Browser measurement helper service setup

This document covers the full flow from services registering their availability with the browser vendor to the reports leaving the browser client with the appropriate encryption.

To setup a helper service you need to:

A. Apply to be a measurement helper service for use in Edge
B. Provide helper service key commitments
C. Follow API and data flow

Helper service registration with the browser

There are two phases we expect with the approval process of helper services:

1. An evaluation phase in which helper service owners provide feedback about the protocol and API shape and enable other ad industry participants to evaluate the effectiveness, performance, and other relevant characteristics of the end-to-end flow.

   Since this phase will happen before broader default 3p cookie lockdowns happen, the auditing and compliance process will be reduced/streamlined because the additive risk is close to zero.

2. After new privacy enforcements are in place, e.g. general 3p cookie support being removed, the measurement APIs will become a critical piece of infrastructure to get the right signals in a privacy-preserving way. Once we enter this phase, we expect the general characteristics of the system to be well-understood and will want to message to consumers that there's a robust set of privacy protections for flows involving helper services. As a result, the bar for approval and ongoing work to maintain that approval will require more effort on the part of the helper service.

A. Apply to be a measurement helper service for use in Edge

To apply to be a measurement helper service for use in Edge, the helper's operator should file an issue on the Microsoft PARAKEET repo: https://github.com/microsoft/PARAKEET and include the following:

1. Helper service operator name: a human-readable name for the operator of the service.
2. The origin used by the helper service (scheme, host, and port). The scheme must be HTTPS.
3. One or more contact email addresses (ideally both an individuals' and an alias that can be updated by the operator to forward to the current contacts for the service).
4. [During evaluation phase]: a description of expected clients for the helper service and which other helper services, if any, they expect to typically be used with their service.
5. [Post-evaluation and lockdown phase]: A declaration from the operator confirming that:
   a. The owning entity (and/or its parent entities) has no ownership or control over any other registered measurement helper services.

b. If a future ownership or contractual change takes place, that notice is provided to Microsoft prior to the change being made with a clear indication of which measurement helper service should no longer be trusted.

During the evaluation phase, we expect to approve all applications. For the final shipping solution, services will need to be re-approved on a recurring basis (e.g. 6 months), with reasonable notice for requirement changes.

B. Provide helper service key commitments

The origin to be used by the helper service MUST provide a list of measurement helper service key commitments via the well known path */.well-known/aggregation-helper/keys.json*. For example, if your helper service's origin is https://measurement-helper.company.example, it must host key commitments at *https://measurement-helper.company.example/.well-known/aggregation-helper/keys.json*.

Key commitment format

```
{
  <protocol_version>: {
    "protocol_version": <protocol version as a string, e.g. "MeasurementHelperV1">,
    "id": <key commitment identifier, as a monotonically increasing integer>
    "key": { "Y": <base64-encoded MeasurementHelperPublicKey>,
             "expiry": <key expiry, encoded as a string representation of an integer
timestamp in microseconds since the Unix epoch> },
    }
  },
  ...
}
```

Process for browser to receive helper key commitments

The browser vendor will take its approved list of measurement helper provider origins and, on a regular cadence (e.g. once per day and/or some reasonable period before the key expiry), fetch the key commitments.

It will in turn package the key commitments into an updateable component to be delivered to the browser on a regular basis. The expected SLA is less than 48 hours from when an update is made to when it's delivered to the vast majority of clients.

All browser clients will regularly receive updates and have a list of approved helper service origins.

C. Follow API and data flow

Specifying the helpers to be used

As seen in the Aggregate Reports explainer, when an aggregation report is being made via script, it must specify the two aggregation services' origins that it will use.

If we assume all browsers have a shared list of trusted helpers, this design may be generally sufficient.

Either way, since helper services may be revoked due to acquisitions, non-compliance, or discontinuation of business, there needs to be mechanisms for users of the services to discover their inability to continue using the helper.

There are at least a few non-mutually exclusive options here:

1. Since aggregate reports will typically be done from a Fenced Frame context, throwing an exception at runtime is unlikely to be useful for discovering the issue. Having the browser schedule Reporting API reports on behalf of the origin would be reasonable.
2. The aggregate attribution reports contain the helpers' origins. If one of more of the requested helpers was unsupported, it could be given context about the issue in the aggregate report.
3. An API could be created to allow querying the list of supported helper origins.

In all of the above examples, we should assume that an adversary could encode sensitive information directly into a fake helper origin (e.g. one that maps to a generated unique ID for the user such as *user123.example*) which we cannot passively pass along as part of an error report. As a result, the information about which helper was not supported should be either omitted or only included if it's in a known list of non-user-specific reporting helper service origins (e.g. the browser could retain a list of previously supported origins and/or origins it knows are supported in other browsers and include those in the clear).

Since the set of supported helpers may be disjoint between browsers, we propose both an API that returns a list of supported helper origins for the reporting script to maximize the chances it will be able to identify two helpers it will be able to leverage.

Browser client generating the aggregate attribution report

Using the context provided by the site, the report payload will be split according to the requirements for the specific reporting flow. The payloads will be encrypted via HPKE via the matching public key provided to the client earlier.

The report payload will look similar to what is described in Aggregate Reports explainer.

Future improvements

Approved helper groupings

If helpers have varying trust levels based on historical reputation, browsers could choose to only allow certain types of pairings, e.g. at least one in a "very high trust" list and one in a "probationary period" list. If certain pairings would be disallowed, any API that exposes the list of approved helpers should expose these requirements.

Fallback to direct fetching of aggregation service keys

For increased resiliency, if the pre-cached aggregation service keys have expired and the browser client is unable to get an updated copy from its own service (e.g. due to a service outage), it can consider directly fetching the keys from the /.well-known endpoint for helper services it would trust if the key wasn't otherwise expired.

Central helper service registration and approval

Ideally, the set of helper services would be common across browser shipping from different vendors. In that scenario, registration would likely be handled through a shared, central authority. It's to be determined if multiple browser vendors can get sufficiently aligned on helper requirements and validation of compliance.

Given the risks of helpers colluding and the attestations they must make around the functionality of their services, a more formal entity that does additional verification and auditing (perhaps similar to a CA doing EV cert validation which is in turn trusted by multiple browsers) may be appropriate.