# Service Worker CSRF Discussion

October 6, 2021
https://github.com/w3c/ServiceWorker/issues/1604

## Attendees

- Ben Kelly (google)
- Anne (Mozilla)
- Jake (Google)
- Artur Janc (Google)
- Andrew Sutherland (Mozilla)
- <add yourself>

## Notes

- Ben:
  - In no SW case, doc sends request to server.
  - With SW in middle in can do all sorts
  - Eg from cache, multiple fetches in one etc etc
  - The request client issue: request has a client, which is an env settings object or global
  - This sets origin header and other sec headers
  - Servers may depend on these for CSRF reasons
  - What happens if a SW is in the middle?
  - Fallback is fine
  - Passthrough or other fetches, the request client becomes SW
  - Now it looks like the req is same origin
  - The SW cannot tell if the request is from another origin
  - Chrome side: We need to change something here
  - Easiest thing to do is, when we create a new req without modifying, preserve original client.
  - Some problems doing this in Chromium due to site isolation
  - It prevents using the client from another origin for security
  - We'd probably have to reject further requests once the navigation is satisfied
- Andrew
  - Would that be an issue for cache update cases?
- Ben
  - It'd be difficult with stale while revalidate, because they'd go to the server after completing the response.
- Andrew
  - Would waitUntil be enough?
- Ben

    - That would depend on what security team think.
    - I don't know where the line is.
    - We're going to get to a point where the request no longer works.
    - Eg getting it out of the cache.
- Anne
    - Client can only be used in the initial phase of fetch
    - That's when fetch gets the data from the client
    - I'm not sure if this works
    - By the time we're in the SW, you've gone in-parallel, so you're not in the right place to get detail from the client
    - This would also change what CSP applies
    - Since they're of a particular client
    - Not of the service worker
- Artur
    - The policy of the SW is in charge, not of the request client.
- Anne
    - But fetch looks up the policy from the client
    - You end up looking at the document not the client
- Ben
    - Instead of propagating client, we could propagate details
- Anne
    - Yeah, when we discussed this earlier, I figured we'd have another field for this stuff
- Jake
    - That's how referrer currently works
    - Does that survive going through the sw cache?
- Ben
    - Implementation dependent
- Anne
    - Problem is we didn't consider some of this in the cache
- Ben
    - In gecko we tried to preserve a lot, in Chromium it's just headers
- Anne
    - That might be ok
- Ben
    - I think there's a potential to re-use the request
- Jake
    - Eg when updating everything in the cache
- Ben
    - Would folks expect Origin etc to be preserved
- Jake
    - I don't think we'd preserve the security stuff in that case
- Anne
    - They might want to do that, eg have different entries for cross-site things

    - But that would require different matching
- Ben
    - Outside of SW, vary on origin is useful for this, but we don't expose that in SW
    - So we'd have to build something new into the matching algo
    - Now CORS is done in network process
    - I don't think we'd go back to adding it earlier
    - We could pretend it's already there based on request metadata, for the sake of matching.
- Anne
    - Should avoid 'as if' language
- Ben
    - Seems we have agreement we need to propagate some origin information
- Anne
    - What about A embeds B, and B redirects to A again
    - The origin could be null because it crosses the origin boundary twice
    - If you go A -> B -> C (origins) then C doesn't see A or B in the Origin header
- Lukasz
    - I thought it was A
- Anne
    - I thought we have things in CORS for this
    - Else you get the same kind of CSRF issue we're trying to prevent here
- Artur
    - If there's a redirect, it changes sec-fetch-site
    - If I had to guess, what Anne says is correct
- Ben
    - SW navigations use manual mode
- Anne
    - Is tainting taken into account?
- Ben
    - I have a slide for this later
    - Thinking about it from SameSite
    - I feel we have consensus here that we need to pass through origin as a field on request
    - Sec-fetch-mode is supposed to be preserved, sec-fetch-mode is not
- Artur
    - It has a property that you can launder requests through a SW and evade server side logic. Final site has logic to prevent something loaded as script.
    - Evil.com can have a SW that does a pass-through, and the dest changes to empty, so the response is allowed.
    - If we also propagated that field, it would help
- Anne
    - The problem is that the CSP from the worker applies differently.
    - Could allow laundering as an image but pass as script
    - Eg via synthetic response

    - They're a little far-fetched, but it would be a big change to switch
    - We decided it wasn't what we wanted
    - Didn't think through all the consequences
- Jake
    - Service worker can always create a new request.
- Anne
    - Say it has a non-strict image policy, but that could be sent back as a document
- Andrew
    - There's no way to force the information propagation
- Anne
    - If you couple the destination with the response then it isn't a problem
- Ben
    - Could we track the initial destination in another header, or extra value
- Andrew
    - What's the SW calling to get that
- Ben
    - It wouldn't help if just creating a new request
    - Seems like we don't have consensus on destination
- Artur
    - Let's move forward, but come back to it later
- Ben
    - Next: setting site for cookies for SW
    - If you current frame is different to parent you get a null site for cookies [not sure I captured this correctly]
    - Chromium is looking at partitioning SWs
    - If we do that we can set site for cookies better
    - The SW will have a storage key, and we can use that
    - We want to add a bit to that, if there's a cross-site frame in the ancestor chain
    - With this bit it would be considered third party
- Anne
    - So you have can have two frames that have sync access to each other but different storage?
- Ben
    - It's weird but don't know how else to do it
    - If we want to have clickjacking protection we need different site for cookies
- Andrew
    - So your SWs will be more than double-keyed?
- Ben
    - We're thinking about it
- Anne
    - The bit is "is there an intermediate origin"
- Ben
    - There are other proposals, eg anon iframe, include a nonce
    - We have storage keys with nonces in them for that

    - Do you have plans to partition SWs?
- Andrew
    - Yep, double keying
- Ben
    - Dunno if you'd consider including this extra bit
- Anne
    - We might
    - We have extra rules around cookies
    - But they don't apply with a SW
    - No committal
- Ben
    - Should I file an issue
- Anne
    - Yeah
- Ben
    - It might land on my plate. Eg integrating SW with storage spec.
- Andrew
    - It'd be nice to be able to reason about it via a spec.
- Ben
    - It's on our roadmap.
    - Unclear yet if it's spec or monkeypatching
- Anne
    - There's some consensus on partitioning, but details aren't clear
    - We already partition network
    - We can spec that UAs "are allowed to…" if there isn't total agreement
- Ben
    - For networking HTTP cache partitioning, they have a "is this is subframe" bit, they don't check cross-site.
    - I prefer the more specific bit.
    - We might improve that.
- Andrew
    - For storage, our storage access stuff will only apply to cookies
- Anne
    - Well, we haven't quite decided that yet, but it's the plan
- Ben
    - If you do a cross-site redirect you don't get strict cookies
    - And manual redirect mode breaks this
    - We probably lose tainting information
    - Need to track it through an internal flag
    - Might just need to write some tests and see what happens
    - Because the spec says to use the same origin
- Anne
    - At a high level it feels this should use the same originating origin as sec-fetch etc, so it should cater for tainting.

    - If we do that it should all work out (at a high level)
    - In general we want redirects to taint
    - If A embeds img from B which redirect to img from A we taint it
    - We want that
- Ben
    - I think that makes sense
    - The last thing I had, is the SW can't check where the request came from
    - Doesn't know if it's safe to put in cache storage
    - I think we talked about exposing something similar to sec-fetch-site
    - This would be in lieu of things like origin header etc
- Anne
    - Seems reasonable
- Ben
    - Don't think this would change eg matching in cache algo
- Anne
    - I think the cache API should just store headers and URL
    - Need to file an issue for that
    - Matching matters, but not total replication
- Andrew
    - Would this apply if we had serialisation of request/response?
- Anne
    - I think that would be a different discussion
- Ben
    - It seems weird to me that we lose information, eg cookie information
    - Going back to this getter, if we have that, why don't we just have the header?
- Anne
    - We don't have the header at that point due to CORS. Eg who set the header?
- Ben
    - We could just expose sec-fetch-site
- Anne
    - Same problem - CORS checks
    - End up removing it for CORS, then adding it back?
    - We make copies of the request before serialising headers (might be impl detail)
    - If we get a redirect we adjust headers and serialise them again
- Ben
    - The SW is a proxy, so they expect the headers to be there
    - We can start with the getter then look later
- Anne
    - There is a tension there
    - Would we expose all the headers there?
    - Should the host header be there?
- Ben
    - Happy to start with the getter
- Anne

    - If we want to expose headers there, we should do it in a principled way (not like client hints)
    - We need info for why certain headers are there, and others aren't
    - It might be solvable, but might be hacky/weird
    - As these things maybe changed later, eg redirects
- Ben
    - We could talk more about destination?
- Artur
    - We could expose some stuff to SW
    - If we look at server side stuff to prevent CSRF, they look at lots of metadata
    - We could look at feature parity in the SW
    - We could look at exposing original destination… it's a bit of a can of worms, eg sec-fetch-original-dest
    - But this all vaguely makes sense to me
- Anne
    - Isn't destination already exposed? There's already a destination field
- Ben
    - Yes
- Artur
    - Ah, I might have been trying a couple of years back
- Ben
    - I think there's a conceptual issue. We want origin A to make a request on behalf of origin B, but we don't communicate that.
    - Seems like a weakness.
    - If there a proxying method we could copy here?
- Anne
    - There's two sources of authority. There's A which makes the action happen, and B which forwards the action (the SW).
    - It sounds like we want to forward the origin from A, which is getting weird.
- Ben
    - I wonder if CSP can specify the proxying
    - "I don't want A to do a POST on behalf of B"
- Anne
    - We need to figure out which fields are right to copy from client vs SW

*Locking down doc to avoid any spam.*