Starred by 4 users

Owner: wanderview@chromium.org

CC: falken@chromium.org
mkwst@chromium.org
davidben@chromium.org
jakearchibald@chromium.org
yhirano@chromium.org
wanderview@chromium.org
yoavweiss@chromium.org
miketaylr@chromium.org
adetaylor@chromium.org
kinuko@chromium.org
kouhei@chromium.org
nhiroki@chromium.org
tzik@chromium.org
bingler@chromium.org
merewood@chromium.org
asamidoi@chromium.org
lukasza@chromium.org
annev...@gmail.com
mek@chromium.org
hiros...@chromium.org
michaelnye@chromium.org

Status: Fixed *(Closed)*

Components: UI>Browser>Navigation
Blink>ServiceWorker

Modified: Jul 29, 2022

Backlog-Rank: ----

Editors: ----

EstimatedDays: ----

NextAction: ----

OS: Linux, Android, Windows, Chrome, Mac, Fuchsia, Lacros

Pri: 1

Type: Bug-Security

reward-3000
Security_Severity-Medium
allpublic
reward-inprocess
CVE_description-submitted

external_security_report
M-98
FoundIn-93

Issue 1241188: Security: "Origin" header incorrectly set for cross-site request via service worker

Reported by holeg...@gmail.com on Wed, Aug 18, 2021, 5:33 PM EDT

Code

VULNERABILITY DETAILS

When a cross-site request is made to a site that uses a service worker to fetch the request - the Origin header is incorrectly set to appear to come from the main site.

In the video you can see the problem in action.

Chrome states the Origin is "http://cc.local"
Firefox states the Origin as "null"

I would consider this a serious security issue, as Chrome is simulating that the request "http://cc.local" which is not the actual Origin of the request.

VERSION

Chrome Version: 92.0.4515.159 (Official Build) (64-bit)
Operating System: Windows 10

REPRODUCTION CASE

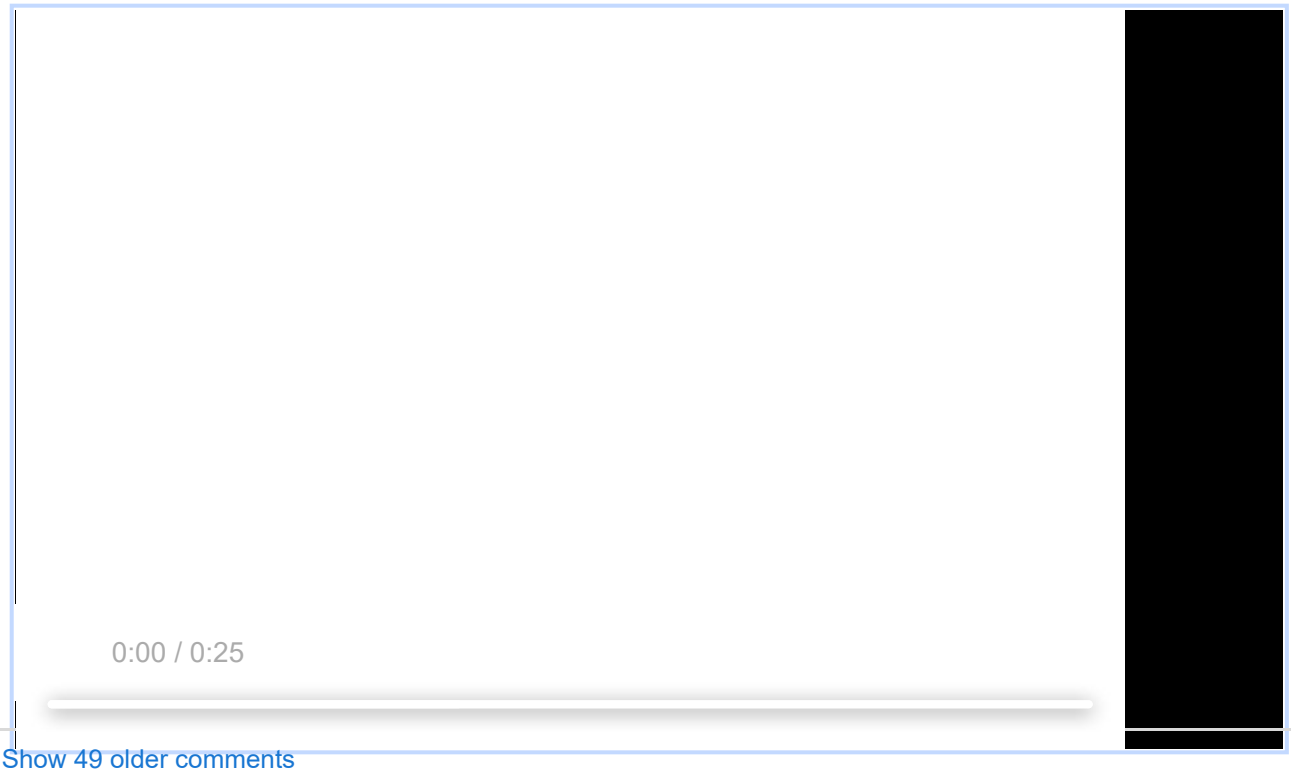
https://github.com/garygreen/chrome-sw-origin-security-issue

CREDIT INFORMATION

Externally reported security bugs may appear in Chrome release notes. If this bug is included, how would you like to be credited?

Reporter credit: garygreen

chrome-sw-origin-prob.mp4
391 KB View Download



Show 49 older comments

Comment 50 by wanderview@chromium.org on Thu, Aug 26, 2021, 11:35 AM EDT

> Can we move forward with using a null origin in this scenario?

I guess I'd rather fully bring us into spec compliance and alignment with other browsers than do a half measure if we can.

> An important property of the current security architecture is that responses to navigation requests never go to/through renderers until the Browser process has determined which renderer can host the HTML document from the response. It seems that this property would be violated by the 4th item from #c42.

How so? The browser has already picked a process in this case (the process containing the service worker as its guaranteed to be same origin to the destination URL). The idea is that the request has a UUID or similarly hard to guess id that a hostile renderer could not forge, so we can rely on only the renderers we give the id to being able to restart the request from the browser process.

An alternative approach would be to try to adapt our logic for handling the case where the service worker does not call respondWith() to this case. Indeed this is very similar to what we do in that case. There might be some differences in how redirects should be handled there, though.

Comment 51 by wanderview@chromium.org on Thu, Aug 26, 2021, 11:39 AM EDT

I'll investigate adapting the "didn't call respondWIth()" fallback logic to handle this case as well. Maybe we don't need to add much at all.

Comment 52 by lukasza@chromium.org on Thu, Aug 26, 2021, 1:01 PM EDT

RE: #c50: wanderview@:

RE: I guess I'd rather fully bring us into spec compliance and alignment with other browsers than do a half measure if we can.

I agree that full spec compliance is a desirable goal. A small, necessary step toward this goal can be called "a half

measure", but such a step still seems desirable to me (fixing a security bug seems desirable on its own; small CLs seem desirable as they are usually easier to understand and less risky).

RE: How so [asking about myearlier claim that "this property would be violated by the 4th item from #c42"]

Good point. The `fetch(...)` API doesn't allow overriding the target URL of a request, and the `init` parameter only allows overriding a limited set of properties. So, the UUID/proxying should work.

I would encourage trying this as a separate step (if possible). This will help with more focused evaluation of the non-zero extra code complexity required by this step. In particular the hierarchy of constituencies asks to treat spec purity as lower priority than complexity of implementations (https://www.w3.org/TR/html-design-principles/#priority-of-constituencies).

Comment 53 by wanderview@chromium.org on Thu, Aug 26, 2021, 1:56 PM EDT

> I would encourage trying this as a separate step (if possible). This will help with more focused evaluation of the non-zero extra code complexity required by this step. In particular the hierarchy of constituencies asks to treat spec purity as lower priority than complexity of implementations (https://www.w3.org/TR/html-design-principles/#priority-of-constituencies).

I would argue that overwriting mode with arbitrary other modes is introducing more complexity and technical debt. We already have a mechanism that says "please restart and continue this navigation request". I think fixing this bug requires a variant of that existing capability, so I think adapting it offers the least added complexity/confusion.

I'll write a design doc and we can have more concrete discussions about added complexity there. Does that sound reasonable?

Comment 54 by lukasza@chromium.org on Fri, Aug 27, 2021, 10:29 AM EDT

RE: #c53: wanderview@: design doc

That sounds totally reasonable. Thank you for looking into fixing this bug.

Maybe to maximize the chance of fixing the security bug in M95 we can "timebox" this exploration and fallback to the narrower solution if needed to land the fix before the M95 branchpoint (which according to https://chromiumdash.appspot.com/schedule will happen on September the 9th).

RE: proxying

Let me try to help with exploring the design space here by sharing one random, brainstorming idea about how the proxying can be potentially implemented. On the Browser process side, the NavigationURLLoaderImpl uses a mojo::Remote<URLLoaderImpl> - it could potentially be stored and wrapped in a new, proxying-kind-of factory (let's call it RequestProxyFactory).

The RequestProxyFactory would remember the original network::ResourceRequest and in its URLLoaderFactory::CreateLoaderAndStart implementation would prevent changing the ResourceRequest's properties, outside of a handful of changes allowed via the `fetch(...)` API's `init` object (which AFAIU makes it possible to override the `method`, `headers`, `body`, `mode`, `credentials` mode, `referrer`, etc.).

Reusing the `mojom::URLLoaderFactory` interface hopefully helps with "response and body data would then have to be plumbed back" from #c42 - we could just reuse the existing mechanism. And on the Browser side, the wrapped URLLoaderFactory is already correctly hooked into DevTools+Extensions/WebRequestApi.

I note that (AFAICT) the DOM Request object can be stashed by the service worker beyond the lifetime of the NavigationURLLoaderImpl. This should be okay with RequestProxyFactory as long as it stays alive while it is bound (this

can be achieved by deriving the implementation from network::SelfDeletingURLLoaderFactory).

2 potential issues that I see with the RequestProxyFactory approach (not necessarily unique to this approach):

1. Have to update RequestProxyFactory's knowledge of allowed ResourceRequest property changes whenever `fetch(...)` API's `init` allows extra properties. I wonder if this needs to be achieved via comments/documentation VS if it would be possible to make RequestProxyFactory compilation fail if there are discrepancies (by having a shadow/duplicate/no-op struct next to RequestProxyFactory that static_asserts that is has to be the same size as the Init payload?).

2. I am not sure how to plumb RequestProxyFactory between the FetchRequestManager layer and wherever the URLLoaderFactory interface is used. Maybe this requires adding a URLLoaderFactoryOverride property/field at the various intermediate layers? (at the C++ class wrapping DOM Request + in other C++ representations of the Request used in Blink).

Comment 55 by wanderview@chromium.org on Fri, Aug 27, 2021, 10:55 AM EDT

In terms of timeframe I was not planning to try to rush something in before September 9. This issue has existed in chrome for years (since service workers were introduced I think) and is only security severity medium. On the other hand, I'm hopeful we will be able to have a solution in place that can be merged up to M95 before it goes stable.

Thanks for the design thoughts on a proxying solution!

Comment 56 by wanderview@chromium.org on Tue, Aug 31, 2021, 8:16 PM EDT

Cc: kouhei@chromium.org asamidoi@chromium.org

Comment 57 by wanderview@chromium.org on Thu, Sep 2, 2021, 5:24 PM EDT

Koehei had another idea in an offline discussion:

1. When a navigation request is sent to the service worker we mint an unguessable token and include it with the request.
2. If the service worker does a fetch(evt.request) then we send the token back to the browser process/network service.
3. The network service would then validate the token if it see a navigation request from a renderer. If the token is valid, then it would permit the request.

The token could be constructed by:

a. Generating a new cryptographic key at browser startup.
b. Use the key to hash key parts of the request together (initiator origin, navigation url, request id, etc.)
c. Use the hash as they token.

To validate the token the network service would just hash the new request and compare the result to the token.

This would require the browser process and network service to share the cryptographic key. Perhaps this would require sending a message to the network service when it starts up.

lukasza@, what do you think of this idea? It would be substantially less complex in terms of plumbing.

Comment 58 by davidben@chromium.org on Thu, Sep 2, 2021, 6:32 PM EDT

Pulling in cryptography for purely local communication always makes me a little suspicious. Not to say it's necessary wrong, just suspicious. It introduces questions like what bits should we hash in, what about replays, or what happens if the request is sent at a much much later time?

Perhaps there's a simpler solution by allowing same-origin requests a bit more control over the Origin header? Or maybe

we just say null is the right answer here?

(NB: I may be misunderstanding some of the parts here. I read through the discussion but had to skim some bits since it's pretty long. It's quite possible all this is nonsense and you should ignore me. In particular I'm not quite following what the state of the spec is. Chrome and Firefox actually match for me, using the link in comment #47, strangely.)

As I understand it, we have some page under origin A making a cross-origin POST navigation to site B. Normally, this would turn into a network request to B's server with "Origin: A", so that B's server knows the request was constructed by A, not B. However, B has a service worker, which means navigations under B go to B/service-worker, not B/server. And the problem is that, when B/service-worker initiates a request, it is tagged "Origin: B". Yet we want to allow B/service-worker to tell B/server that A was involved. And now the conundrum is that we don't want to, in general, allow a service-worker or a compromised renderer process to spoof requests as coming from arbitrary other origins.

Is that right?

It seems to me there are a couple of options that don't require careful binding of request parameters:

Option 1: we say that going through B/service-worker is like a redirect and thus the correct answer is "Origin: null", not "Origin: A", because mixing two origins together gives null. Then we say that a fetch caller is always allowed to downgrade "Origin: yourself" to "Origin: null" because it is strictly lowering capabilities. But this means going through a service worker does not give the same answer as without the service worker, and I gather it not what the spec says? (Again, I'm not quite following what's going on with the spec.)

Option 2: we say that content is allowed to spoof the Origin header *only on same-origin requests*. The observation is B/service-worker already can lie to B/server by upgrading "Origin: A" to "Origin: B". It just makes a new request with the same parameters. So maybe we're willing to say that B/service-worker can claim it's acting on behalf of any other origin, specifically when talking to B/server (i.e. same-origin requests).

Option 2 is incompatible with B/server somehow granting "Origin: C" requests some capability not granted to "Origin: B" requests, but maybe that's fine? It's an odd case, and seems a cleaner model than saying "you can spoof this, even replaying it over time, but only if these various parameters match".

Or maybe I'm completely misunderstanding the issue and this comment is nonsense. :-)

Comment 59 by holeg...@gmail.com on Thu, Sep 2, 2021, 6:48 PM EDT

@davidben

> Chrome and Firefox actually match for me, using the link in comment #47, strangely.)

I thought that too - but they are slightly different if you look at the end of the URL...

Without SW: Origin: https://service-worker-echo-origin-header-form.glitch.me
With SW: Origin: https://service-worker-echo-origin-header.glitch.me

(notice "-form" difference)

Not sure I understand option 2 and the lieing spoofing thing? The issue is that a service worker can claim it's the initiator and override "Origin" during a cross site request.

All that needs to happen here is just to ensure during a cross-site request the service worker set's the origin as "null", or as you say "downgrade" the origin.. Though I have no idea how the renderer plays into this, sounds like there is existing

security protocols in place which prevents this from being a straightforward fix 🤷‍♀️

@wanderview

For the cryptographic key idea that sounds quite complex - would that have any effect on performance on these requests? If so would be good to explore simpler solutions

Comment 60 by davidben@chromium.org on Thu, Sep 2, 2021, 6:53 PM EDT

> I thought that too - but they are slightly different if you look at the end of the URL...

Right. I'm getting that Safari reports -form in both cases, while Chrome and Firefox reports -form w/o service worker and unsuffixed w/ service worker.

Comment 61 Deleted

Comment 62 by holeg...@gmail.com on Thu, Sep 2, 2021, 7:07 PM EDT

Ignore that comment, was testing wrong. Mozilla Firefox seems all secure

Comment 63 Deleted

Comment 64 by lukasza@chromium.org on Thu, Sep 2, 2021, 7:17 PM EDT

RE: #c55: wanderview@: This issue has existed in chrome for years [...] and is only security severity medium

I am worried that we are pursuing a functionally-correct long-term solution, while the security aspect of this Pri1 bug remains unaddressed. I am not sure if I agree with the current label of Security_Severity-Medium (brought up in #c55, stamped in #c18) - I think there are arguments for treating this as high severity, because an unexpected Origin header effectively does allow an attacker to
execute code in the context of, or otherwise impersonate other origins.

RE: #c58: davidben@: "Origin: null" + don't worry about proxying other aspects of the original request

This is what I was trying to encourage in #c54 ("timebox") and #c52. OTOH, in the long-term we will also want to preserve (or at least consider preserving) other aspects of the original request - most importantly mode=navigate (which is visible on the wire in the Sec-Fetch-Mode http request header).

RE: #c57: wanderview@: Introducing a new kind of a capability token

If the capability token doesn't remember the target URL, then this approach seems insecure - a compromised renderer that has received a capability token would be able to issue navigation requests to arbitrary URLs.

If the Browser or NetworkService process remembers the target URL associated with the capability token, then this requires extra complexity to clean-up the Browser/NetworkService-side data structures when the DOM Request object gets garbage collected in the Renderer process. As I pointed out in #c54, the DOM Request object can live for a long time - potentially much longer than the content::NavigationRequest object.

There is potentially one other option to consider for the long-term solution - see item B below. So, I'd like to propose the following next steps:

Step 1: First land the security fix to use request_initiator=opaque-origin in this scenario (this seemed to be within reach according to my understanding of #c47)

Step 2. Then start allowing mode=navigate in CorsURLLoaderFactory::IsValidRequest, but only if the request_initiator.GetTupleOrPrecursorTupleIfOpaque() matches the target URL.

Reasons for doing the 2nd step above as a separate CL:

*) There are quite a few of URLLoaderFactoryParams and ResourceRequest properties - I worry that this approach wouldn't use exactly the same properties as navigation requests initiated from the Browser process.

*) Smaller CLs are desirable in general (easier to review, more bisect friendly, safer to revert, less likely to cause merge conflicts, etc.)

*) It seems desirable to give ourselves more time to get more input from the Chrome Security Architecture and Network teams on whether matching request_initiator and target URL is sufficient. (i.e. if we can ignore other properties of the Renderer process lock [e.g. coep/coop?] and of the ResourceRequest [NIK?])

Comment 65 by lukasza@chromium.org on Thu, Sep 2, 2021, 9:09 PM EDT

RE: #c64: lukasza@ (myself :-)

RE: allowing mode=navigate if request_initiator or its precursor match the target URL

One caveat here is that request_initiator_origin_lock currently allows any opaque origin (i.e. doesn't verify the precursor origin of opaque origins). This would need to be fixed before we could take step2 from #c64. I don't know if fixing this is easy or hard, because this extra validation of request_initiator_origin_lock hasn't been tried yet (OTOH, there are definitely some other known cases where origin precursors might be mismatched in some security or navigation checks - e.g. an Android WebView exception under ChildProcessSecurityPolicyImpl::CanCommitOriginAndUrl).

RE: severity

This was discussed on the security chat and there were good arguments both for keeping it as severity=medium and bumping it up to medium=high. Omitting the Origin header in _most_ cases would definitely be severity=high (this allows for impersonation of another origin), but the extra prerequisites for this bug seem to be a mitigating factor that allows for downgrading to severity=medium. This isn't clear-cut, because the bug doesn't require a specific user action or gesture, using the Origin header as an XSRF defense-in-depth seems reasonable [1], and service workers are an important feature of the modern web. Given that this bug seems to be near the boundary of high and medium severity, it probably makes sense to keep the bug as severity=medium for now.

[1] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

Comment 66 by wanderview@chromium.org on Fri, Sep 3, 2021, 8:44 AM EDT

> If the capability token doesn't remember the target URL, then this approach seems insecure - a compromised renderer that has received a capability token would be able to issue navigation requests to arbitrary URLs.

What I proposed was a hash of the target URL plus additional information from the request. So it would "remember" the target URL.

> If the Browser or NetworkService process remembers the target URL associated with the capability token, then this requires extra complexity to clean-up the Browser/NetworkService-side data structures when the DOM Request object gets garbage collected in the Renderer process. As I pointed out in #c54, the DOM Request object can live for a long time - potentially much longer than the content::NavigationRequest object.

I suggested a hashing mechanism for the token to avoid the necessity of this kind of clean up code.

But again, this was just an idea. We don't have to go this way.

> Step 1: First land the security fix to use request_initiator=opaque-origin in this scenario (this seemed to be within reach according to my understanding of #c47)

If we can set the request_initiator to achieve this, then I agree that we should do that now. I'll take a look. (My previous impression was that the short term fix required overwriting mode with CORS, etc, which I felt would just create more confusion.)

I'll still work on a design doc for a longer term solution.

Comment 67 by wanderview@chromium.org on Fri, Sep 3, 2021, 8:48 AM EDT

Yea, I believe firefox behavior just changed in regards to the origin header. Testing in browserstack shows that firefox 89 does prints "-form" in the SW test case, but FF90+ now don't. I'll let the mozilla folks know.

Comment 68 by wanderview@chromium.org on Fri, Sep 3, 2021, 10:41 AM EDT

Actually, I'm not sure sure setting request_initiator to a null origin will work. If we continue to send "same-origin" for mode, then I think the network service will reject the request. Is that correct?

I also don't think we can set mode to "cors" as was suggested above. That would require the server to provide CORS headers for these kinds of requests to load which is very atypical for navigations.

We might be able to set the mode to "no-cors" and depend on fetch understanding its really a same-origin request to the SW to provide access to the Response, but that could run into CORB.

If we want something like this solution I think perhaps we would need to add a "force null Origin header" flag that fetch could set. We would then leave the request_initiator as the SW's origin, set the flag, and the network service would check the flag when adding the Origin header.

Comment 69 by lukasza@chromium.org on Fri, Sep 3, 2021, 11:05 AM EDT

RE: #c68: "same-origin" for mode [...] will reject the request. Is that correct?

I think so. We already do that (that = overwrite the mode with kSameOrigin) today in some cases pointed out in #c46.

RE: #c68: That would require the server to provide CORS headers for these kinds of requests to load which is very atypical for navigations.

I think this (requiring Access-Control-Allow-Origin response header) should already be the case for cross-origin POST navigations without a service worker. Is it not the case?

RE: #c68: force null Origin header flag

This is tricky, because request_initiator is consulted not only for CORS (where it influences the Origin header), but also for other features (e.g. Sec-Fetch-Site). And if we change request_initiator then mode=same-origin will fail the request for opaque initiators.

Maybe the following would work:
1. Production code changes:
    1.1. Tweak CorsURLLoaderFactory::IsValidRequest to accept mode=navigate, but only if:
        1.1.1. request_initiator or its precursor origin matches request_initiator_origin_lock

(this requires a TODO to eventually move precursor checks into VerifyRequestInitiatorLock)
1.1.2. request_initiator or its precursor is same-origin with the target URL of the request
2. Test coverage (verifying Origin, Sec-Fetch-Site, Sec-Fetch-Mode):
2.1. This bug: foo.com -> service-worker.com POST (expecting "Sec-Fetch-Site: cross-origin" and "Origin: null")
2.2. service->worker.com -> service-worker.com POST (expecting "Sec-Fetch-Site: same-origin", right?)

PS. "Origin or its precursor origin" = url::Origin::GetTupleOrPrecursorTupleIfOpaque (repackaged as url::Origin??)
PPS. Sorry for misunderstanding the token proposal yesterday :-/.

Comment 70 by wanderview@chromium.org on Fri, Sep 3, 2021, 11:27 AM EDT

> I think this (requiring Access-Control-Allow-Origin response header) should already be the case for cross-origin POST navigations without a service worker. Is it not the case?

It is not needed for navigations, no. This is a POST navigation per the spec and not a CORS request. No CORS headers should be required.

> This is tricky, because request_initiator is consulted not only for CORS (where it influences the Origin header), but also for other features (e.g. Sec-Fetch-Site). And if we change request_initiator then mode=same-origin will fail the request for opaque initiators.

What if we made this flag affect Sec-Fetch-Site as well? Something like `expose_request_initiator_as_opaque = true`.

Comment 71 by lukasza@chromium.org on Fri, Sep 3, 2021, 11:30 AM EDT

RE: #c70: What if we made `expose_request_initiator_as_opaque` flag affect Sec-Fetch-Site as well?

This seems fragile - it requires that whenever somebody reads network::ResourceRequest::request_initiator they might need to be careful and consider taking `expose_request_initiator_as_opaque` into account.

Comment 72 by wanderview@chromium.org on Fri, Sep 3, 2021, 11:33 AM EDT

I guess that leads us back to the longer term solutions, then.

Comment 73 by wanderview@chromium.org on Fri, Sep 3, 2021, 2:03 PM EDT

Looking at this idea from #c69:

> Maybe the following would work:
> 1. Production code changes:
> 1.1. Tweak CorsURLLoaderFactory::IsValidRequest to accept mode=navigate, but only if:
> 1.1.1. request_initiator or its precursor origin matches request_initiator_origin_lock
> (this requires a TODO to eventually move precursor checks into VerifyRequestInitiatorLock)
> 1.1.2. request_initiator or its precursor is same-origin with the target URL of the request

To make sure I understand, you are suggesting that we:

a. Have the renderer keep the mode as navigate instead of overwriting to same-origin.
b. If the request_initiator is cross-origin, then we set it to `request_initiator.DeriveNewOpaqueOrigin()`. This captures the precursor origin.
c. We then make the network service changes you mention above.

Yes, that does seem doable. Let me try that in a CL to see if anything blows up. Thanks!

Comment 74 by wanderview@chromium.org on Fri, Sep 3, 2021, 2:08 PM EDT

Also, I am floating the idea of forcing to opaque origin in the spec discussion as well.

Comment 75 by wanderview@chromium.org on Fri, Sep 3, 2021, 2:11 PM EDT

lukasza corrected something in chat. #c73 should read:

b. If the request_initiator is cross-origin to context_origin, then we set it to `context_origin.DeriveNewOpaqueOrigin()`. This captures the precursor origin of the context.

We need to derive the opaque from the context origin and not from the request_initiator.

Comment 76 by wanderview@chromium.org on Fri, Sep 3, 2021, 5:18 PM EDT

Here's a twist. It turns out we set the Origin header for POST navigation requests in the renderer anyway. So in theory I can just patch that up in FetchManager. (This seems pretty bad if we just let renderers make this stuff up.)

That feels terribly unsatisfying, though, as the request_initiator is still using same-origin and will trigger other headers to perhaps be wrong. So I'm still going to try forcing the request_initiator to an opaque origin.

Comment 77 by wanderview@chromium.org on Fri, Sep 3, 2021, 5:19 PM EDT

Note, setting the request_initiator to an opaque origin does appear to be working so far. So I don't think do this extra work will take much extra work.

Comment 78 by wanderview@chromium.org on Fri, Sep 3, 2021, 5:34 PM EDT

WIP CL:

https://chromium-review.googlesource.com/c/chromium/src/+/3115917

Comment 79 by holeg...@gmail.com on Mon, Sep 6, 2021, 9:21 PM EDT

I've found another security vulnerability - it may share some similarities with this issue as it's to do with cross-site requests and vulnerability in a service worker. However, it works based on an entirely different mechanism.

Someone on your team has already closed the issue claiming it's a "duplicate" however I am very dubious, as service workers didn't even exist when the "duplicate" issue was created.

Are any of you in a position to check if this was a mistake? I would be most grateful if you can provide a second opinion:

https://bugs.chromium.org/p/chromium/issues/detail?id=1246961

Comment 80 by wanderview@chromium.org on Tue, Sep 7, 2021, 9:51 AM EDT

I would need someone to cc me to that bug.

Comment 81 by davidben@google.com on Tue, Sep 7, 2021, 10:46 AM EDT

It seems to have been triaged correctly to me.

> as service workers didn't even exist when the "duplicate" issue was created.

Hrm? The issue it was merged into also involves service workers and long post-dates them.

Comment 82 by kouhei@chromium.org on Tue, Sep 7, 2021, 10:20 PM EDT

Cc: hiros...@chromium.org

Comment 83 by wanderview@chromium.org on Fri, Sep 10, 2021, 5:04 PM EDT

Ok, the WIP CL is now fully working:

https://bugs.chromium.org/p/chromium/issues/detail?id=1241188

I will update comments, add tests, and send for review on Monday.

I also updated:

https://service-worker-echo-origin-header.glitch.me/

To echo a few more headers. This shows that the WIP sends consistent values for origin, referer, and sec-fetch-site. It also shows that we no longer overwrite sec-fetch-mode.

We still haven't decided anything at the spec level, though. We will likely have a session at TPAC about service worker CSRF issues to discuss.

Comment 84 by wanderview@chromium.org on Mon, Sep 13, 2021, 11:05 AM EDT

Note, my initial testing of firefox was flawed. On some versions of FF I started in the middle of my demo and missed getting the SW installed. That is what produced the "regression" in the results. Retesting correctly shows firefox has been sending the SW's origin in the header consistently.

Comment 85 by wanderview@chromium.org on Mon, Sep 13, 2021, 5:46 PM EDT

Status update:

I am still working on the test. I'm running into some test harness issues that I need to figure out.

Also, I am aiming now to preserve the original cross-origin referer and origin headers. Since these are set in the renderer we can set them as expected by the spec even if the request_initiator is an opaque origin. We can bite off the longer term solution if/when folks want to invest in moving the navigation origin header code to network service.

Comment 86 by wanderview@chromium.org on Tue, Sep 14, 2021, 10:25 AM EDT

Cc: ann...@annevk.nl

So now that its clear chrome and firefox have the same legacy behavior and match what has been in the spec, the fetch spec editor is asking us to hold off on making changes to chromium until we can determine the desired spec change.

How does the security team feel about that?

CC'ing Anne as the fetch spec editor.

Comment 87 by wanderview@chromium.org on Tue, Sep 14, 2021, 10:45 AM EDT

Cc: -ann...@annevk.nl annev...@gmail.com

Comment 88 by lukasza@chromium.org on Tue, Sep 14, 2021, 12:30 PM EDT

RE: aiming now to preserve the original cross-origin referer and origin headers

AFAIU there is no clear requirement for that + that diverges from the current behavior of Firefox. When

RE: Since these are set in the renderer we can set them as expected by the spec even if the request_initiator is an opaque

origin.

I want to note a TODO that you've pointed out exists in services/network/cors/cors_url_loader.cc:

```
// We exclude navigation requests to keep the existing behavior.
// TODO(yhirano): Reconsider this.
```

I think that our high-level principles would require that the Origin header (and other security-sensitive headers like Sec-Fetch-Site, etc) should *not* be controlled by a renderere process. It may seem okay in this case, because *initially* the request initiator (or its precursor) is the same origin as the target URL of the request, but I wonder what happens during 307 redirects. See also a longer comment in https://chromium-review.googlesource.com/c/chromium/src/+/3115917/10#message-9b6b60b0916ccc4eafdb24156db8fa5501748895

RE: How does the security team feel about that?

Not sure if I can speak for the whole security team, but 1) we should verify what happens during 307 redirects and 2) even if 307 redirects are okay, using "Origin: null" would still seem like a safer starting point ("safer" meaning: A) more restrictive = more secure + B) safer to relax in the future without backcompatibility concerns).

Comment 89 by wanderview@chromium.org on Tue, Sep 14, 2021, 1:30 PM EDT

> AFAIU there is no clear requirement for that + that diverges from the current behavior of Firefox.

If we are going by firefox behavior then this bug is a WONTFIX. What I am proposing matches safari behavior.

Also, the discussion from the spec group so far is fairly against setting an opaque origin header.

> I think that our high-level principles would require that the Origin header (and other security-sensitive headers like Sec-Fetch-Site, etc) should *not* be controlled by a renderere process.

Sure, but throughout this bug you have been consistently pushing me to do something short term to fix the security problem quickly. Changing how we set origin headers for all non-get navigations seems like a separate bug, does not seem like a quick fix, and is not something I feel prepared to take on. I think the network or loading team would need to drive the work for that.

When that work happens, though, we can also take the time to design the correct long term fix for service worker forwarded navigation requests without breaking service worker behavior. I was prepared to do this longer term SW solution here, but you convinced me that we should get a shorter term solution done instead.

> Not sure if I can speak for the whole security team, but 1) we should verify what happens during 307 redirects and 2) even if 307 redirects are okay, using "Origin: null" would still seem like a safer starting point ("safer" meaning: A) more restrictive = more secure + B) safer to relax in the future without backcompatibility concerns).

I'm not sure I understand. My question for the security group was about holding off on doing anything here until the spec discussion resolved. It doesn't seem like you addressed that at all?

> I wonder what happens during 307 redirects. See also a longer comment

Service worker navigation requests do not follow redirects. They use the manual redirect mode. So I think it will do the right thing, but we can test it.

Also, can we please keep security sensitive discussions in the private bug and out of the public CL?

Comment 90 by wanderview@chromium.org on Tue, Sep 14, 2021, 3:50 PM EDT

FWIW, I updated https://service-worker-echo-origin-header-form.glitch.me/ to have some redirect test paths. The redirect flow is:

1. "-form" origin has a button with action pointing to "-redirect" origin.
2. "-redirect" origin redirects to main target origin (no suffix on the origin)

Without a service worker the 3 browser behaviors for this kind of redirect are:

chrome: "origin: null"
firefox: "origin: https://service-worker-echo-origin-header-form.glitch.me" (form origin)
safari: "origin: null"

With a service worker the 3 browser behaviors are currently:

chrome: "origin: https://service-worker-echo-origin-header.glitch.me" (sw origin as described by this bug)
firefox: "origin: https://service-worker-echo-origin-header.glitch.me" (sw origin)
safari: "origin: https://service-worker-echo-origin-header-form.glitch.me" (form origin)

I realized I can actually remove any changing of origin headers in my CL. With that we get:

modified chrome: "origin: null"

Matching the no service worker case.

In regards to a compromised renderer, though, I think you are correct that allowing a renderer to send a navigation will allow it to spoof any origin header it likes. This probably applies to the referer as well.

This suggests two solutions:

1. The longer term solution I proposed earlier of proxying the navigation request back through the FetchEvent mojo connection so it can be initiated through browser process.
2. Change all non-get navigations to set origin and referer in the network service and then also do the proposed change currently in the WIP CL.

I'm inclined to wait for the spec discussion to resolve and then, if we need to make a change, implement (1). It gives us the best match to the way the spec is architected and doesn't force any artificial restrictions.

I'm disinclined to pursue (2) because changing how all non-get navigations work seems like a big task and I don't feel prepared to take that on. In addition, it would leave us with technical debt in terms of this weird artificial opaque origin.

Comment 91 by holeg...@gmail.com on Tue, Sep 14, 2021, 3:57 PM EDT

@wanderview - did Firefox update the behaviour? Because I'm sure when I was testing this initially Firefox was returning null for cross site requests.

Makes me thing guys over at Mozilla decided to update Firefox to less secure behaviour in order to follow spec? 8_d

Could be mistaken though

Comment 92 by wanderview@chromium.org on Tue, Sep 14, 2021, 4:03 PM EDT

No, Firefox did not change anything recently. At least in my case I ran some tests accidentally without the SW installed which led to previously incorrect results. It's also possible that you are testing in an environment with a different referrer

policy set.

Comment 93 by holeg...@gmail.com on Tue, Sep 14, 2021, 4:11 PM EDT

I was testing using file:// as cross-site initiator. I've confirmed in Mozilla service worker is registered and running and even now it's returning `null` as origin when submitted thru sw

So looks like Mozilla generally sees file:// as dodgy and cross site, but behaves differently on live site with https. Not sure why that would be though, if this is a spec thing

Comment 94 by wanderview@chromium.org on Tue, Sep 14, 2021, 4:20 PM EDT

I guess an option is:

3. Modify the CL to keep the origin (and referer) in sync with the request_initiator and then validate the headers in the network service for navigations coming from a renderer.

I don't love this option as I think the opaque origin header is a bit weird and unexpected for developers. Also, if we keep the referer in sync as being opaque then we deprive developers of information they could be usefully collecting for their product today. That's a bit breaking.

But I think we should see where the spec discussion goes. My impression from Anne is that we may end up saying the current firefox and chrome behavior is intended and we shouldn't change anything at all.

Comment 95 by holeg...@gmail.com on Tue, Sep 14, 2021, 4:25 PM EDT

> My impression from Anne is that we may end up saying the current firefox and chrome behavior is intended and we shouldn't change anything at all.

Are you suggesting this may not be fixed in the spec... and origin + sec headers + same site cookies will stay vulnerable when a site has a service worker? That seems bonkers.

I would push for a spec change if this is seen as "intended behaviour"...

Comment 96 by wanderview@chromium.org on Tue, Sep 14, 2021, 4:35 PM EDT

To help get the spec discussion to conclusion I scheduled a TPAC session:

https://github.com/w3c/ServiceWorker/issues/1604

Comment 97 by holeg...@gmail.com on Tue, Sep 14, 2021, 5:08 PM EDT

Thanks for doing that, sounds good! It's the "and we shouldn't change anything at all" comment that's got me concerned.

This change isn't just for CSRF but I suspect is having an impact on same-site cookies vulnerability, and probably others.

I think if nothing is changed it would be sensible to discuss what wider trust issues that could cause in the community for security of PWAs. If nothing changes, then this vulnerability is open for public consumption. Both seem unpalatable.

Comment 98 by wanderview@chromium.org on Wed, Sep 15, 2021, 11:41 AM EDT

I'm continuing to work on the short term fix in my CL so we can land it ASAP if we get a resolution.

I have my CL updated to send opaque origin/referer headers and to check them in network service. I also have a wpt_internal test.

I still need to add a unit test that shows the CorsURLLoaderFactory rejects invalid renderer requests. I tried doing this in a normal unit test, but it crashes the entire test. I think I might need to figure out how to do this in a browser test.

Comment 99 by lukasza@chromium.org on Wed, Sep 15, 2021, 12:08 PM EDT

RE: #c89: wanderview@: Service worker navigation requests do not follow redirects. They use the manual redirect mode. So I think it will do the right thing.

I see. That would indeed help. We should make sure there is an explicit verification of the redirects mode in CorsURLLoaderFactory::IsValidRequest (i.e. only allow mode=navigate from renderer if 1) it is same-origin/precursor wrt the target URL *and* 2) it uses manual redirect mode).

RE: #c90: wanderview@: I think you are correct that allowing a renderer to send a navigation will allow it to spoof any origin header it likes.

If we enforce manual redirect mode, then the renderer would only be able to control the Origin header in same-origin requests, right? So it seems okay maybe? (Unless the renderer-controlled Origin header can be used cross-origin in other requests, where redirects are not required.)

RE: #c89: wanderview@: Changing how we set origin headers for all non-get navigations seems like a separate bug, does not seem like a quick fix, and is not something I feel prepared to take on.

That's fair. It seems okay to proceed with renderer-controlled Origin header if we are confident that this doesn't allow spoofing of the header in cross-origin requests (otherwise it would be a regression on https://chromium.googlesource.com/chromium/src/+/main/docs/security/compromised-renderers.md#http-request-headers). It does seem more risky then enforcing the Origin header in OOR-CORS/NetworkService, but there are no known security issues + it would fix a known security bug (this bug).

Comment 100 by lukasza@chromium.org on Wed, Sep 15, 2021, 12:14 PM EDT

Cc: adetaylor@chromium.org

+adetaylor@ help with the urgency/scheduling/spec considerations

RE: #c89: wanderview@: My question for the security group was about holding off on doing anything here until the spec discussion resolved.

@adetaylor, could you please chime in on this? IMHO if we have a fix that 1) addresses this CSRF / incorrect-Origin-header bug and 2) doesn't introduce new security concerns, then 3) we should proceed with landing the fix sooner rather than later. This is a tricky area and I may be missing something, but I think that such fix might indeed be possible (e.g. see #c99). I also think that it is okay to proceed with a partial/imperfect fix, even if subsequent follow-ups might be needed (e.g. to get spec agreement + to cover mode=navigate in OOR-CORS).

Comment 101 by mmenke@chromium.org on Wed, Sep 15, 2021, 12:14 PM EDT

Cc: -mmenke@chromium.org

Comment 102 by wanderview@chromium.org on Wed, Sep 15, 2021, 3:16 PM EDT

Re #c99 lukasza@: If we enforce manual redirect mode, then the renderer would only be able to control the Origin header in same-origin requests, right? So it seems okay maybe? (Unless the renderer-controlled Origin header can be used cross-origin in other requests, where redirects are not required.)

I thought the threat model was a compromised renderer. If the renderer is compromised, it seems like it could run arbitrary code to append an origin header to a navigation request like this regardless of redirection. I think the network service needs to validate the origin header on renderer-sourced navigations to prevent this kind of spoofing in compromised

renderers.

Or maybe I misunderstand the threat model here? Sorry for my confusion.

Comment 103 by lukasza@chromium.org on Wed, Sep 15, 2021, 3:46 PM EDT

RE: #c102: wanderview@:

Even if a renderer process is compromised, the NetworkService process can securely enforce that:

1. The renderer can only control the Origin header if mode=navigate (I believe this is already done in OOR-CORS; if not, then it would be a separate bug)

2. The renderer can only use mode=navigate if
2.1. (secure, trustworthy, browser-process-controlled) request_initiator_origin_lock matches the (untrustworthy, renderer-controlled) request_initiator (or its precursor for opaque initiators)
2.2. the target URL of the request is same-origin with request_initiator (or its precursor for opaque initiators)
2.3. the request uses RedirectMode::kError (I may have misspoke earlier in #c99 about RedirectMode::kManual - it seems that the manual mode might still allow redirects to go through if the renderer calls FollowRedirect later).

I assume that 2.1-2.3 are sufficient to prevent a compromised renderer from triggering HTTP requests that target a URL that is cross-origin from request_initiator. (Please shout if I missed some scenario here; redirects seem to be the only known problem here.)

I apologize for missing that in the WIP CL 2.1 was not fully present (because VerifyRequestInitiatorLockWithPluginCheck doesn't verify precursor origins today) - the CL should explicitly compare the initiator-or-precursor with request_initiator_origin_lock (by calling VerifyRequestInitiatorLockWithPluginCheck).

All of 2.1 - 2.3 seem doable in the WIP CL. It seems that the approach in the WIP CL is most viable one in the short term (although I recognize that its complexity has grown as we continue to explore it; my unscientific gut-feeling is that proxying would be more complex to achieve).

I don't know if RedirectMode::kError (rather than RedirectMode::kManual) is compatible with functional requirements here.

RE: I think the network service needs to validate the origin header on renderer-sourced navigation

This sounds like a good idea. As a defense-in-depth it might indeed be worth checking the Origin header in mode=navigate requests to ensure that it is either "Origin: null" or that it matches a non-opaque request_initiator. (OTOH, it doesn't seem strictly necessary, because (given 2.1-2.3) it seems that a compromised renderer can only lie to the same-origin HTTP server.)

And if RedirectMode::kError cannot be use for some reason, then maybe checking the Origin header is a good idea.
OTOH, it might be worth double-checking with yhirano@ and toyoshim@ on whether applying OOR-CORS to mode=navigate requests might be a preferrable approach.

Comment 104 by wanderview@chromium.org on Wed, Sep 15, 2021, 4:08 PM EDT

> I assume that 2.1-2.3 are sufficient to prevent a compromised renderer from triggering HTTP requests that target a URL that is cross-origin from request_initiator. (Please shout if I missed some scenario here; redirects seem to be the only known problem here.)

I think this is only adequate if the network service is adding the origin header based on the request_initiator. In the current architecture, though, non-get navigations have the origin header added much earlier (in the renderer I believe). This gives

the compromised renderer the ability to change the origin header to whatever it wants irrespective of the request_initiator.

> I don't know if RedirectMode::kError (rather than RedirectMode::kManual) is compatible with functional requirements here.

No, that would not be ok from a spec perspective. It would be very breaking for the exposed web API.

But can you explain more why the renderer following redirect would be a problem? Wouldn't it trigger another request that will be validated again by the network service? It would seem the origin header validation checks would catch any shenanigans here.

Comment 105 by wanderview@chromium.org on Wed, Sep 15, 2021, 4:11 PM EDT

To clarify, I don't think there is anything that says the renderer must set the origin header based on the request_initiator for navigation requests. (And my testing supports that.) The network service needs to enforce the restriction.

Comment 106 by wanderview@chromium.org on Wed, Sep 15, 2021, 5:03 PM EDT

The CL is now updated with working unit tests that verify the CORS loader factory is actually blocking bad initiators, headers, and redirect modes.

The CL still needs some cleanup before its ready for review, though.

Comment 107 by lukasza@chromium.org on Thu, Sep 16, 2021, 11:32 AM EDT

RE: #c104: But can you explain more why the renderer following redirect would be a problem? Wouldn't it trigger another request that will be validated again by the network service?

The validation needs to happen not only in CreateLoaderAndStart / CorsURLLoaderFactory::IsValidRequest, but also in URLLoader::FollowRedirect (AFAIU in RedirectMode::kManualRedirect the network::CORSURLLoader [in NetworkService] will call mojom::URLLoaderClient::OnReceiveRedirect [in a renderer process] which has a comment indicating that the receipient can call back [into the NetworkService process] mojom::URLLoader::FollowRedirect which has an ability to modify and remove http request headers). Some earlier work in this area happened in ~~issue 973103~~ and r668849 and mmenke@ might be able to help.

Comment 108 by wanderview@chromium.org on Thu, Sep 16, 2021, 11:59 AM EDT

We had an internal meeting about this today. Our position going into the spec meeting is going to be that we need to do something like aligning to the safari behavior. (This means preserving the origin header across pass-through fetch handlers.)

Given that, we plan to land our short term fix when its ready, even if the spec discussion is not resolved. If the spec aligns on a different solution we will have plenty of time to revert the change before it hits stable channel.

Comment 109 by adetaylor@chromium.org on Thu, Sep 16, 2021, 7:27 PM EDT

Re #c100:

It sounds like lukasza@ (#c100) and wanderview@ (#c108) are aligned that we should land an interim fix, when it's ready, even if a fuller fix might take longer. That SGTM. I would be opposed to merging this to stable, as I think we should give maximal time for any unanticipated compatibility consequences to show up.

Comment 110 by annev...@gmail.com on Tue, Sep 21, 2021, 6:17 AM EDT

I don't see how preserving the Origin header helps with CSRF. Are you also going to change "site for cookies"? It seems like a very selective change that doesn't address the problem of where authority lies with requests initiated by the service

worker.

Comment 111 by wanderview@chromium.org on Tue, Sep 21, 2021, 9:21 AM EDT

The short term fix does more than change the origin header. It also changes the request_initiator internal field that is used by the network service to populate Sec-Fetch-* headers, etc. I believe if the request_initiator is not the same as the site for cookies, then samesite cookies will not be sent. I'm double checking on that, though.

Comment 112 by wanderview@chromium.org on Tue, Sep 21, 2021, 9:44 AM EDT

I think this code is where we use the request_initiator in the samesite cookie computation:

https://source.chromium.org/chromium/chromium/src/+/main:net/cookies/cookie_util.cc;l=153;drc=8f95b5eab2797a3e26bba299f3b0df85bfc98bf5

Comment 113 by annev...@gmail.com on Tue, Sep 21, 2021, 12:08 PM EDT

Ah, that does sound like an interesting middle ground. There will still be the two sources of authority as the service worker will be responsible for enforcing various policies (and presumably Sec-Fetch-Dest would still be "empty", not "document", to avoid confusing CSP) and a number of values associated with the request, but maybe it's good enough? It would not surprise me if we keep running into subtle issues here though.

Comment 114 by wanderview@chromium.org on Wed, Sep 22, 2021, 5:31 PM EDT

The WIP CL is getting close. I think I have all the requested changes (except comments) in place now. There are a couple things I still want to do before cleanup/review.

I realized I need to plumb request_initiator through the request stored in cache_storage. Otherwise we could end up with oddities like:

```
// uses opaque initiator
evt.respondWith(fetch(evt.request));

// uses SW origin initiator
evt.respondWith(async function() {
  const c = await caches.open('foo');
  await c.put(evt.request, new Response(''), { ignoreMethod: true });
  const keys = await c.keys(evt.request);
  return fetch(keys[0]);
}());
```

This doesn't matter for the origin header since we only permit GET requests in cache_storage, but it could matter for sec-fetch-site, samesite cookies, etc.

I also want to write a test showing that this CL fixes samesite cookie behavior in service workers.

Comment 115 by annev...@gmail.com on Thu, Sep 23, 2021, 7:03 AM EDT

The cache API shouldn't have access to any of those headers though, right? Or are you thinking of including the origin in what it takes to create a cache match?

Comment 116 by wanderview@chromium.org on Thu, Sep 23, 2021, 9:31 AM EDT

>The cache API shouldn't have access to any of those headers though, right? Or are you thinking of including the origin in what it takes to create a cache match?

I'm talking about preserving the internal request_initiator origin field when storing in cache_storage. There would be no observable difference to the code running in the SW. The server, however, would continue to get headers/cookies consistent with a passthrough fetch.

The alternative would be to not preserve the request_initiator so it appear the request coming from cache_storage was created by the current service worker's origin. I guess this might be ok since laundering through cache_storage is a bit weird. But I hesitate to leave it like that because service workers have no way to tell if an incoming navigation request is from a potentially hostile intiator or not. So they can't know if its safe to put that request in cache_storage or not. Maybe we could consider something like a FetchEvent.request.same-origin-initiator flag or something (or maybe same-site-initiator).

Comment 117 by wanderview@chromium.org on Thu, Sep 23, 2021, 1:36 PM EDT

I will say, one thing I dislike about the short term CL fix is that it makes same-site requests appear to be cross-site. This could be breaking for some sites.

Comment 118 by wanderview@chromium.org on Thu, Sep 23, 2021, 4:51 PM EDT

I've got enough of my same-site cookie test working to see that there is also a potential interop problem here as well. When we set the request_initiator to be an opaque origin it apparently blocks SameSite=Lax cookies from being sent. That seems like a potentially pretty big breaking change to me. It would mean site using passthrough service workers would suddenly have users complaining that when they navigate to their page they are signed out.

lukasza@, do you have any ideas to fix this in the short term approach? I wonder if I should instead try to get the longer term approach working.

Comment 119 by wanderview@chromium.org on Thu, Sep 23, 2021, 5:13 PM EDT

I'm checking with the SameSite folks as well in case I am missing some other piece of information that allows Lax for these "auxiliary" navigations.

Comment 120 by lukasza@chromium.org on Thu, Sep 23, 2021, 8:04 PM EDT

RE: #c118 and #c119:

Maybe I don't fully understand how samesite cookies are supposed to work, but I would expect that without a service worker a top-level navigation to bar.com would work the same way if it is initiated from 1) opaque origin and 2) cross-site origin foo.com (i.e. I would expect no samesite cookies in both of these cases). It seems to me that having the same behavior in presence of a "forwarding" service workers is okay.

OTOH, you are right that the fix does risk breaking some websites - requests that were treated as same-origin (no CORS, SameSite cookies, Sec-Fetch-Site, etc) will after the fix be treated as cross-site. This is unfortunate, but ultimately we *do* want CORS for POST navigations initiated by another origin (and similar argument can be made for SameSite cookies, Sec-Fetch-Site, etc.).

Comment 121 by wanderview@chromium.org on Fri, Sep 24, 2021, 11:31 AM EDT

> Maybe I don't fully understand how samesite cookies are supposed to work, but I would expect that without a service worker a top-level navigation to bar.com would work the same way if it is initiated from 1) opaque origin and 2) cross-site origin foo.com (i.e. I would expect no samesite cookies in both of these cases). It seems to me that having the same behavior in presence of a "forwarding" service workers is okay.

SameSite=Strict are not sent in this case, but SameSite=:Lax (the default) are supposed to be sent for cross-site main frame navigations.

I think this is failing in my CL because the "main frame" bit comes from the IsolationInfo which is of course not main frame for the service worker. I'll see if I can find a reasonable way to fix this without being too hacky.

> OTOH, you are right that the fix does risk breaking some websites - requests that were treated as same-origin (no CORS, SameSite cookies, Sec-Fetch-Site, etc) will after the fix be treated as cross-site. This is unfortunate, but ultimately we *do* want CORS for POST navigations initiated by another origin (and similar argument can be made for SameSite cookies, Sec-Fetch-Site, etc.).

My other concern was more about make same-site initiators look like cross-site initiators. I agree there should be an origin header, sec-fetch-site, etc, that says something other than same-origin, but forcing same-site to cross-site may break some product integrations. This one is less likely to be a problem then the cookies, though.

Comment 122 by wanderview@chromium.org on Fri, Sep 24, 2021, 3:27 PM EDT

Cc: miketaylr@chromium.org bingler@chromium.org

Ok, I was able to fix SameSite=Lax cookies on navigations. I'm not sure the fix will be acceptable, though. It does the following:

1. Plumbs the original request.destination through a fetch(). Not sure this will pass spec discussion. If not, we can use an internal flag instead.
2. Makes url_request_http_job.cc look at the destination in addition to the IsolationInfo. [1] This unfortunately moves this SameSite check from being wholely controlled by the browser process to accepting input from the renderer process. Not sure if this is acceptable.

Networking folks on this bug, what do you think? Also adding some SameSite cookie folks for their opinion.

[1]:
https://source.chromium.org/chromium/chromium/src/+/main:net/url_request/url_request_http_job.cc;l=600;drc=cd71ede85341000da32cc74a2044eb5834e5e05b

Comment 123 by wanderview@chromium.org on Thu, Oct 7, 2021, 1:33 PM EDT

In regards to comment 122 I got some feedback from a cookie owner that this is probably ok. We can also validate that its only set in certain controlled circumstances.

We also had the spec meeting and have general agreement to propagate the internal request origin value.

The meeting did raise another problem, though. We need to properly taint requests that have performed cross-site redirects. My CL does not currently do that. Looks like that will require plumbing through the entire list of redirected URLs in order for this logic to trigger:

https://source.chromium.org/chromium/chromium/src/+/main:services/network/sec_header_helpers.cc;l=107;drc=bc987503e9be79c8d2de7487ef2ef12133fec908

Comment 124 by wanderview@chromium.org on Thu, Oct 7, 2021, 5:44 PM EDT

I have a rough working draft of a CL to fix the redirect tainting:

https://chromium-review.googlesource.com/c/chromium/src/+/3213310

Not sure if I should fold this into my original (quite large) CL or keep it a separate CL.

Comment 125 by sheriffbot on Fri, Oct 22, 2021, 12:21 PM EDT

wanderview: Uh oh! This issue still open and hasn't been updated in the last 14 days. This is a serious vulnerability, and we want to ensure that there's progress. Could you please leave an update with the current status and any potential blockers?

If you're not the right owner for this issue, could you please remove yourself as soon as possible or help us find the right one?

If the issue is fixed or you can't reproduce it, please close the bug. If you've started working on a fix, please set the status to Started.

Thanks for your time! To disable nags, add the Disable-Nags label.

For more details visit https://www.chromium.org/issue-tracking/autotriage - Your friendly Sheriffbot

Comment 126 by wanderview@chromium.org on Fri, Oct 22, 2021, 1:24 PM EDT

I'm getting close to requesting review on my main CL. There will then be a couple follow-up CLs.

Comment 127 by wanderview@chromium.org on Fri, Oct 22, 2021, 3:44 PM EDT

Here is an (internal) document describing all the changes I intend to make over 3 CLs.

https://docs.google.com/document/d/1KZscujuV7bCFEnzJW-0DaCPU-I40RJimQKoCcI0umTQ/edit?usp=sharing

I put this together because it felt like a lot of context to include in the CLs while the bug is still private.

Comment 128 by wanderview@chromium.org on Mon, Oct 25, 2021, 4:55 PM EDT

The first CL is finally out for review:

https://chromium-review.googlesource.com/c/chromium/src/+/3115917

Comment 129 by Git Watcher on Thu, Oct 28, 2021, 3:20 PM EDT

The following revision refers to this bug:
  https://chromium.googlesource.com/chromium/src/+/da0a6501cf321579bd46a27ff9fba1bb8ea910bb

commit da0a6501cf321579bd46a27ff9fba1bb8ea910bb
Author: Ben Kelly <wanderview@chromium.org>
Date: Thu Oct 28 19:19:49 2021

Fetch: Plumb request initiator through passthrough service workers.

This CL contains essentially two changes:

1. The request initiator origin is plumbed through service workers
   that do `fetch(evt.request)`. In addition to plumbing, this
   requires changes to how we validate navigation requests in the
   CorsURLLoaderFactory.
2. Tracks the original destination of a request passed through a
   service worker. This is then used in the network service to force
   SameSite=Lax cookies to treat the request as a main frame navigation
   where appropriate.

For more detailed information about these changes please see the
internal design doc at:

https://docs.google.com/document/d/1KZscujuV7bCFEnzJW-0DaCPU-I40RJimQKoCcI0umTQ/edit?usp=sharing

In addition, there is some discussion of these features in the following spec issues:

https://github.com/whatwg/fetch/issues/1321
https://github.com/whatwg/fetch/issues/1327

The test includes WPT tests that verify navigation headers and SameSite cookies. Note, chrome has a couple expected failures in the SameSite cookie tests because of the "lax-allowing-unsafe" intervention that is currently enabled. See:

https://source.chromium.org/chromium/chromium/src/+/main:third_party/blink/web_tests/TestExpectations;l=4635;drc=e8133cbf2469adb99c6610483ab78bcfb8cc4c76

~~Bug: 1115847~~,~~1241188~~
Change-Id: I7e236fa20aeabb705aef40fcf8d5c36da6d2798c
Reviewed-on: https://chromium-review.googlesource.com/c/chromium/src/+/3115917
Reviewed-by: Matt Menke <mmenke@chromium.org>
Reviewed-by: Yutaka Hirano <yhirano@chromium.org>
Reviewed-by: Nasko Oskov <nasko@chromium.org>
Reviewed-by: Łukasz Anforowicz <lukasza@chromium.org>
Commit-Queue: Ben Kelly <wanderview@chromium.org>
Cr-Commit-Position: refs/heads/main@{#936029}

[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/renderer/platform/loader/fetch/resource_request.h
[add] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/web_tests/external/wpt/service-workers/service-worker/same-site-cookies.https-expected.txt
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/services/network/public/cpp/url_request_mojom_traits.h
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/services/network/cors/cors_url_loader_unittest.cc
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/content/common/fetch/fetch_request_type_converters.cc
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/services/network/public/cpp/resource_request.h
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/fetch-rewrite-worker.js
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/renderer/core/fetch/fetch_request_data.h
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/net/url_request/url_request_http_job.cc
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/renderer/core/fetch/fetch_manager.cc
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/net/url_request/url_request.cc
[add] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/same-site-cookies-unregister.html
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/renderer/platform/loader/fetch/url_loader/request_conversion.cc
[modify]

https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/public/mojom/fetch/fetch_api_request.mojom
[modify]
https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/services/network/public/mojom/url_request.mojom
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/net/url_request/url_request_unittest.cc
[modify]
https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/renderer/core/fetch/fetch_request_data.cc
[add] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/form-poster.html
[add] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/web_tests/external/wpt/service-workers/service-worker/same-site-cookies.https.html
[add] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/fetch-rewrite-worker.js.headers
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/services/network/cors/cors_url_loader_factory.cc
[add] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/location-setter.html
[modify]
https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/services/network/cors/cors_url_loader_factory_unittest.cc
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/net/url_request/url_request.h
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/services/network/BUILD.gn
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/services/network/cors/cors_url_loader.cc
[modify]
https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/renderer/modules/cache_storage/inspector_cache_storage_agent.cc
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/renderer/core/fetch/request.cc
[modify] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/services/network/url_loader.cc
[add] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/web_tests/external/wpt/service-workers/service-worker/navigation-headers.https.html
[add] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/navigation-headers-server.py
[modify]
https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/services/network/public/cpp/url_request_mojom_traits.cc
[modify]
https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/content/common/background_fetch/background_fetch_types.cc
[add] https://crrev.com/da0a6501cf321579bd46a27ff9fba1bb8ea910bb/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/same-site-cookies-register.html

Comment 130 by wanderview@chromium.org on Thu, Oct 28, 2021, 3:55 PM EDT

Note, this still needs another CL in order to fix redirects.

Comment 131 by Git Watcher on Thu, Oct 28, 2021, 6:05 PM EDT

The following revision refers to this bug:
https://chromium.googlesource.com/chromium/src/+/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a

commit a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a
Author: Alan Screen <awscreen@chromium.org>
Date: Thu Oct 28 22:04:50 2021

Revert "Fetch: Plumb request initiator through passthrough service workers."

This reverts commit da0a6501cf321579bd46a27ff9fba1bb8ea910bb.

Reason for revert: Failure on many bots with the following error message:
The service worker navigation preload request was cancelled before 'preloadResponse' settled. If you intend to use 'preloadResponse', use waitUntil() or respondWith() to wait for the promise to settle.", source: (0)

Original change's description:
> Fetch: Plumb request initiator through passthrough service workers.
>
> This CL contains essentially two changes:
>
> 1. The request initiator origin is plumbed through service workers
>   that do `fetch(evt.request)`. In addition to plumbing, this
>   requires changes to how we validate navigation requests in the
>   CorsURLLoaderFactory.
> 2. Tracks the original destination of a request passed through a
>   service worker. This is then used in the network service to force
>   SameSite=Lax cookies to treat the request as a main frame navigation
>   where appropriate.
>
> For more detailed information about these changes please see the
> internal design doc at:
>
> https://docs.google.com/document/d/1KZscujuV7bCFEnzJW-0DaCPU-I40RJimQKoCcI0umTQ/edit?usp=sharing
>
> In addition, there is some discussion of these features in the following
> spec issues:
>
> https://github.com/whatwg/fetch/issues/1321
> https://github.com/whatwg/fetch/issues/1327
>
> The test includes WPT tests that verify navigation headers and SameSite
> cookies. Note, chrome has a couple expected failures in the SameSite
> cookie tests because of the "lax-allowing-unsafe" intervention that is
> currently enabled. See:
>
>
 https://source.chromium.org/chromium/chromium/src/+/main:third_party/blink/web_tests/TestExpectations;l=4635;drc=e8133cbf2469adb99c6610483ab78bcfb8cc4c76
>
> ~~Bug: 1115847~~,~~1241188~~
> Change-Id: I7e236fa20aeabb705aef40fcf8d5c36da6d2798c
> Reviewed-on: https://chromium-review.googlesource.com/c/chromium/src/+/3115917
> Reviewed-by: Matt Menke <mmenke@chromium.org>
> Reviewed-by: Yutaka Hirano <yhirano@chromium.org>
> Reviewed-by: Nasko Oskov <nasko@chromium.org>
> Reviewed-by: Łukasz Anforowicz <lukasza@chromium.org>
> Commit-Queue: Ben Kelly <wanderview@chromium.org>
> Cr-Commit-Position: refs/heads/main@{#936029}

~~Bug: 1115847~~,~~1241188~~
Change-Id: I3044a6d20de172b4a8ab7e39a9f26191580003fa
No-Presubmit: true

No-Tree-Checks: true
No-Try: true
Reviewed-on: https://chromium-review.googlesource.com/c/chromium/src/+/3251692
Auto-Submit: Alan Screen <awscreen@chromium.org>
Bot-Commit: Rubber Stamper <rubber-stamper@appspot.gserviceaccount.com>
Commit-Queue: Alan Screen <awscreen@chromium.org>
Owners-Override: Alan Screen <awscreen@chromium.org>
Cr-Commit-Position: refs/heads/main@{#936125}

[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/third_party/blink/renderer/platform/loader/fetch/resource_request.h
[delete] https://crrev.com/b91ae55530943cc4c51f30f90a63ce77c65808dd/third_party/blink/web_tests/external/wpt/service-workers/service-worker/same-site-cookies.https-expected.txt
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/services/network/public/cpp/url_request_mojom_traits.h
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/services/network/cors/cors_url_loader_unittest.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/content/common/fetch/fetch_request_type_converters.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/fetch-rewrite-worker.js
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/services/network/public/cpp/resource_request.h
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/third_party/blink/renderer/core/fetch/fetch_request_data.h
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/net/url_request/url_request_http_job.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/third_party/blink/renderer/core/fetch/fetch_manager.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/net/url_request/url_request.cc
[delete] https://crrev.com/b91ae55530943cc4c51f30f90a63ce77c65808dd/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/same-site-cookies-unregister.html
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/third_party/blink/renderer/platform/loader/fetch/url_loader/request_conversion.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/third_party/blink/public/mojom/fetch/fetch_api_request.mojom
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/net/url_request/url_request_unittest.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/services/network/public/mojom/url_request.mojom
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/third_party/blink/renderer/core/fetch/fetch_request_data.cc
[delete] https://crrev.com/b91ae55530943cc4c51f30f90a63ce77c65808dd/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/form-poster.html
[delete] https://crrev.com/b91ae55530943cc4c51f30f90a63ce77c65808dd/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/fetch-rewrite-worker.js.headers
[delete] https://crrev.com/b91ae55530943cc4c51f30f90a63ce77c65808dd/third_party/blink/web_tests/external/wpt/service-workers/service-worker/same-site-cookies.https.html
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/services/network/cors/cors_url_loader_factory.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/services/network/cors/cors_url_loader_factory_unittest.cc
[delete] https://crrev.com/b91ae55530943cc4c51f30f90a63ce77c65808dd/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/location-setter.html

[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/net/url_request/url_request.h
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/services/network/BUILD.gn
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/third_party/blink/renderer/modules/cache_storage/inspector_cache_storage_agent.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/third_party/blink/renderer/core/fetch/request.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/services/network/cors/cors_url_loader.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/services/network/url_loader.cc
[delete] https://crrev.com/b91ae55530943cc4c51f30f90a63ce77c65808dd/third_party/blink/web_tests/external/wpt/service-workers/service-worker/navigation-headers.https.html
[delete] https://crrev.com/b91ae55530943cc4c51f30f90a63ce77c65808dd/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/navigation-headers-server.py
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/services/network/public/cpp/url_request_mojom_traits.cc
[modify] https://crrev.com/a6601b2cf2bb7c0a0ffa3c795a0dbc730ef81d1a/content/common/background_fetch/background_fetch_types.cc
[delete] https://crrev.com/b91ae55530943cc4c51f30f90a63ce77c65808dd/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/same-site-cookies-register.html

Comment 132 by Git Watcher on Fri, Oct 29, 2021, 5:20 PM EDT

The following revision refers to this bug:
https://chromium.googlesource.com/chromium/src/+/2d916566085e4f09bca93021f2b1650ea6237077

commit 2d916566085e4f09bca93021f2b1650ea6237077
Author: Ben Kelly <wanderview@chromium.org>
Date: Fri Oct 29 21:19:29 2021

Reland "Fetch: Plumb request initiator through passthrough service workers."

This is a reland of da0a6501cf321579bd46a27ff9fba1bb8ea910bb

This CL also includes a change to mark the two WPT tests as requiring
long timeout durations. On my fast build machine with an opt build
they take ~5 seconds each to complete and the default timeout is 10
seconds. On slower bots with debug builds its highly likely that these
tests would be marked as timing out. This change gives them a 60 second
timeout instead.

Original change's description:
> Fetch: Plumb request initiator through passthrough service workers.
>
> This CL contains essentially two changes:
>
> 1. The request initiator origin is plumbed through service workers
> that do `fetch(evt.request)`. In addition to plumbing, this
> requires changes to how we validate navigation requests in the
> CorsURLLoaderFactory.
> 2. Tracks the original destination of a request passed through a
> service worker. This is then used in the network service to force
> SameSite=Lax cookies to treat the request as a main frame navigation
> where appropriate.
>

> For more detailed information about these changes please see the
> internal design doc at:
>
> https://docs.google.com/document/d/1KZscujuV7bCFEnzJW-0DaCPU-I40RJimQKoCcI0umTQ/edit?usp=sharing
>
> In addition, there is some discussion of these features in the following
> spec issues:
>
> https://github.com/whatwg/fetch/issues/1321
> https://github.com/whatwg/fetch/issues/1327
>
> The test includes WPT tests that verify navigation headers and SameSite
> cookies. Note, chrome has a couple expected failures in the SameSite
> cookie tests because of the "lax-allowing-unsafe" intervention that is
> currently enabled. See:
>
>
https://source.chromium.org/chromium/chromium/src/+/main:third_party/blink/web_tests/TestExpectations;l=4635;drc=e8133cbf2469adb99c6610483ab78bcfb8cc4c76
>
> ~~Bug: 1115847~~,~~1241188~~
> Change-Id: I7e236fa20aeabb705aef40fcf8d5c36da6d2798c
> Reviewed-on: https://chromium-review.googlesource.com/c/chromium/src/+/3115917
> Reviewed-by: Matt Menke <mmenke@chromium.org>
> Reviewed-by: Yutaka Hirano <yhirano@chromium.org>
> Reviewed-by: Nasko Oskov <nasko@chromium.org>
> Reviewed-by: Łukasz Anforowicz <lukasza@chromium.org>
> Commit-Queue: Ben Kelly <wanderview@chromium.org>
> Cr-Commit-Position: refs/heads/main@{#936029}

~~Bug: 1115847~~,~~1241188~~
Change-Id: Ia26acbdd0d7ce6583d9a44f83ed086708657b8bd
Reviewed-on: https://chromium-review.googlesource.com/c/chromium/src/+/3251368
Reviewed-by: Matt Menke <mmenke@chromium.org>
Reviewed-by: Yutaka Hirano <yhirano@chromium.org>
Reviewed-by: Nasko Oskov <nasko@chromium.org>
Reviewed-by: Łukasz Anforowicz <lukasza@chromium.org>
Auto-Submit: Ben Kelly <wanderview@chromium.org>
Commit-Queue: Ben Kelly <wanderview@chromium.org>
Cr-Commit-Position: refs/heads/main@{#936560}

[modify]
https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/renderer/platform/loader/fetch/resource_request.h
[add] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/web_tests/external/wpt/service-workers/service-worker/same-site-cookies.https-expected.txt
[modify]
https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/services/network/public/cpp/url_request_mojom_traits.h
[modify]
https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/services/network/cors/cors_url_loader_unittest.cc
[modify]
https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/content/common/fetch/fetch_request_type_converters.cc

[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/fetch-rewrite-worker.js
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/services/network/public/cpp/resource_request.h
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/renderer/core/fetch/fetch_request_data.h
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/net/url_request/url_request_http_job.cc
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/net/url_request/url_request.cc
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/renderer/core/fetch/fetch_manager.cc
[add] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/same-site-cookies-unregister.html
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/renderer/platform/loader/fetch/url_loader/request_conversion.cc
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/public/mojom/fetch/fetch_api_request.mojom
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/net/url_request/url_request_unittest.cc
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/services/network/public/mojom/url_request.mojom
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/renderer/core/fetch/fetch_request_data.cc
[add] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/form-poster.html
[add] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/web_tests/external/wpt/service-workers/service-worker/same-site-cookies.https.html
[add] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/fetch-rewrite-worker.js.headers
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/services/network/cors/cors_url_loader_factory.cc
[add] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/location-setter.html
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/services/network/cors/cors_url_loader_factory_unittest.cc
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/net/url_request/url_request.h
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/services/network/BUILD.gn
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/services/network/cors/cors_url_loader.cc
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/renderer/modules/cache_storage/inspector_cache_storage_agent.cc
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/renderer/core/fetch/request.cc
[add] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/web_tests/external/wpt/service-workers/service-worker/navigation-headers.https.html
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/services/network/url_loader.cc
[add] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/navigation-headers-server.py
[modify] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/services/network/public/cpp/url_request_mojom_traits.cc
[modify]

https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/content/common/background_fetch/background_fetch_types.cc
[add] https://crrev.com/2d916566085e4f09bca93021f2b1650ea6237077/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/same-site-cookies-register.html

Comment 133 by wanderview@chromium.org on Wed, Nov 3, 2021, 5:23 PM EDT

Second CL up for review:

https://chromium-review.googlesource.com/c/chromium/src/+/3213310

Comment 134 by Git Watcher on Tue, Nov 9, 2021, 10:49 AM EST

The following revision refers to this bug:
https://chromium.googlesource.com/chromium/src/+/579df7b562fd2a85591e44fd314a1710c93e6901

commit 579df7b562fd2a85591e44fd314a1710c93e6901
Author: Ben Kelly <wanderview@chromium.org>
Date: Tue Nov 09 15:48:30 2021

Fetch: Plumb navigation redirect chain through service workers

Navigation redirection works differently than normal redirection.
Navigation requests are made using "manual" redirect mode which means
the redirect is not immediately followed. Instead the redirect location
is handed back up to the NavigationURLLoaderImpl which then manually
follows the redirect. This results in a new request being sent for each
step in the redirect chain.

This CL plumbs the redirect chain information from
NavigationURLLoaderImpl down through each request so it can be included
with requests proxied by a passthrough service worker.

For more detailed information about these changes please see the
internal design doc at:

https://docs.google.com/document/d/1KZscujuV7bCFEnzJW-0DaCPU-I40RJimQKoCcI0umTQ/edit?usp=sharing

We have rough consensus to make this change in this spec issue:

https://github.com/whatwg/fetch/issues/1335

Note, this CL includes some expected test failures. These are due to
the "lax-allowing-unsafe" intervention that is currently enabled. See:

https://source.chromium.org/chromium/chromium/src/+/main:third_party/blink/web_tests/TestExpectations;l=4635;drc=e8133cbf2469adb99c6610483ab78bcfb8cc4c76

~~Bug: 1115847~~,~~1241188~~
Change-Id: I2a2a17639e0bec3222684e0d444d6d98a21402ed
Reviewed-on: https://chromium-review.googlesource.com/c/chromium/src/+/3213310
Commit-Queue: Ben Kelly <wanderview@chromium.org>
Reviewed-by: Nasko Oskov <nasko@chromium.org>
Reviewed-by: Matt Menke <mmenke@chromium.org>

Reviewed-by: Yutaka Hirano <yhirano@chromium.org>
Cr-Commit-Position: refs/heads/main@{#939851}

[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/renderer/platform/loader/fetch/resource_request.h
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/web_tests/external/wpt/service-workers/service-worker/same-site-cookies.https-expected.txt
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/services/network/public/cpp/url_request_mojom_traits.h
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/services/network/cors/cors_url_loader_unittest.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/content/common/fetch/fetch_request_type_converters.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/services/network/public/cpp/resource_request.h
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/renderer/core/fetch/fetch_request_data.h
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/renderer/core/fetch/fetch_manager.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/net/url_request/url_request.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/renderer/platform/loader/fetch/url_loader/request_conversion.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/public/mojom/fetch/fetch_api_request.mojom
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/services/network/public/mojom/url_request.mojom
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/net/url_request/url_request_unittest.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/renderer/core/fetch/fetch_request_data.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/form-poster.html
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/web_tests/external/wpt/service-workers/service-worker/same-site-cookies.https.html
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/services/network/cors/cors_url_loader_factory.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/location-setter.html
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/services/network/cors/cors_url_loader_factory_unittest.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/web_tests/external/wpt/service-workers/service-worker/resources/redirect.py
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/net/url_request/url_request.h
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/renderer/core/fetch/request.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/renderer/modules/cache_storage/inspector_cache_storage_agent.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/third_party/blink/web_tests/external/wpt/service-workers/service-worker/navigation-headers.https.html

[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/services/network/url_loader.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/content/browser/loader/navigation_url_loader_impl.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/services/network/public/cpp/url_request_mojom_traits.cc
[modify] https://crrev.com/579df7b562fd2a85591e44fd314a1710c93e6901/content/common/background_fetch/background_fetch_types.cc

Comment 135 by wanderview@chromium.org on Tue, Nov 9, 2021, 10:54 AM EST

Status: Fixed (was: Assigned)
Labels: -M-94 M-98

I believe this is now fixed. The first CL landed in M97 and I believe the second CL landed in M98. I would prefer not to merge anything here because the CLs are huge, but we could maybe merge the second CL to M97 since the branch point was recent.

Comment 136 by wanderview@chromium.org on Tue, Nov 9, 2021, 10:55 AM EST

FWIW, I'm also still working on spec pull requests to reflect the new behavior.

Comment 137 by sheriffbot on Tue, Nov 9, 2021, 12:42 PM EST

Labels: reward-topanel

Comment 138 by sheriffbot on Tue, Nov 9, 2021, 1:41 PM EST

Labels: -Restrict-View-SecurityTeam Restrict-View-SecurityNotify

Comment 139 by amyressler@google.com on Wed, Nov 17, 2021, 2:31 PM EST

Labels: -reward-topanel reward-unpaid reward-3000

*** Boilerplate reminders! ***
Please do NOT publicly disclose details until a fix has been released to all our users. Early public disclosure may cancel the provisional reward. Also, please be considerate about disclosure when the bug affects a core library that may be used by other products. Please do NOT share this information with third parties who are not directly involved in fixing the bug. Doing so may cancel the provisional reward. Please be honest if you have already disclosed anything publicly or to third parties.
Lastly, we understand that some of you are not interested in money. We offer the option to donate your reward to an eligible charity. If you prefer this option, let us know and we will also match your donation - subject to our discretion. Any rewards that are unclaimed after 12 months will be donated to a charity of our choosing.

Please contact security-vrp@chromium.org with any questions.
****

Comment 140 by amyressler@chromium.org on Wed, Nov 17, 2021, 3:53 PM EST

Congratulations -- the VRP Panel has decided to award you $3000 for this report. A member of our finance tam will be in touch soon to arrange for payment. Thank you for reporting this issue to us!

Comment 141 by holeg...@gmail.com on Wed, Nov 17, 2021, 4:00 PM EST

@amyressler thank you so much! Also like to extend a BIG THANK YOU to everyone involved in this from managing the issue, to spec discussion and clarification and eventual patch! Your all amazing :-)

Comment 142 by amyressler@google.com on Thu, Nov 18, 2021, 3:50 PM EST

Labels: -reward-unpaid reward-inprocess

Comment 143 by sheriffbot on Sat, Dec 11, 2021, 2:12 PM EST

Labels: Merge-NA-98

Not requesting merge to dev (M98) because latest trunk commit (939851) appears to be prior to dev branch point (950365). If this is incorrect, please replace the Merge-NA-98 label with Merge-Request-98. If other changes are required to fix this bug completely, please request a merge if necessary.

For more details visit https://www.chromium.org/issue-tracking/autotriage - Your friendly Sheriffbot

Comment 144 by amyressler@chromium.org on Tue, Jan 4, 2022, 12:32 PM EST

Labels: Release-0-M97

Comment 145 by amyressler@google.com on Tue, Jan 4, 2022, 1:34 PM EST

Labels: CVE-2022-0111 CVE_description-missing

Comment 146 by yoavweiss@chromium.org on Thu, Jan 13, 2022, 5:13 AM EST

Cc: yoavweiss@chromium.org merewood@chromium.org

Comment 147 by sheriffbot on Tue, Feb 15, 2022, 1:39 PM EST

Labels: -Restrict-View-SecurityNotify allpublic

This bug has been closed for more than 14 weeks. Removing security view restrictions.

For more details visit https://www.chromium.org/issue-tracking/autotriage - Your friendly Sheriffbot

Comment 148 by kidi...@gmail.com on Wed, Apr 20, 2022, 7:14 AM EDT

Since the restriction removed, I've been able to finally follow up on the whole issue (I reported ~~bug 1115847~~) and I see many conversations happened here. Great work all!

Comment 149 by amyressler@chromium.org on Fri, Jul 29, 2022, 5:36 PM EDT

Labels: -CVE_description-missing CVE_description-submitted