Cross-Origin-Opener-Policy reporting API

Author: clamy@chromium.org
Status: Final
Last modified: 2020-02-06

Objective

Provide a report-only mode to help developers deploy CrossOriginOpenerPolicy on their websites. The reporting API should surface useful information about potential breakages to the developers. The reporting API should not introduce side channels that bad actors could exploit.

Background

CrossOriginOpenerPolicy

With CrossOriginOpenerPolicy, pages can restrict their browsing context group to same-origin top-level documents. The browser will place a COOP document in a different browsing context group from its opener when they are cross-origin. The opener document cannot access the opened document through the WindowProxy.

Even without full SiteIsolation, browsers open the new COOP document in a new process as the opener document cannot access it. Thus, COOP documents only share processes with same-origin top-level documents. This mitigates side-channel attacks from other top-level documents in the browsing context group. The browser will enforce setting COOP to have access to powerful new APIs that create side channels (eg. process memory measurement API).

The deployment of COOP on a document risks breaking the page itself and any page that opens it. The communication between pages will break when their COOP policy and/or origins do not match. The document that deploys COOP can no longer interact with documents it opens. Likewise, communication from other pages through WindowProxy is impossible.

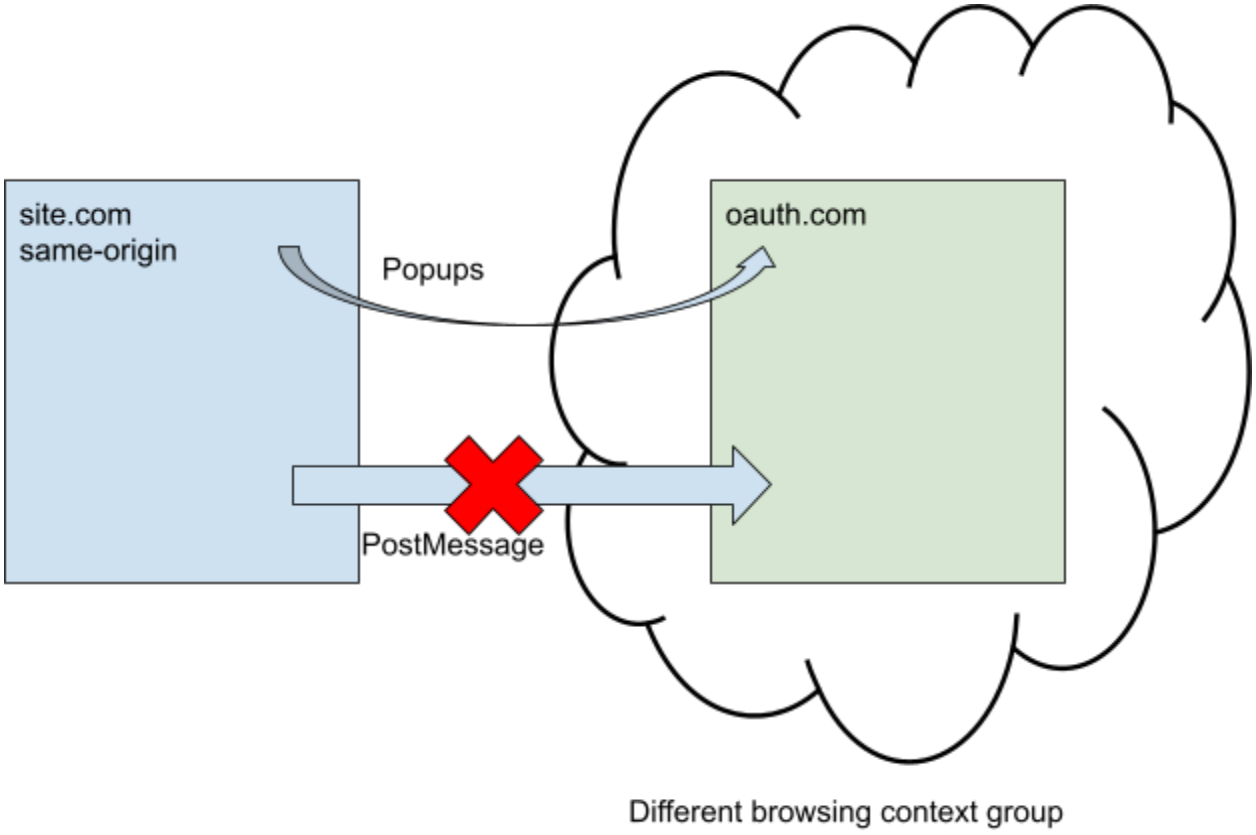
COOP brings security benefits for websites at the cost of breakages when deployed. Like CSP, there is real value for developers to have a reporting API for COOP. Websites can deploy COOP in report-only mode, to get reports of what might break when COOP is enforced.

Breakages due to COOP

Let’s have a look at what can break when deploying COOP.

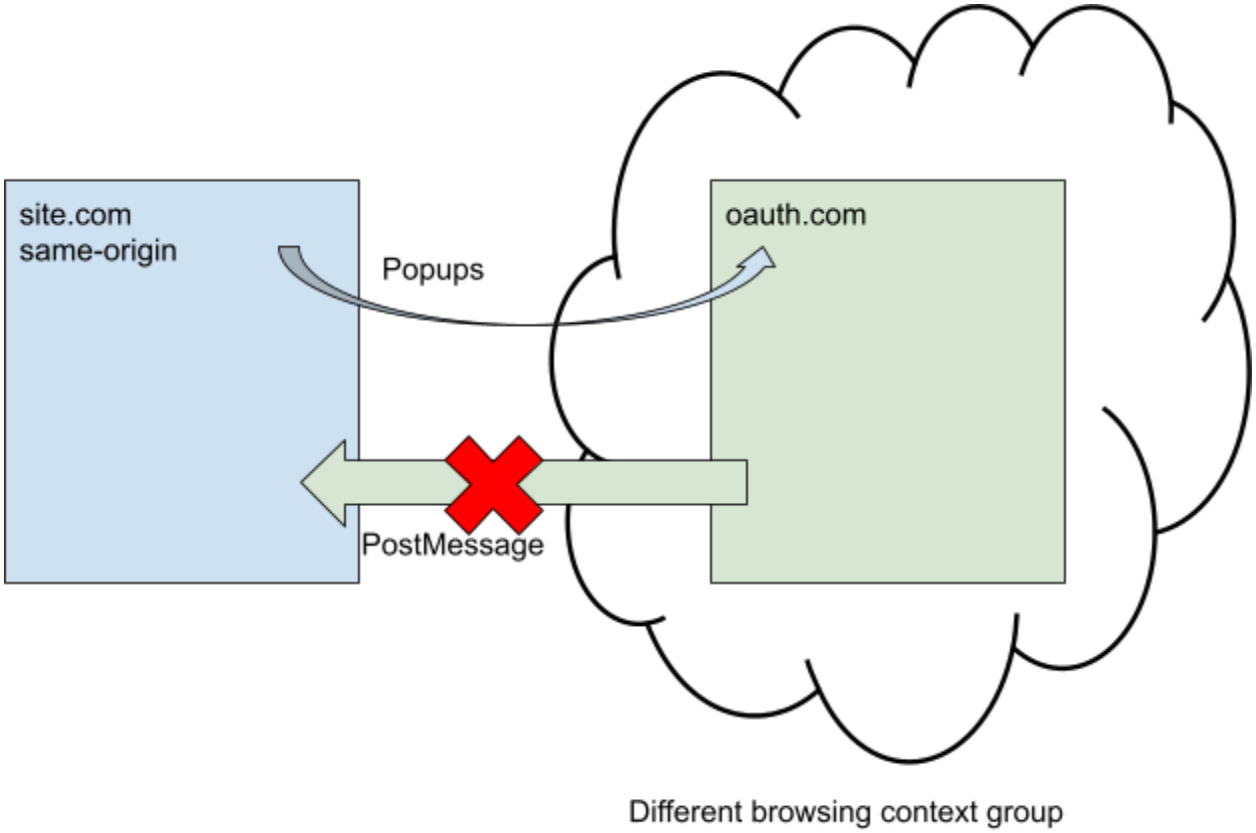
Case 1: COOP document access its opened windows

Let's have a look at an OAuth flow. Imagine a page sets COOP and opens a cross-origin OAuth popup . If COOP is ‘same-origin’, the popup will be in a different browsing context group. If the page tries to access its popup using WindowProxy, it won’t work.



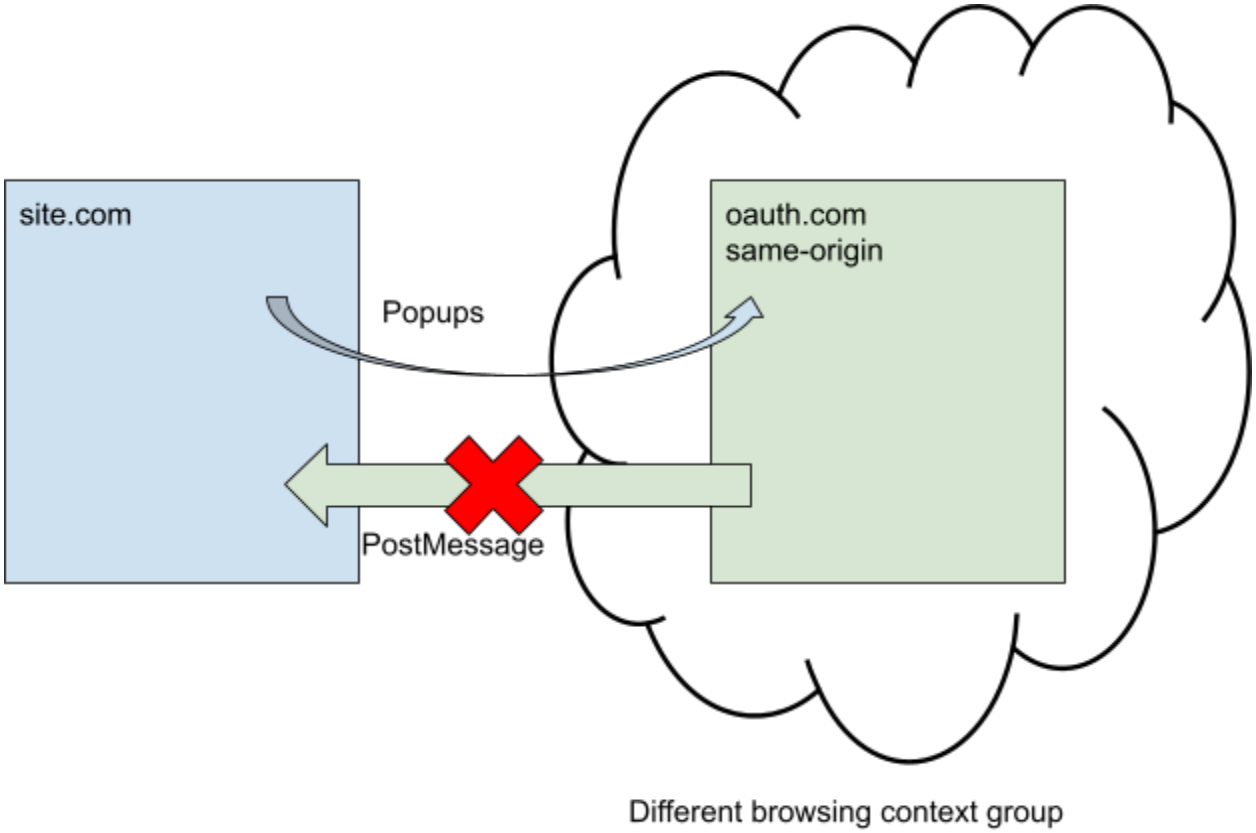
Case 2: COOP document accessed by its opened windows

Looking at the OAuth flow again. If COOP is ‘same-origin’, the page and its OAuth popup will be different browsing context groups. The OAuth popup will not be able to communicate with the page through WindowProxy. It cannot tell the page the user logged in.



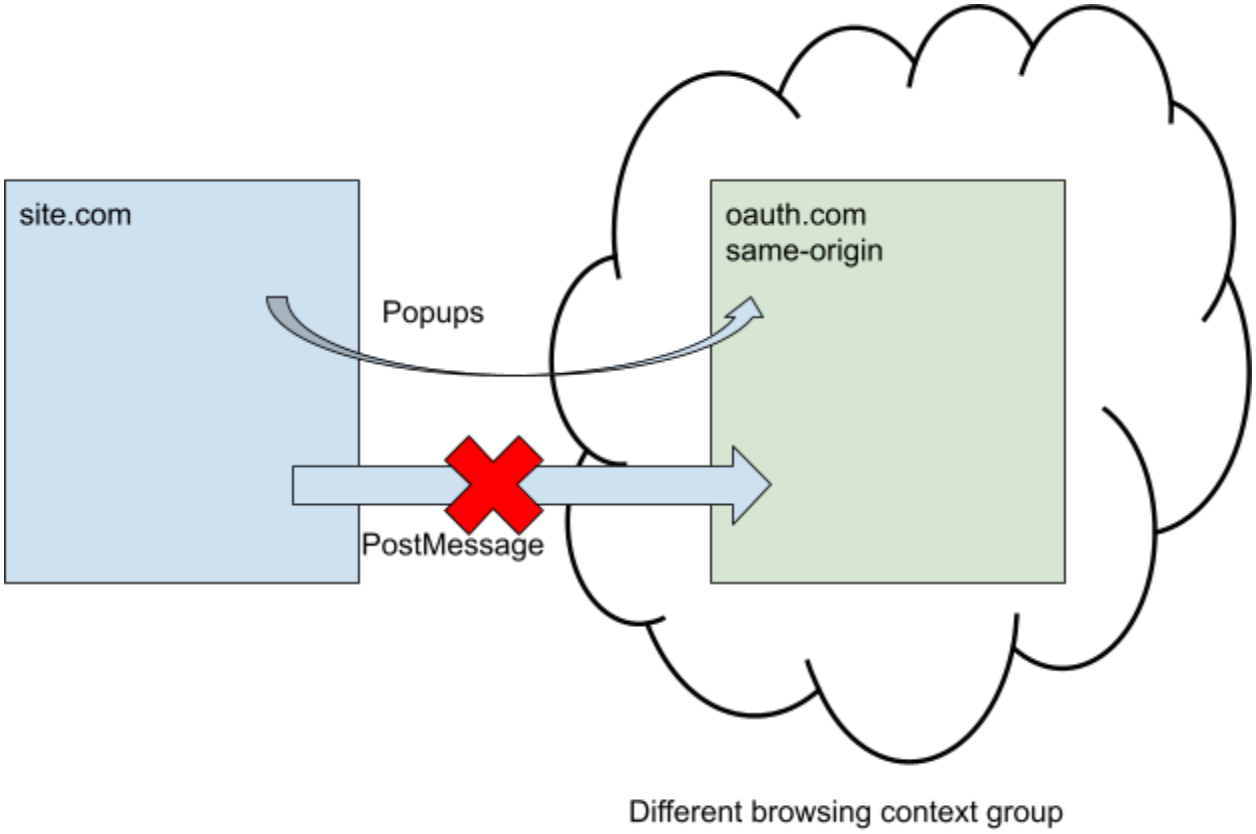
Case 3: COOP document accesses its opener

Let's get back to an OAuth flow, and imagine the OAuth popup sets COOP. If COOP is ‘same-origin’ or ‘same-origin-allow-popups’, the popup and its opener will be in different browsing context groups. So the OAuth popup cannot access its opener using WindowProxy.



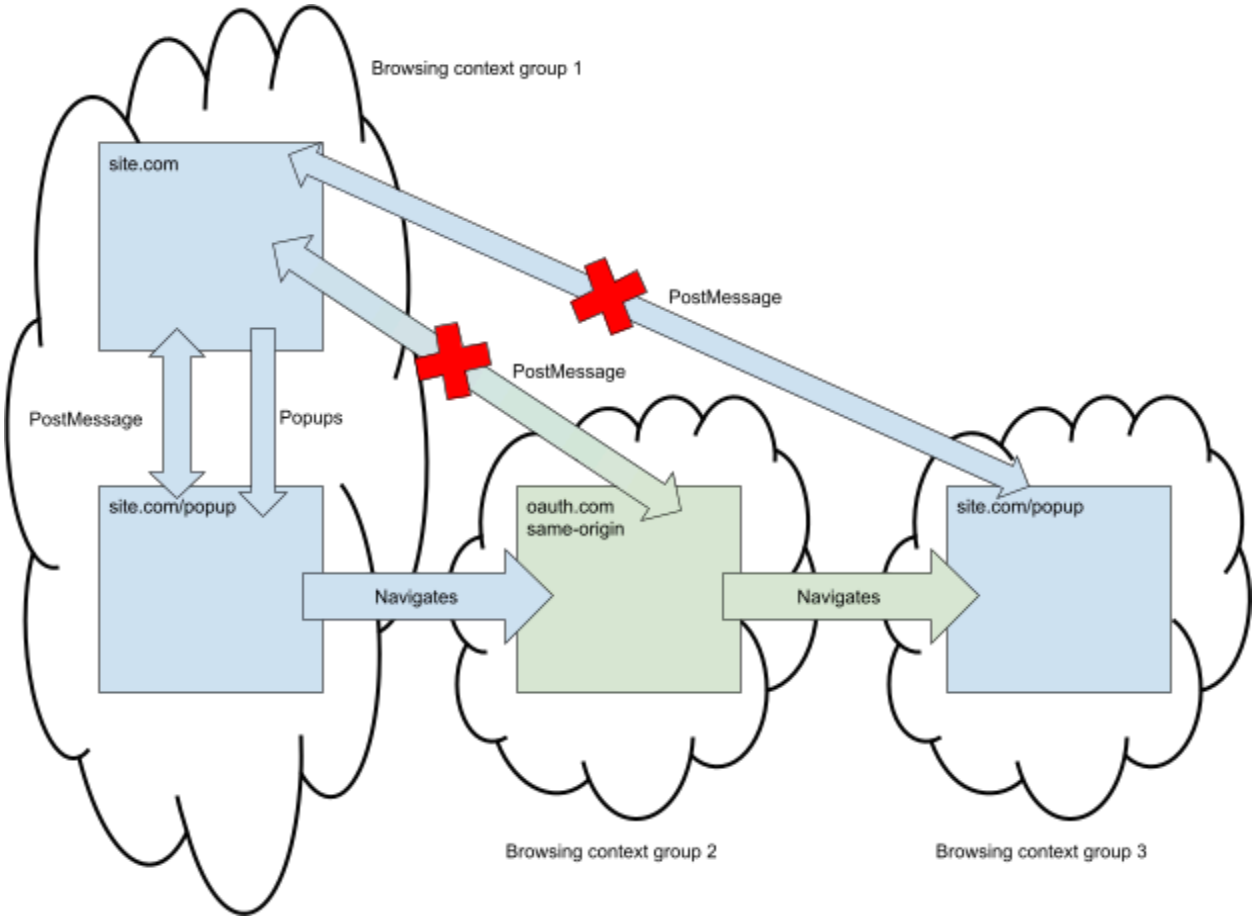
Case 4: COOP document accessed by its opener

Let's take the same OAuth flow, where the OAuth popup sets COOP. If COOP is ‘same-origin’ or ‘same-origin-allow-popups’, the popup and its opener will be in different browsing context groups. So the opener cannot access the OAuth popup using WindowProxy.



Case 5: Navigation to COOP document in a window with an opener
Let's look at a more complex OAuth flow. The website opens a same-origin popup. The popup navigates cross-origin to an OAuth provider. Once the user has logged in, the popup navigates back to the website. Then, the popup notifies its opener that the user has logged in.

If the OAuth provider login page sets COOP, navigating to it puts the popup in a different browsing context group. When the popup navigates back to the site, it is still in a different browsing context group. The popup cannot communicate with its opener using WindowProxy.



Side-channel leaks risks of the COOP reporting API

We would like to report all the blocked cross-window accesses to the page with COOP reporting. This would include the origin of the cross-origin window and the API used. But this would provide the page with information it currently doesn't have. Nefarious actors could exploit the reporting API. Thus, they would gain information on other cross-origin documents. So we need to restrict the information given in the COOP reporting API to avoid introducing new side-channels.

URLs/origins the document already possesses today are safe to include in the reporting API:

- Referrer when reporting issues with the opener window.
- Initial URL of an opened window when reporting issues with opened windows. Note that this URL may have changed due to server redirects - only the initial URL is safe to include.

Reporting an access violation from the COOP document is safe. This includes reporting the API involved in the access violation and a script sample. This is information the COOP document already has.

Reporting an access violation to the COOP document gives the document more information. In particular if it includes the API used that triggered the violation. This should be considered carefully.

Detailed design

Header

To enable COOP reporting we use the following headers in conjunction:

- Cross-Origin-Opener-Policy: defines the CrossOriginOpenerPolicy that the browser will enforce. By default, this is 'unsafe-none'. We add a 'report-to' member to the header to specify an endpoint to send reports to.
- Cross-Origin-Opener-Policy-Report-Only: the CrossOriginOpenerPolicy value of this header is not enforced, only reports are generated for it. The possible values for this header are the same as for the Cross-Origin-Opener-Policy header. This header also has a 'report-to' member to specify the endpoint where the reports should be sent.

Examples of usage:

1. `Cross-Origin-Opener-Policy: same-origin` -> enforces a CrossOriginOpenerPolicy of same-origin
2. `Cross-Origin-Opener-Policy: same-origin; report-to=https://foo` -> enforces a CrossOriginOpenerPolicy of same-origin and generates violation reports for a policy of same-origin that are sent to https://foo.
3. `Cross-Origin-Opener-Policy: same-origin-allow-popups; report-to=https://foo` and `Cross-Origin-Opener-Policy-Report-Only: same-origin; report-to=https://foo` -> enforces a CrossOriginOpenerPolicy of same-origin-allow-popups and generates violation reports for policies of same-origin-allow-popups and same-origin that are sent to https://foo.
4. `Cross-Origin-Opener-Policy: same-origin-allow-popups` `Cross-Origin-Opener-Policy-Report-Only: same-origin` -> enforces a CrossOriginOpenerPolicy of 'same-origin-allow-popups' but does not send violation reports because no endpoint for the reports were provided.
5. `Cross-Origin-Opener-Policy-Report-Only: same-origin; repor-to=https://foo` -> enforces a CrossOriginOpenerPolicy of unsafe-none and and generates violation reports for a policy of same-origin that are sent to https://foo.
6. `Cross-Origin-Opener-Policy: report-to=https://foo` -> generates violation reports when the document tries to access other documents which have COOP.

Reporting policy violations coming from the document

Here, we report cross-window access from the COOP document that COOP prevents/would prevent. This corresponds to cases 1 and 3 from the background part of this doc. We are not

reporting that windows are/would be in different browsing context groups due to COOP. We only generate a violation report when the document tries to access one of these windows. It doesn't/wouldn't work because of COOP.

When this happens, we generate a JSON file with the following fields:

- `accessed-document-uri`: an identifier of the window where the access was blocked
    - For windows with top-level same-origin document, this is the URI of the top level document.
    - For a cross-origin opener window, this is the referrer.
    - For cross-origin windows opened by the document, this is the initial URL requested by the document.
    - For other cross-origin windows, it is empty.
- `disposition`: either "`enforce`" or "`reporting`" depending on whether the `Cross-Origin-Opener-Policy` header or the `Cross-Origin-Opener-Policy-Report-Only` header is used.
- `document-uri`: the URI of the COOP document.
- `effective-policy`: the policy whose enforcement caused the violation.
- `script-sample`: the first 40 bytes of inline script or event handler that caused the violation.
- `violation-type:` "`access-from-document`"

Reporting policy violations coming from external access to the document

We report cross-window access to the COOP document that COOP prevents/would prevent. This corresponds to cases 2 and 4 from the background part of this doc. We are not reporting that windows are/would be in different browsing context groups due to COOP. We only generate a violation report when one of these windows tries to access the document. It doesn't/wouldn't work because of COOP.

When the violation happens, we generate a JSON file with the following fields:

- `accessed-document-uri`: an identifier of the window whose access was blocked
    - For windows with top-level same-origin document, this is the URI of the top level document.
    - For a cross-origin opener window, this is the referrer.
    - For cross-origin windows opened by the document, this is the initial URL requested by the document.
    - For other cross-origin windows, it is empty.
- `disposition`: either "`enforce`" or "`reporting`" depending on whether the `Cross-Origin-Opener-Policy` header or the `Cross-Origin-Opener-Policy-Report-Only` header is used.
- `document-uri`: the URI of the COOP document.
- `effective-policy`: the policy whose enforcement caused the violation.
- `blocked-api`: the API which triggered the violation (eg. "PostMessage").

- `violation-type:` “`access-to-document`”

This reveals extra information to the document using the COOP reporting API. The document may now know that its opener or a window it opened tried to access it using a specific API. Yet, we do believe that it is fine for a document to know that other documents tried to access it using specific APIs. We plan on introducing a new general API to audit cross-origin document accesses.

Note that the COOP violation does not happen in the document that enabled the reporting API. We do not guarantee synchronization between the two documents. The COOP document could have been closed before the violation is reported. In that case, the reporting API will not generate a JavaScript event in the COOP document. Instead, the reporting API will only send a report to the reporting endpoint.

Reporting browsing context group switch when navigating a document with an opener

We report browsing context group switches from navigations to/from COOP documents in **existing windows with an opener**. This corresponds to case 5 from the background part of this doc. Navigations that cause browsing context switches might cause issues with follow-up navigations. Other documents our window navigate to will no longer be able to access the opener of our window.
When this happens, we generate a JSON file with the following fields:

- `disposition`: either "`enforce`" or "`reporting`" depending on whether the `Cross-Origin-Opener-Policy` header or the `Cross-Origin-Opener-Policy-Report-Only` header is used.
- `document-uri`: the URI of the COOP document.
- `effective-policy`: the policy whose enforcement caused the violation.
- `navigation-uri`: the URL we were trying to navigate to/from.
    - When the URL is same-origin it is the full URL.
    - For cross-origin navigations to the COOP document, use the referrer URL.
    - For cross-origin navigations from the COOP document, use the initial URL requested by the document.
- `violation-type:` “`navigation-to-document`” or “`navigation-from-document`” depending on whether the browsing context group switch happened when navigating to the document or from the document. Note that if it happens both times, we should get two reports.

If the violation happens when navigating away from the COOP document, we only know about it when we commit the new document. When this happens, the COOP document is unloading or deleted already. In that case, the reporting API will not generate a JavaScript event in the COOP document. Instead, the reporting API will only send a report to the reporting endpoint.

Reporting policy violations to a non-COOP document

Non-COOP documents can enable the COOP reporting API by setting the header "Cross-Origin-Opener-Policy-Report-Uri". In that case, we will only report blocked accesses to other documents that have COOP. We will not generate violations when the document accessed a document with report-only COOP.
When an access from the document to a COOP document is blocked, we generate a JSON file with the following fields:

- `blocked-uri`: an identifier of the window where the access was blocked
    - For windows with top-level same-origin document, this is the URI of the top level document.
    - For a cross-origin opener window, this is the referrer.
    - For cross-origin windows opened by the document, this is the initial URL requested by the document.
    - For other cross-origin windows, it is empty.
- `document-uri`: the URI of the non-COOP document that enabled reporting.
- `script-sample`: the first 40 bytes of inline script or event handler that caused the violation.
- `violation-type:` “`non-coop-reporting`”