Origin isolation: hints, or no?

domenic@chromium.org
Public

Background

Origin isolation is a proposal wherein web pages volunteer themselves to lose some cross-origin capabilities, and in exchange browsers may change their implementation strategies.

The proposal currently includes a number of hints which an origin can use to declare why they are requesting isolation. The vision is that an implementation might use these hints to govern its choices.

In Chrome, our current implementation plan is to always treat any origins opting in to isolation in the same way: by giving them their own process. We would ignore these hints.

This document is to explore which path we should take:

1. Remove the hints from the proposal, at least for now.
2. Keep the hints in the proposal, but do not use them in our implementation yet.
3. Keep the hints in the proposal, and explore ways of using them (in the near term, e.g. 0-2 releases after the initial implementation).

General comments about hint utility

Will pages lie?

If pages are not honest about how they use the hints, then the hints become less useful. In particular, if every page asserts the same set of hints (e.g. the maximal set), then the hints are useless, and the proposal might as well be a simple boolean. It is plausible that this might be the end state, e.g. via cargo-culting from example code, or via an arms race where every frame on the page believes they are more deserving of isolation than others.

domenic@ thinks this is OK. It doesn't mean hints are *less* useful than a boolean. It just means that there is a potential future in which they were wasted work.

Speccing hints before implementing them

The argument for path (2), of speccing hints but not using them in our implementation yet, is that we can try to nudge the ecosystem toward using the hints, and then later our implementation will "light up" with the right behavior on a preexisting corpus of sites.

For example, one partner is mostly interested in origin isolation for parallelism and memory measurement. If they sent the Origin-Isolation header with those hints on day 1, they would get process isolation. But later, when multiple Blink threads ships, we could implement the strategy in Choice of implementation technologies. Then, the same site would get a separate thread/isolate within the same process, with no change on their part. This would translate to decreased memory usage for their users. Nice!

The dangers here are that:

- Designing the hints without a concrete implementation backing may lead to different choices than if we designed them after the implementation was ready. E.g., maybe one of the hints we specify ends up being useless. Or maybe a hint that would have been very helpful was never specified and deployed.
- Transparently changing the implementation strategy used for a site might have unintended negative consequences. Although the changes are not JavaScript-observable, perhaps the site gets less smooth. Thus, a preexisting corpus using the hints might actually *prevent* us from implementing the hints.

Potential ways of using the hints

Force process-isolation if hints are present

Cross-origin isolation (COOP+COEP) also induces origin isolation, in terms of web observable effects on document.domain and WebAssembly.Module. However, the plan is to still share processes if we are over the process limit.

We could by default follow the cross-origin isolation process model and respect the process limit. But, if there are hints present, then we could allocate the extra process even beyond the process limit. We could do this only for some hints, or even combine it with other signals (e.g. only for installed PWAs which send certain hints.)

This helps the two features make sense together conceptually. Both COOP+COEP and the Origin-Isolation headers achieve the same web-observable result and base implementation strategy. But the Origin-Isolation header's additional hints can be used to further change the implementation strategy.

Prioritization in resource-constrained situations

With our current plan of always giving a new process to origins that opt in, we may run into resource constraints, especially on mobile platforms.

We could use the hints to prioritize process allocation decisions. For example:

```
  side-channel-protection
> parallelism = large-allocation
> memory-measurement
```

Choice of implementation technologies

The multiple Blink isolates and multiple Blink threads project provides new technologies, besides dedicated processes, which can fulfill the desires of some of the hints, while being less resource intensive. For example:

- side-channel-protection ⇒ separate process
- parallelism ⇒ separate thread/isolate
- memory-measurement, large-allocation ⇒ separate isolate (same thread)

Note that MBI/MBT are longer-term projects, so if this were the only way we planned to use the hints, then it argues for (1) or (2), and not (3).

Engineering costs

(1) minimizes engineering costs. It also removes the need for any further design discussions on the hints.

(2) puts the engineering cost mostly on the header parser, including the specification and tests. This is relatively minimal (~2-4 days of domenic@'s time). There may be some additional cost on the design side, e.g. resolving the naming of the hints and the header design (mostly captured in issue #18).

(3) needs more detail, e.g. choosing among Potential ways of using the hints, and then getting more information from the site isolation team.

Conclusion

After some time soliciting opinions on this document, no Chromium engineers seemed enthused about hints. domenic@ is taking this as a sign that we should go with (1). This is being committed to the explainer in pull request #26.

If concrete opportunities present themselves in the future, we can add hints back to the specification in parallel with implementing them, and testing the results with partners.