since event-level reporting will be removed in several years, this API surface will only ever exist subject to attestation checks.

Security-conscious alternatives

The main difference between this behavior and the current behavior from a security standpoint is the lack of consent from the frame on whether an automatic beacon is sent out to registered destinations. It isn't completely clear whether the bit of information that's going out - "user clicked leading to a top-level navigation" requires an opt-in but if that ends up being a sticking point, the behavior can be modified so that a frame needs to explicitly opt in for an automatic beacon to send out to all destinations. This will remove any potential security concerns, but will re-introduce the problem of a potentially uncooperative ad frame refusing to allow an automatic beacon to be sent out.

This can be done in a few ways:

Use setReportEventDataForAutomaticBeacons as an opt-in for all destinations.

This will have a call to `setReportEventDataForAutomaticBeacons()` be the source of explicit consent for an ad frame. If this function is called, all registered destinations will get a beacon, but the "destination" field will determine what destinations get the beacon data. If this function is not called, no beacon will be sent out.

This will allow for a frame to decide if an automatic beacon is sent out, but it will not have granular control over who gets a beacon; it will only have granular control over who gets the beacon data. This also might get confusing for a developer, since this function will now be handling both setting data and determining if a beacon is sent out without data.

Add a new opt-in function.

This would introduce a new function, `enableAutomaticBeacon(event_type)`, that would allow for an automatic beacon to be sent, even if no event data is called with `setReportEventDataForAutomaticBeacons()`. If `enableAutomaticBeacon` is not called but `setReportEventDataForAutomaticBeacons` is, then this will only have automatic beacons send out to the destinations specified in `setReportEventDataForAutomaticBeacons`. This would allow for more explicit opt-in, but would increase API complexity and would still not allow for granular control over who gets a beacon without data.

Add a new opt-in function with granular control.

Similar to the previous solution, this would add a new parameter so it would have the interface `enableAutomaticBeacon(event_type, destinations)`. This would allow for granular control over who gets a beacon without data, but a non-cooperative ad frame could simply refuse to have automatic beacons be sent out, reducing the efficacy of the reporting system.

Use Supports-Loading-Mode: fenced-frame header as an opt-in

This is something we already have in place. As part of using this header, it will be understood that a top-level navigation will result in any interested seller/buyer parties that called registerAdBeacon() to learn that the navigation happened. While this works for fenced frames, URN iframes do not require this header, so the problem will persist in that use case. Since URN iframes are being used for the time being, this solution can't be used.

Add a new permissions policy

This would add a new permissions policy that allows data to be sent out in an automatic beacon. The main issue with this is that an ancestor frame can simply disable the permissions policy, and now the ad frame has no way to allow automatic beacons.

Recommendation

~~The security considerations mentioned in the guide here that require an opt-in is mostly information about cross-origin resources. Since that's not the case here (no cross-site resources information) and the fact that the entities receiving this information need to be enrolled/attested, it seems ok to not have an opt-in. But we would like to confirm this understanding with security reviewers.~~

Since explicit opt-in is needed to not introduce any security issues, one of the security-conscious alternatives should be chosen. Using `setReportEventDataForAutomaticBeacons()` as an opt-in for all destinations will involve the least amount of change to the API shape, and will not require more changes than is currently required from ad frames.