Delayed Clipboard Rendering

Github issue: https://github.com/w3c/editing/issues/439
Explainer:
https://github.com/MicrosoftEdge/MSEdgeExplainers/blob/main/DelayedClipboard/DelayedClipboardRenderingExplainer.md

Meeting (11/30)

Attendees:
Evan Stade (Google)
Anne van Kesteren (Apple)
Anupam Snigdha (Microsoft)
Sanket Joshi (Microsoft)
Wenson Hsieh (Apple)
Megan Gardner (Apple)
Christine Hollingsworth (Google)
Thomas Steiner (Google)
Ajay Rahtekar (Google)
Ayu Ishii (Google)
Olli Pattay(Mozilla)
Simon Pieters(Mozilla)

Scribe: Dan Clark (Microsoft)

Anupam: Want to discuss privacy concerns related to delayed rendering of custom formats. When source app delay renders format, registers callback. When the user pastes into an app that tries to read it, the system clipboard calls back to the source app which triggers a callback. In the callback the source app populates the data. For web custom format, random site can register callback for custom format. Don't need to produce data at copy time. When the callback is triggered during paste, the source app can determine that the paste is happening. <gives example with Photoshop custom format>.
The source app can then determine that the user is pasting into Photoshop.
Proposed mitigation in issue is restricting delayed rendering to only builtin formats. That's really restrictive for API since web custom formats can have high fidelity content. Source app doesn't know destination app, doesn't want to spend cycles to generate the data for custom format if not needed. Primary use case is to support delayed rendering for custom formats, so trying to find middleground to reduce fingerprinting concerns. Photoshop can advertise photoshop format, figma can advertise figma format, etc. If we allow site to advertise more than 1 custom format then there are a couple cases. One case where site is malicious, paste will fail if page doesn't generate data. Or will paste if page does generate data. Fingerprinting is related to first case where pase fails, malicious site is able to detect user paste activity. In 1st case where paste

fails, will be bad experience for user. User would only try as one time thing. Try to paste, it fails. User won't try again. So in malicious site, do we want to restrict number of formats?
Simon: Don't understand why only problematic if paste fails.
Anupam: source app already knows web custom format is intended for their app. Already know ecosystem.
Simon: That's part of the issue. Don't want the site to know when a given app is installed or not.
Anne: not sure you're thinking about failing correctly. Source app can write multiple formats at the same time. Upon paste, maybe custom one gets read, maybe not. In case where it does, they have signal user is using particular app. When it doesn't, paste doesn't necessarily fail because falls back to text/plain.
Anupam: Native apps try to read the most preferred formats first. If there is custom format, will try to read that first.
Anne: <missed>
Anupam: If plaintext read and delay rendered, can't determine destination app.
Anne: You're saying paste fails therefore user will stop, but it's not clear that the paste will fail.
Anupam: Talking about custom formats only here.
Anne: It can populate builtin formats as well so paste will always succeed.
Anupam: Builtin formats aren't a concern here.
Wenson: With builtins, not a fingerprinting concern because user knows what's being pasted. With custom formats there's more obscure data that can be written.
Anupam: Fingerptinging concern is for custom formats
Simon: Not obvious can't fingerprint with standard formats
Wenson: Referring to urls, text/plain, text/html, and image/png. IN all these cases in Webkit, with HTML if thing writing data writes extra info in HTML comment, we'll sanitize it when writing to clipboard. Only visible content preserved. For all builtin types can sanitize to preserve privacy.
Simon: Important thing to me is you're talking about only those 4 types.
Evan: In Chromium, <missed> Not removing all comments. There will be signs in HTML, don't think sanitization scrubs all info about where it's coming from.
Wenson: We try to. Really only visible text.
Sanket: Concern raised here is that w/ delayed rendering, source app gets callback. Info getting exposed is where user pasted. If malicious site exposes format for e.g. photoshop, they can know that user pasted into photoshop.
Anne: Secondary concern: is now communication channel between source and dest app.
Simon: That's the point of the feature.
Sanket: When user pastes, the app has to produce the payload. The point Anupam is making is why this is unique to web custom formats is that when site's rendering native format, may be read by N different destinations. Problem with custom formats is they may be very app/ecosystem specific. Goal is can we find mitigation that enables API to be usable with custom formats since it's important for them to be delay renderable. While still minimizing fingerprinting risk.
Wenson: Use case: paste from web version to native version of same app. Know the app itself owns the website too. So no risk there since it's the same owner. Can we use that to allow delay rendering in that case, if it's the same associated domain for the destination.
Sanket: Discussed this before briefly. Question is what's the mechanism.

Evan: That's important question but also sounds like reduction of interop. In example we're using here, some other version of Office can't interop with most popular version. Not a good direction.
Johannes: This was the concern raised last time, could shut out smaller competitors.
Anne: Like the same origin limitation which makes it not that useful.
Anupam: Need to know what the payload should be, can't just be random bytes. Or paste fails in dest app. Paste fails if you don't populate the data, or dest app doesn't know what format should be, can't parse it. Can't just populate random bytes.
Sanket: Case that Johannes and others are taking about, there's a format you want to use across an ecosystem. E.g. Office/text format used by multiple editors. Yes the mitigation we're discussing here would limit that. But don't think use cases heard from web devs are about that. Use cases are in the realm of what Wenson described. Custom formats used for copy/paste within that ecosystem.
Anupam: If other apps, let's say office publishes web custom format publicly and want other apps to read it, can move it from custom format to builtin bucket.
Johannes: Concern is if MSFT word doesn't want competitors, can close it. No one else can create interoperable thing with Word.
Anupam: By design. Not mean to act like built in. Great if they all know what it is, but if they don't then it's by design they won't be able to read it. If there's web text/html format, and has HTML content and metadata, and Office/Photoshop wants to publish this format and want other apps to read it. Can describe the format
Johannes: Is that currently the case? If I create Word competitor, am I initially shut out to receive paste from Word or can I try to write a parser for Word format?
Anupam: There's no restriction. Any app can read it. App can add enterprise policy to restrict but that's out of scope for this discussion. So yes, in general clipboard does not restrict what app can read. If you don't know what data is present in custom format, wouldn't be able to process it. Can use HTML parser but if it has metadata the parsing will fail. Fingerprinting concern is that with web custom formats, if it's restricted to a particular ecossytem/set of apps, in last meeting we proposed mitigation of restricting delay rendering to 5 or small number. Source app can't cast wider net and advertise all custom formats on clipboard and hope to detect one.
Anne: Even for single format don't think we have something that works yet. Fingerprinting concern and concern you can't sanitize it. Don't see a viable way around those other than same origin/same-app-bounds solutionn. Neither seem great because seem to prevent other origins from competing with yours.
Evan: I still think contents of HTML can't be effectively sanitized. Uses this font size, this layout, nests table elements in this way…
But that info only goes in one direction when you're pasting into web app. This can go the other way, which is potentially bad. That said you know only the one app. Fingerprinting is to identify user base on which apps are installed but you only get info about one and it's fuzzy.
Simon: Have options for targeting many formats. Build up search tree, with couple of pastes get a good idea of where paste is going, user habits.
Evan: Narrows the vector if you have to convince user to copy/paste. Is sign of user trust.
In Chromium, for certain ways in case you use async API there's a permission prompt. We don't love these but it's a way to get the user to make a trust decision. User won't just grant it to totally

malicious site. If it's their chosen office software..what kind of site are we concerned about for fingerprinting?
Olli: Company A's office products may want to know if you're pasting to Company B's products., might be useful for Company A's advertising.

Sanket: Copy-paste should succeed in cases where the formats are expected to work. It's very hard for different apps to provide legit copy-paste with formats that they are not aware of.

Olli: Not talking about failing. Company A learns about user paste activity and advertises things.

Sanket: Primary goal is for the paste to succeed.
Anne: Why can't it learn how to write the format?

Sanket: It's hard to do.
Anne: That is not really tricky.
Evan: This is tangential. This is for delay web custom formats not built-in formats. Delay formats are important for sophisticated apps. We are only allowing it for a few web custom formats. If we can't provide that then why do we need delay rendering?
Sanket: It's costly to generate the payload so we need delay rendering even if it's just for one format. For photoshop, the cost for generating format is wasted if it's not pasted in photoshop. Web custom formats can be used to extend support for more formats. Target apps

https://mozilla.zoom.us/j/92378268466?pwd=OHRQSnRycXd3VXk5L3NhMlBIYVpVdz09

Olli: These issues are there. Even if you know one external app then you know a lot. Copy-pasting from company A to B products can happen and advertisements can be targeted to the user.

Evan: If you are typing in a website then they already know a lot about you. Weigh the utility for delay rendering. Are we concerned about malicious website?

Olli: That is one part. Even if there is one app then they can exploit a zero day vulnerability.

Evan: Most attacks cast a wider net. If there is a zero day then target everyone.

Anne: Target everyone then it gets noticed.

Sanket: The thing is fingerprinting not security. You can write malicious data to the clipboard. Callback is what indicates the source app. Web devs are asking for it so there is utility for delay rendering. Trade off is worthy.

Wenson: Is the use case for 1 or 2 formats time intensive?

Sanket: Only handful of custom types. Eco system specific formats are handful.

Wenson: Can be mitigated by asking for all delay formats including the builtin types during paste.

Sanket: All custom or builtin formats?

Wenson: Either would have the advantage of not leaking types?

Sanket: It would kill the benefits of delay rendering. Custom format is pasted then we would trigger callback for both custom and builtin formats which is a waste of resources.
If it's just the custom formats, then it might be ok?

Anne: Not quite sure how that would mitigate it? Concern is with custom format (one or more). There is a secondary concern that we can't santize.
Wenson: The only advantage with an all at once approach is you can't cast a wider net. Can only target one specific app. Doesn't address Anne's concern though.
Anne: Not sure about custom formats.
Wenson: Same origin and same app is fine. No privacy risk.
Evan: All of the things are concerned with all formats?
Anne: We are concerned about custom formats.
Evan: Different concern.
Anne: It's a concern with custom format.
Evan: Not sure what to do with that concern. We're trying to make progress with delay rendering.
Anne: When I proposed custom formats I didnt have that concern in my mind.
Evan: There is a proposal for reading unsanitized html. Is that something you thought about?
Wenson: Only allow for same origin. Sanitize for cross origin.
Evan: That is tangential but was just curious.
Sanket: Where are we at? Fingerprinting is unique to web custom format. What is the objection?
Simon: If we add more native formats then it might have the same issue
Anne: Monitor user habits but delayed clipboard rendering of builtin formats is fine.