SPURFOWL

Sandboxed Private User Reporting Functions Operating Within Limits

Motivation

- Be able to do complicated reporting in an environment where user activity data stays in the browser.

Examples:

- Multi-touch attribution models
- Site metrics sliced by organic/non-organic traffic

SPURFOWL requirement: Secure aggregation framework

SPURFOWL assumes the existence of a secure aggregation framework, that can compute aggregate reports in a private way.

In particular, SPURFOWL is written assuming Chrome's "Multi-Browser Aggregation Service Explainer".

https://github.com/WICG/conversion-measurement-api/blob/master/SERVICE.md

SPURFOWL proposal

Add two mechanisms to the browser:

1) The “trail store”. The trail store stores impression events, click events, any first-party events. The trail store stores JavaScript objects and is write-only except in:
2) Sandboxed JavaScript functions. These run without access to network and cannot make persistent changes to the browser. They can see inside the full trail store.

Example: shoes.com user journey

Step 1) new user appears on shoes.com

They visit some pages. Every time they do that,
some JavaScript on shoes.com uses a web API call to store an event to the browser:

```
// top-level origin = shoes.com
window.trailStore.push('shoes.com',
                       { 'path': '/some-path',
                         'segment': 'some-segment' })
```

Step 2) User navigates away, gets served ads

At impression render, we insert an impression event to the browser.

```
// top-level origin = some-other-site.com
window.trailStore.push('shoes.com',
                       { 'ad_id': 12345,
                         'bid_price': 12.5,
                         ...  })
```

Step 3) Reporting code is run

- Browser fetches https://shoes.com/.well-known/reporting.js
- reporting.js implements whatever reporting we want to do

```
// reporting.js
function trailMap(trail_store) {
    var count = 0;
    for (var i = 0; i < trail_store.length; ++i) {
        count += 1;
    }
    return { 'number-of-events': {'value': count} }
}
```

Step 4) report is sent out

24-48 hours later at a random time, the report is sent to (encrypted):

https://shoes.com/.well-known/report

shoes.com will have to send it to secure aggregation servers (along with other similar reports) to get the final, aggregate values.

Handwaved details

- In practice, shoes.com will likely contract some other service to do all this. But technically they wouldn’t have to.
- window.trailStore.push(...) should not honor requests to add events for just any site, follow same-origin policy by default.
- How to deal with users who hack their reporting code and insert garbage? (Can use Prio to mitigate using zero-knowledge proofs)
- All reporting code is public

Summary

Add two mechanisms to the browser:

1) The “trail store”. The trail store stores impression events, click events, any first-party events. The trail store stores JavaScript objects and is write-only except in:
2) Sandboxed JavaScript functions. These run without access to network and cannot make persistent changes to the browser. They can see inside the full trail store.

These two mechanisms together allow computing complicated metrics and allows innovation to happen in reporting, in a privacy-first ad world.