



Dealing with Cross-Site Attacks

German OWASP Day 2024
Frederik Braun



HTML Design Principles

W3C Working Draft

When considering changes to legacy features or behavior...

... the benefit of the proposed change should be weighed against the likely **cost of breaking content** [...]

In some cases, **it may be desirable to make a nonstandard feature or behavior part of the conforming language**, if it satisfies a valid use case.



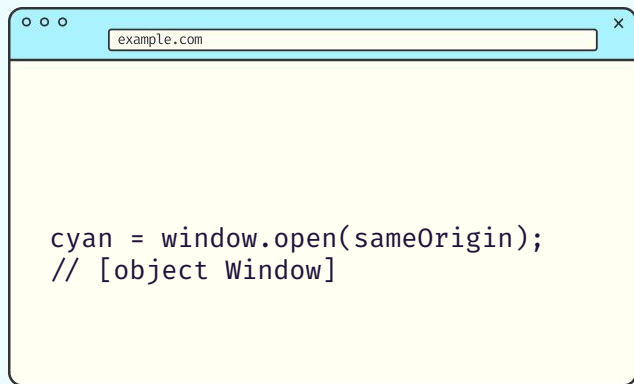
XMLHttpRequest

**The real world
does not reflect
best practices**

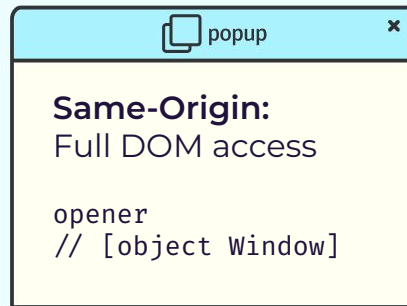
xsleaks

<https://xsleaks.dev>

popups

A diagram of a browser window with a light blue title bar containing three small circles on the left and an 'x' on the right. The address bar shows 'example.com'. The main content area is yellow and contains the following code:

```
cyan = window.open(sameOrigin);  
// [object Window]
```

A diagram of a popup window with a light blue title bar containing a document icon on the left and an 'x' on the right. The title bar text is 'popup'. The main content area is yellow and contains the following text:

```
Same-Origin:  
Full DOM access  
  
opener  
// [object Window]
```

Browsing Context Group



cross-origin popups

```
example.com  
  
yellow = window.open(yellowOrigin)  
// [object Window]  
  
// the frames are leaking  
yellow.length;  
  
yellow[0]; // first frame
```



```
popup  
  
Cross-Origin:  
  
opener.length;  
// frames are leaking
```

Browsing Context Group



Paper “Finding All Cross-Site Needles in the DOM Stack”

“20 of [the Tranco Top 50 websites] were vulnerable to login detection”

Finding All Cross-Site Needles in the DOM Stack: A Comprehensive Methodology for the Automatic XS-Leak Detection in Web Browsers

Dominik Trevor Noß
Ruhr University Bochum
dominik.noss@rub.de

Lukas Knittel
Ruhr University Bochum
lukas.knittel@rub.de

Christian Mainka
Ruhr University Bochum
christian.mainka@rub.de

Marcus Niemietz
Niederrhein University of Applied
Sciences
marcus.niemietz@hs-niederrhein.de

Jörg Schwenk
Ruhr University Bochum
joerg.schwenk@rub.de

ABSTRACT

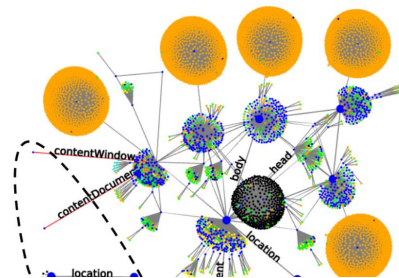
Cross-Site Leaks (XS-Leaks) are a class of vulnerabilities that allow a web attacker to infer user state from a target web application cross-origin. Fixing XS-Leaks is a cat-and-mouse game: once a published vulnerability is fixed, a variant is discovered. To end this game, we propose a methodology to find *all* leak techniques for a given state-dependent resource and a set of inclusion method. We translate a website’s DOM at runtime into a directed graph. We execute this translation twice, once for each state. The outputs are two slightly different graphs. We then get the set of *all* leak techniques by computing these two graphs’ differences. The remaining nodes and edges differ between the two states, and the corresponding DOM properties and objects can be observed cross-origin.

We implemented AUTOLEAK, our open-source solution for automatically detecting known and yet unknown XS-Leaks in *web browsers* and *websites*. For our systematic study, we focus on XS-Leak test cases for *web browsers* with detectable differences induced by HTTP headers. We created and evaluated a total of 151 776 test cases in Chrome, Firefox, and Safari. AUTOLEAK executed them automatically without human interaction and identified up to 8 403 leak techniques per test case. On top, AUTOLEAK’s systematic evaluation uncovers 5 novel classes of XS-Leaks based on leak techniques that allow detecting novel HTTP headers cross-origin. We show the applicability of our methodology on 24 web sites in the Tranco Top 50 and uncovered XS-Leaks in 20 of them.

ACM Reference Format:

Dominik Trevor Noß, Lukas Knittel, Christian Mainka, Marcus Niemietz, and Jörg Schwenk. 2023. Finding All Cross-Site Needles in the DOM Stack: A Comprehensive Methodology for the Automatic XS-Leak Detection in Web Browsers. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS ’23)*, November 26–30, 2023, Copenhagen, Denmark. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3576915.3616598>

1 INTRODUCTION



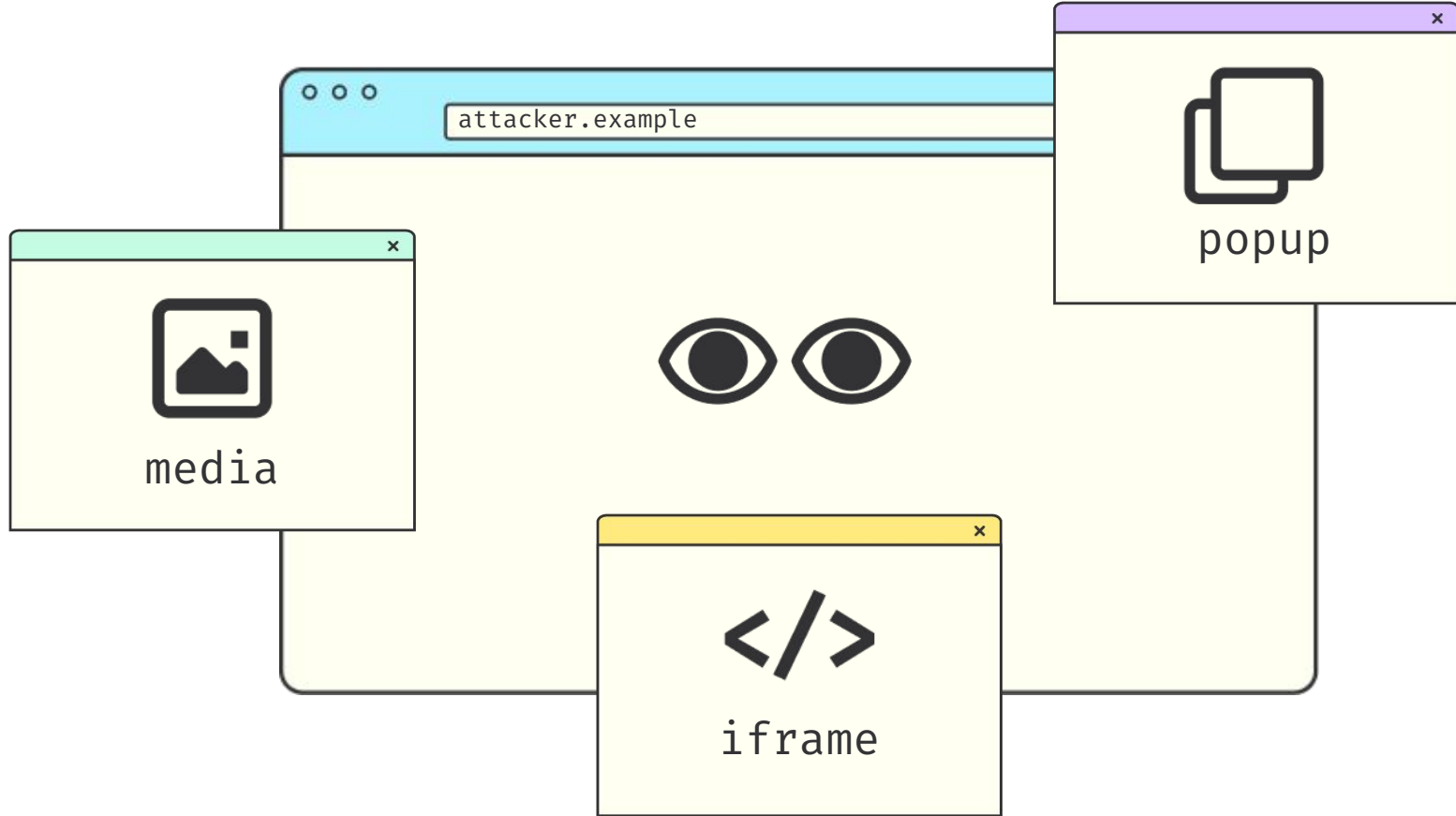
Paper “XSinator.com: From a Formal Model to the Automatic Evaluation of Cross-Site Leaks in Web Browsers”

	Chrome			Edge			Firefox			Opera			Safari	
XS-Leak	89.0	89.0	86.0	46.3.4	90.0	46.3.7	81.1.4	87.0	33.0	60.1	75.0.3	3.0.2	14.0	14.0
OS	📌	📌	📌	📌	📌	📌	📌	📌	📌	📌	📌	📌	📌	📌
Detectable Difference: Status Code														
Performance API Error	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Style Reload Error	●	●	●	●	●	●	○	○	○	○	○	○	○	○
Request Merging Error	●	●	●	●	●	●	○	○	○	○	○	○	○	○
Event Handler Error	●	●	●	●	●	●	○	○	○	○	○	○	○	○
MediaError	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Detectable Difference: Redirects														
CORS Error Leak	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Redirect Start	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Duration Redirect	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Fetch Redirect	●	●	●	○	○	○	○	○	○	○	○	○	○	○
URL Max Length	●	●	●	●	●	●	○	○	○	○	○	○	○	○
Max Redirect	●	●	●	●	●	●	○	○	○	○	○	○	○	○
History Length	●	●	●	●	●	●	○	○	○	○	○	○	○	○
CSP Violation	○	○	○	○	○	○	○	○	○	○	○	○	○	○
CSP Redirect	●	●	●	●	●	●	○	○	○	○	○	○	○	○
Detectable Difference: API Usage														
WebSocket	●	○	○	○	○	○	○	○	○	○	○	○	○	○
Payment API	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Detectable Difference: Page Content														
Performance API Empty Page	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Performance XSS Auditor	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Cache	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Frame Count	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Media Dimensions	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Media Duration	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Id Attribute	○	○	○	○	○	○	○	○	○	○	○	○	○	○
CSS Property	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Detectable Difference: Header														
SRI Error	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Performance API Download	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Performance API CORP	○	○	○	○	○	○	○	○	○	○	○	○	○	○
COOP Leak	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Performance API XFO	○	○	○	○	○	○	○	○	○	○	○	○	○	○
CSP Directive	○	○	○	○	○	○	○	○	○	○	○	○	○	○
CORP	○	○	○	○	○	○	○	○	○	○	○	○	○	○
CORB	○	○	○	○	○	○	○	○	○	○	○	○	○	○
ContentDocument XFO	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Download Detection	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Σ Attackable (max. 34)	22	23	22	24	23	22	14	13	22	24	23	24	24	24

Leaking

- HTTP Status Code
- HTTP Response Headers
- Amount of Redirects
- Length of a URL
- Cache Status
- Image Dimensions
- Video Duration
- ...





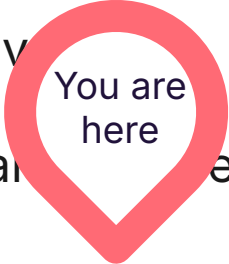
Removing bad APIs: in a complex code base

1. Write a new, improved function
2. Disallow new code to use old function
3. Switch all existing code to the new API
4. Remove the bad API.



Removing bad APIs from the web

1. Provide a better way
2. Discourage & wait for developer tools
3. Allow websites to opt-out of existing behavior
4. Advocacy? Site Outreach? Wait?
5. Maybe never actually remove the bad thing? :-)



You are
here



**There's
hope**

Cross-Origin Opener Policy

```
example.com  
  
cyan = window.open(sameOrigin);  
// [object Window]
```

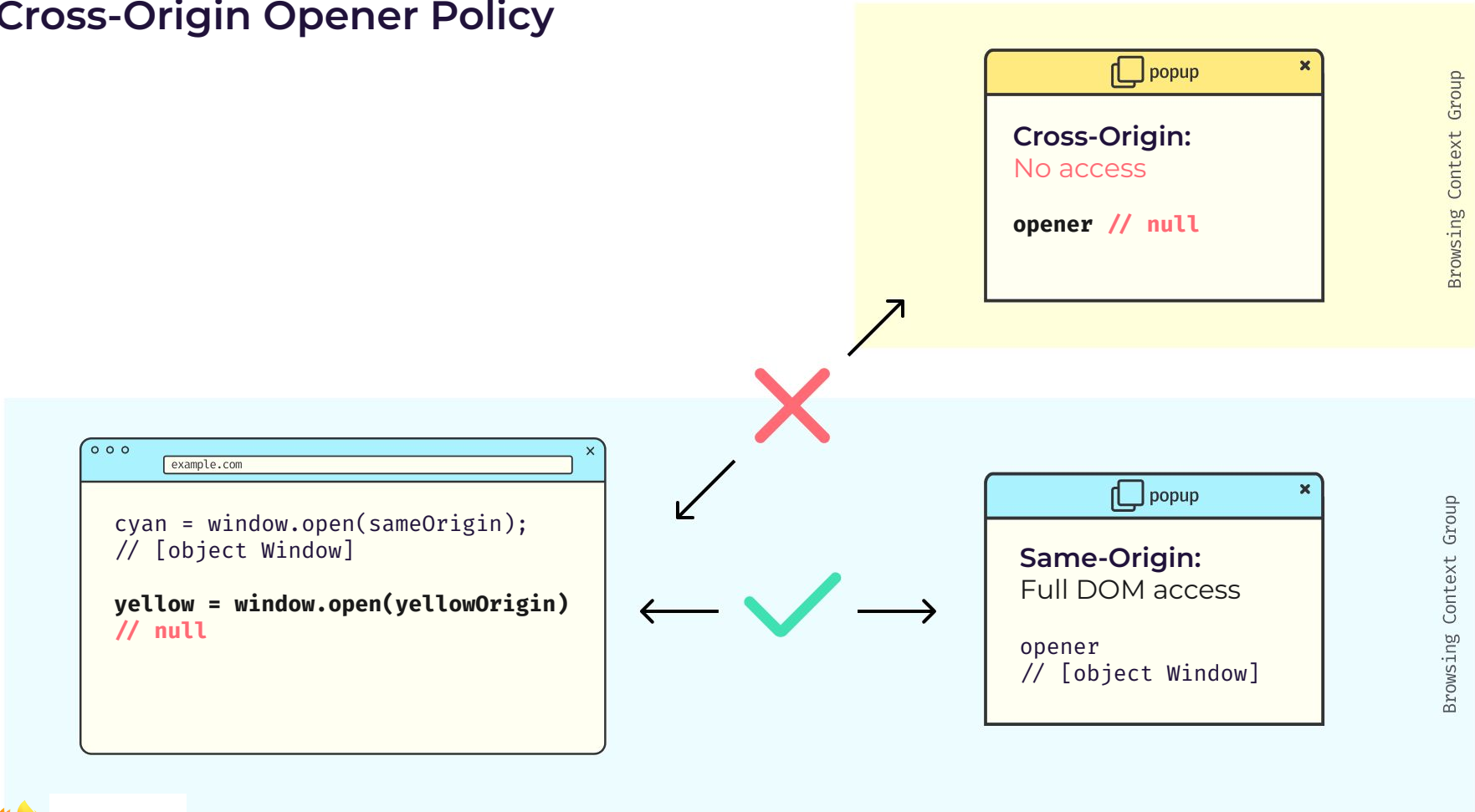


```
popup  
  
Same-Origin:  
Full DOM access  
  
opener  
// [object Window]
```

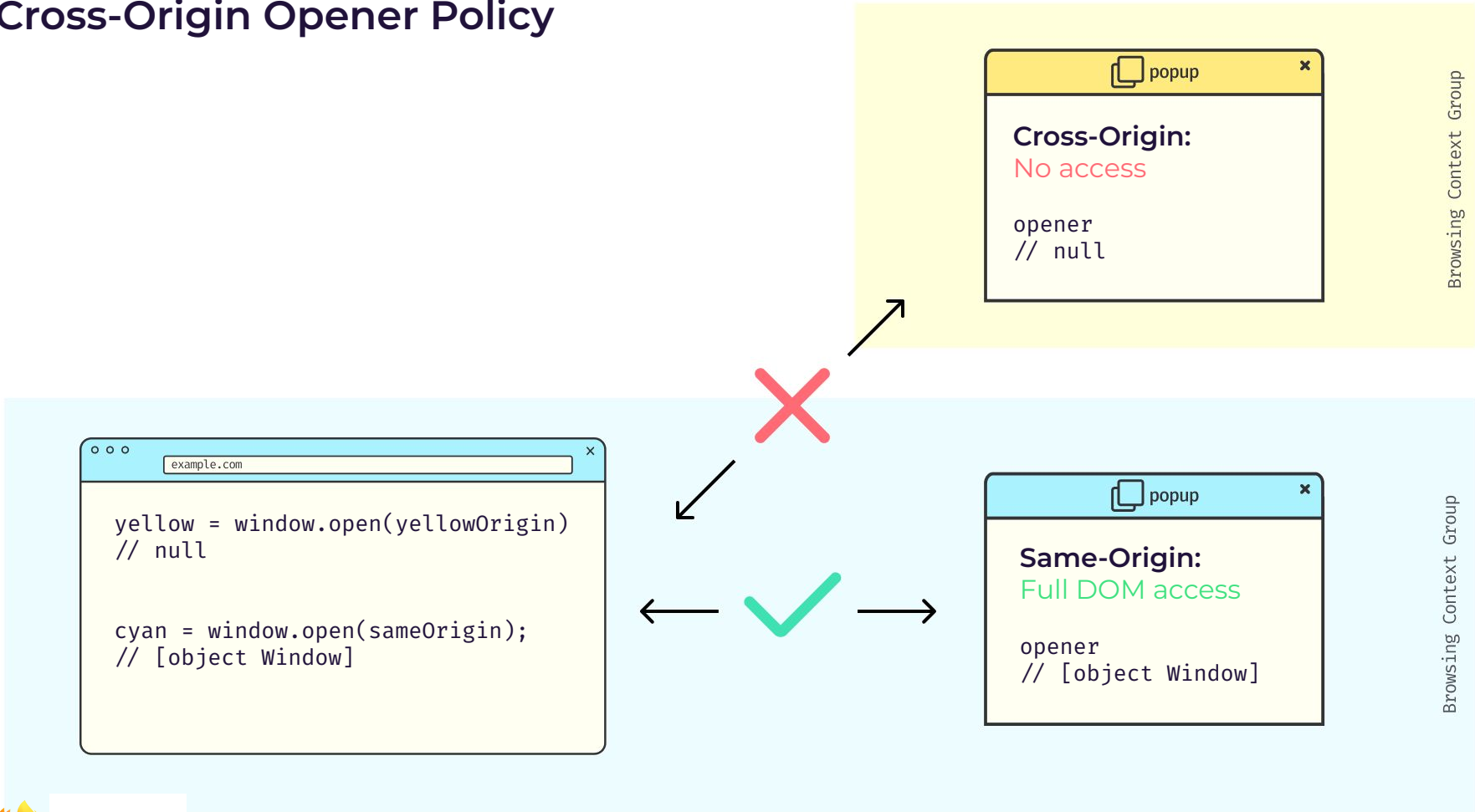
Browsing Context Group



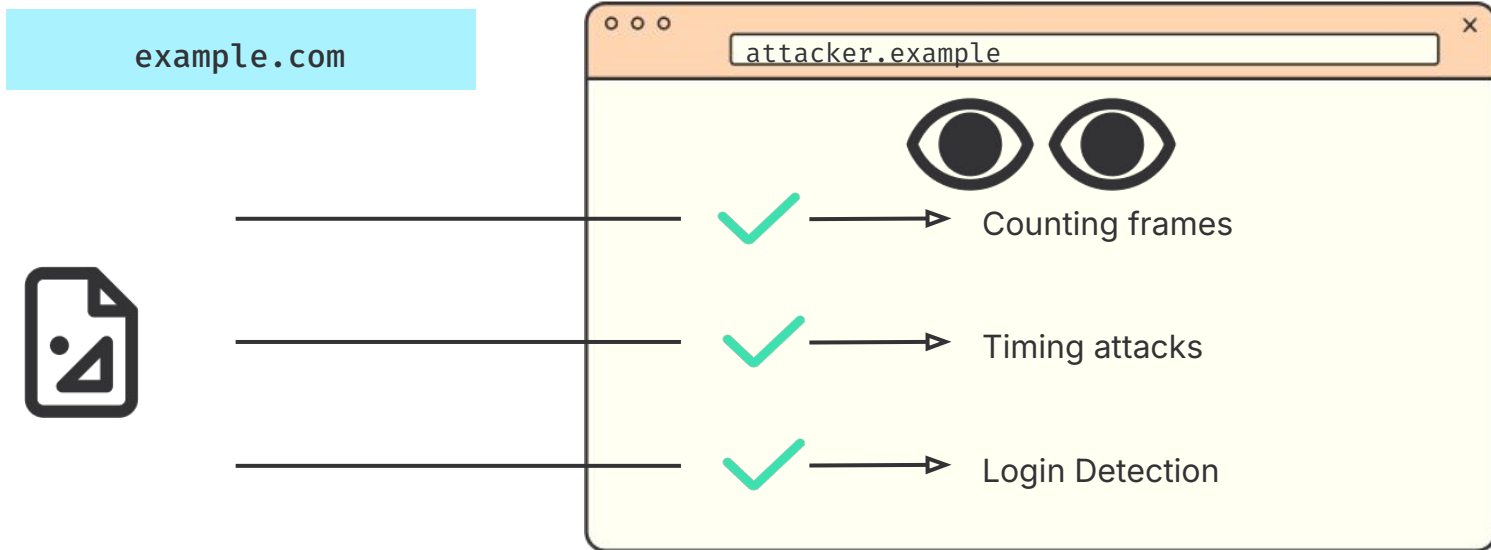
Cross-Origin Opener Policy



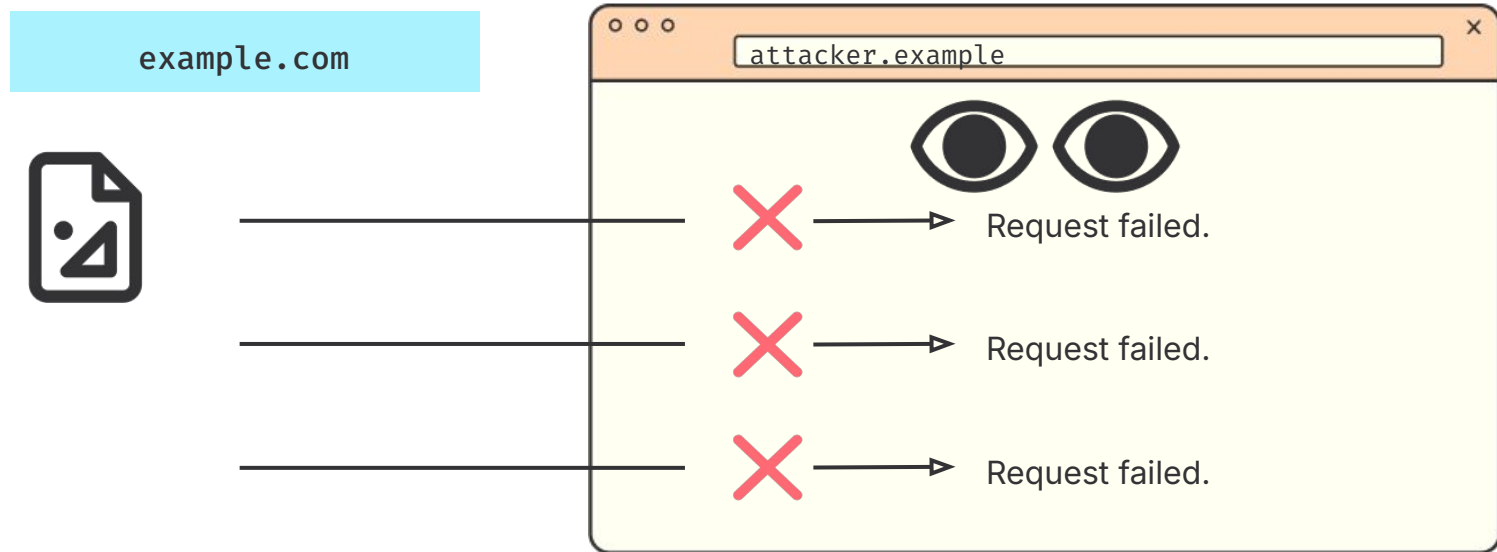
Cross-Origin Opener Policy



other leaks



Cross-Origin-Resource-Policy: same-site



Fetch

Metadata

Request

Headers

HTTP GET in curl

```
GET / HTTP/2
```

```
Host: example.com
```

```
User-Agent: curl/8.7.1
```

```
Accept: */*
```



HTTP GET in the browser

GET / HTTP/2

Host: example.com

User-Agent: ..

Accept: Accept-Language, Accept-Encoding: ...

Upgrade-Insecure-Requests: 1

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

Sec-Fetch-Site: none

Sec-Fetch-User: ?1

Priority: u=0, i

Pragma: no-cache

Cache-Control: no-cache



HTTP GET in the browser (redacted)

GET / HTTP/2

Host: example.com

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

Sec-Fetch-Site: none

Sec-Fetch-User: ?1



Fetch Metadata

Sec-Fetch-Dest:

- `<audio>`, ``, `<script>`, `<style>`, ...
- `document`, `<iframe>`, `<embed>`, `<object>`, ...



Fetch Metadata

Sec-Fetch-Mode:

- `navigate`
- `cors, no-cors`
- ...



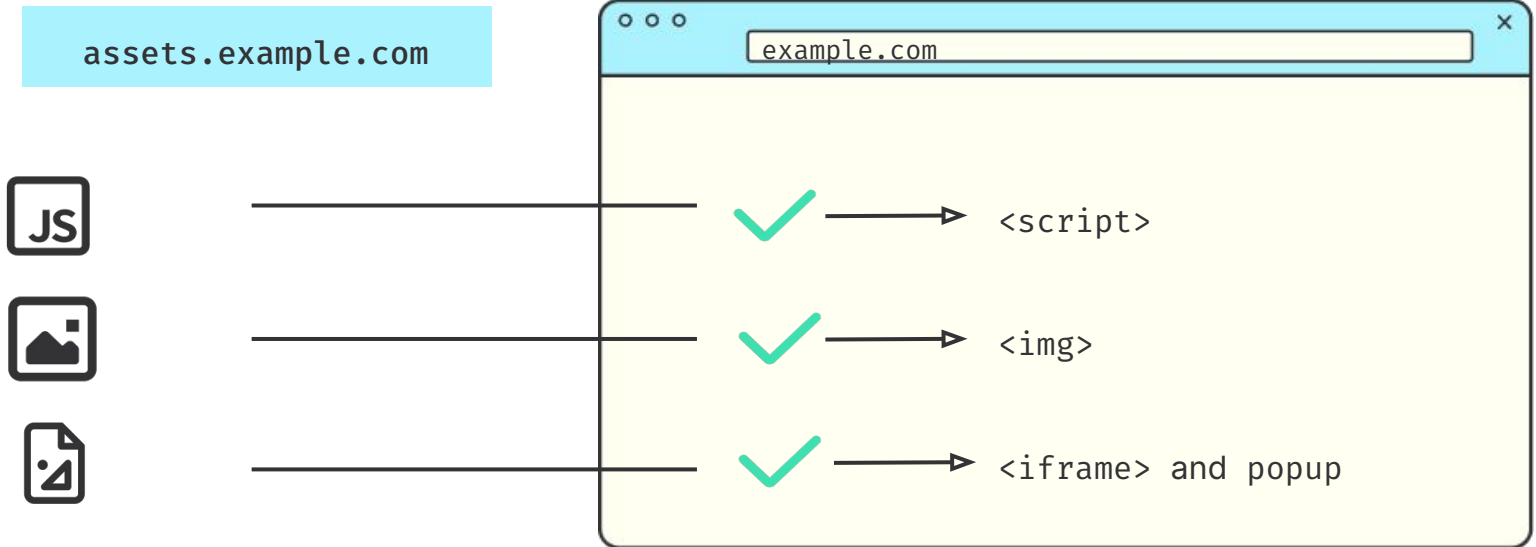
Fetch Metadata

Sec-Fetch-Site:

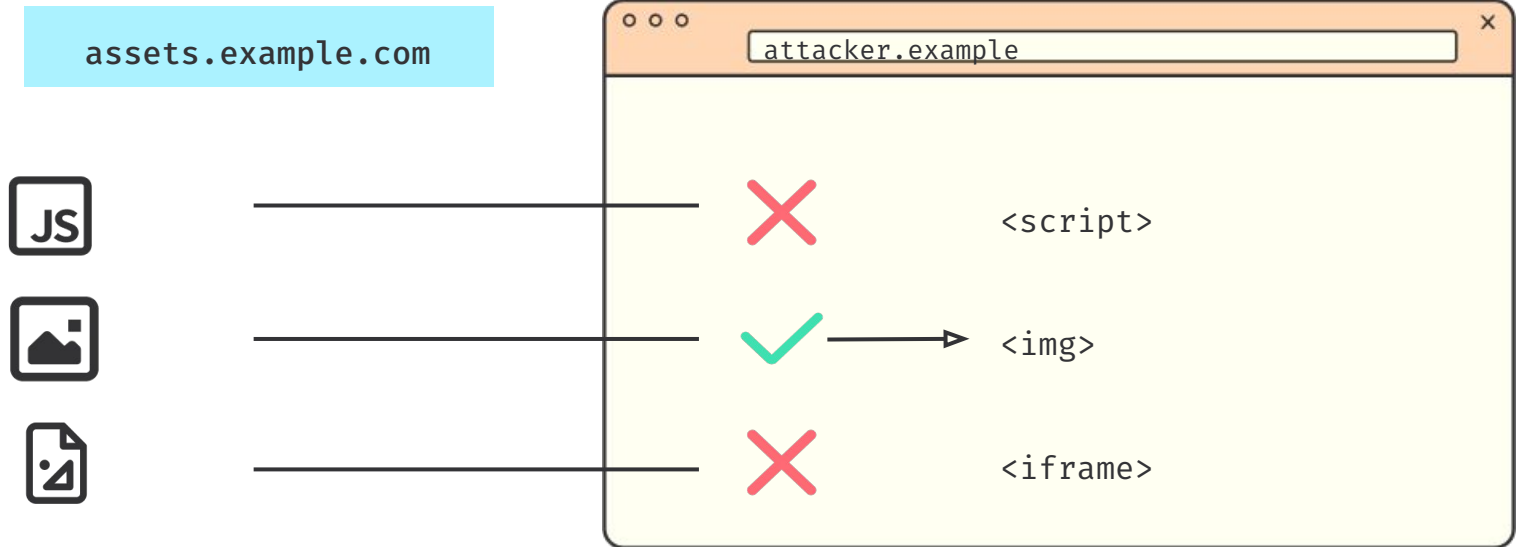
- cross-site, same-site, same-origin
- None (= user initiated)



Fetch Metadata



Fetch Metadata



**Build your own
protections**

DEMO

<https://noframes.on.web.security.plumbing/>



Thank you

Frederik Braun

 @freddy@security.plumbing

 freddy@mozilla.com

Thank you



Frederik Braun

 @freddy@security.plumbing

 freddy@mozilla.com

More

- <https://xsleaks.dev>
- [Cross-Origin-Opener-Policy on MDN](#)
- [Cross-Origin Resource Policy \(CORP\) on MDN](#)
- [Cross-Origin-Embedder-Policy on MDN](#)
- [Security headers quick reference on web.dev](#)
- [Fetch metadata request header on MDN](#)
- [Protect your resources from web attacks with Fetch Metadata on web.dev](#)
- [Fetch Metadata Request Headers playground](#)



I would be wrong if I said that there were no meaningful deprecations

- Total Cookie Protection
- SameSite Cookies
- State Partitioning
- Opaque Response Blocking (ORB)
- ...and many more

