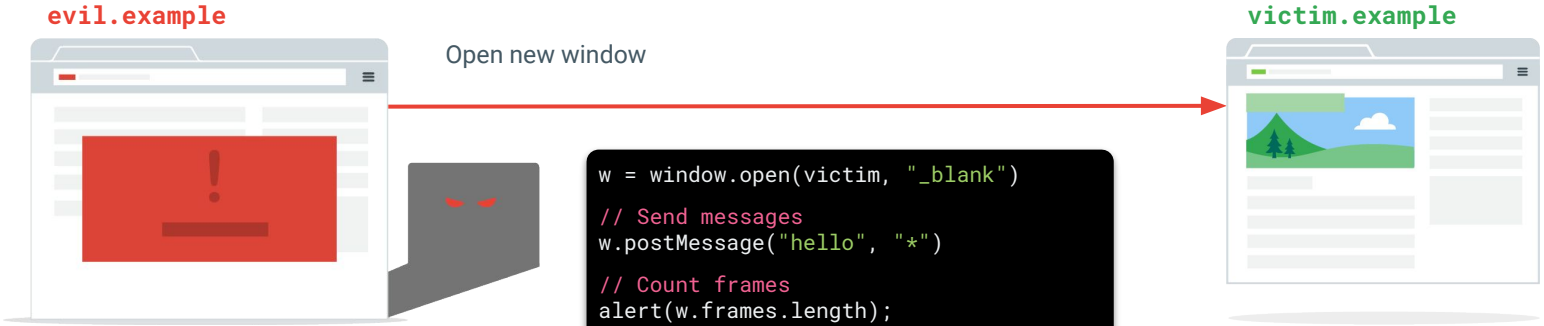
Cross Origin Opener Policy

lwe@google.com, W3C TPAC '19

COOP – Why do we need isolation?




```
w = window.open(victim, "_blank")

// Send messages
w.postMessage("hello", "*")

// Count frames
alert(w.frames.length);

// Navigate to attacker's site
w.location = "//evil.example"
```

COOP – Why do we need isolation?

A window with a Cross-Origin-Opener-Policy will be put in a **different browsing context group** from its cross-site opener:

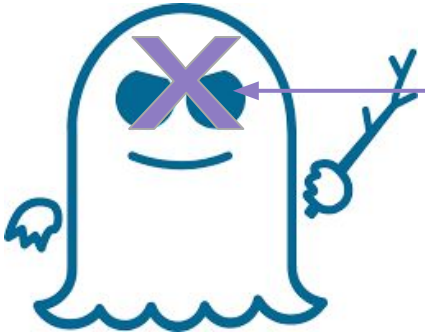
- External documents will lose direct references to the window

```
>> window.opener.postMessage('evil!', '*')
TypeError: window.opener is null [Learn More]
```

  ➡️ prevents attacks on window references:
  XS-Search, tabnabbing, login detection, Spectre

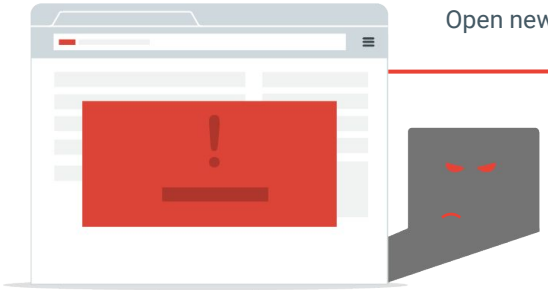
- Allows browsers without Site Isolation to put document in separate process
  ➡️ **protects** the data from speculative execution bugs (Spectre)

COOP – Why do we need isolation?



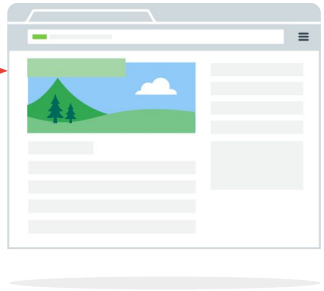
`Cross-Origin-Opener-Policy: same-origin`

evil.example



Open new window

victim.example



```
w = window.open(victim, "_blank")

// Send messages
w.postMessage("hello", "*")

// Count frames
alert(w.frames.length);

// Navigate to attacker's site
w.location = "//evil.example"
```

COOP - Overview

- If the COOP values are the same, and the origins of the documents match the relationship declared in the COOP header
  ➡️ documents can interact with each other.

- Otherwise, if at least one of the documents sets COOP
  ➡️ the browser will create a new browsing context group, severing the link between the documents.

COOP - Overview

- The browser enforces COOP by consulting the Cross-Origin-Opener-Policy **response header** on every top-level navigation

Origin 1	COOP 1	Origin 2	COOP 2	New browsing context group?	Why?
a.example.org	[none]	b.example.org	[none]	No	No COOP
a.example.org	[none]	b.example.org	same-origin	**Yes**	COOP values don't match
a.example.org	same-site	b.example.org	same-site	No	COOP value & same-site match
a.example.org	same-origin	b.example.org	same-origin	**Yes**	Origin sameness doesn't match COOP
a.example.org	same-origin	a.example.org	same-site	**Yes**	COOP values don't match
a.example.org	[none]	a.example.org	same-origin	**Yes**	COOP values don’t match

COOP - Overview

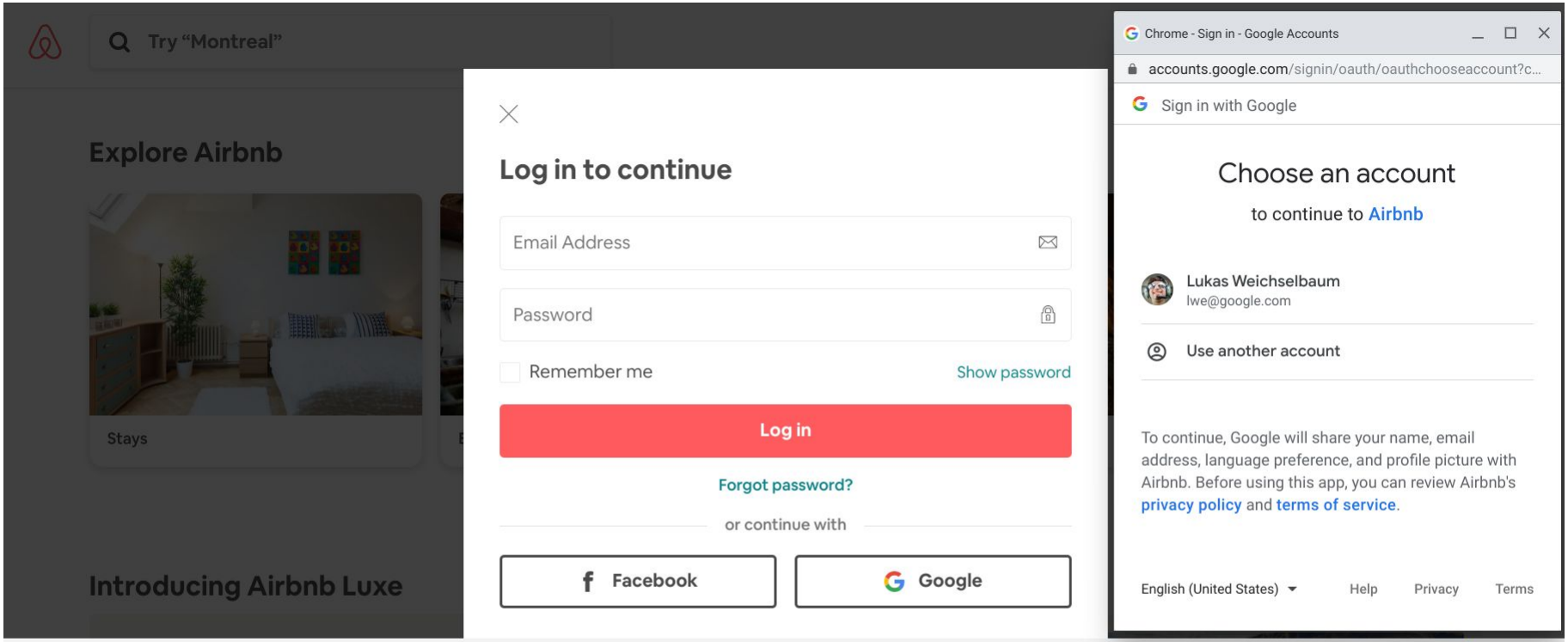
- Only applies to top-level documents (--> doesn't apply to iframes)

- Relaxations:
  - **'unsafe-allow-outgoing'** can be used to allow pop-ups being opened in same browsing context group
  - **'unsafe-inherit'** to accept the opener's choice of COOP
    → site can be used as a pop-up same-origin/same-site/cross-site.

- Neither of these values can be used in documents which desire threaded access to SharedArrayBuffer

COOP – Case Study

- Investigated over 50 Google and external sites
- Methodology
  - Created Chrome extension to track window.open and cross window interactions
    - Log such interactions
    - Keep track of origins (esp. log if same-site openee navigates to cross-origin)
  - Picked ~ 50 popular sites and sites where we expected cross-window interaction
  - Manually interacted with sites and monitored for cross window interactions
  - Investigated cross window interactions
    - Confirmed assumptions in Firefox nightly with injected COOP headers
- Credits: Raluca Radu who did all the work!



Elements Console Sources Network Performance Memory Application Security Audits COOP

Count	From	Action	Opener	Openee	Origin	Site	Stack Trace
	fromOpener	open	https://www.airbnb.com/	https://www.airbnb.com/oauth...	same-origin	same-site	window.open (<anonymous>:4:...
10	fromOpener	GET closed	https://www.airbnb.com/	https://www.airbnb.com/oauth...	same-origin	same-site	Object.get (<anonymous>:1:42...
	fromOpener	GET window	https://www.airbnb.com/	https://www.airbnb.com/oauth...	same-origin	same-site	Object.get (<anonymous>:1:42...
17	fromOpener	GET closed	https://www.airbnb.com/	https://www.airbnb.com/oauth...	same-origin	same-site	Object.get (<anonymous>:1:42...
6	fromOpener	GET closed	https://www.airbnb.com/	https://accounts.google.com/...	cross origin	cross site	Object.get (<anonymous>:1:42...
1277	fromOpener	GET closed	https://www.airbnb.com/	https://accounts.google.com/s...	cross origin	cross site	Object.get (<anonymous>:1:42...

tripadvisor

tripadvisor

I'm not a robot

reCAPTCHA
Privacy - Terms

Continue with Facebook

Continue with Google

OR

Continue with email

By proceeding, you agree to our Terms of Use and confirm you have read our Privacy Policy.

Chrome - Sign in - Google Accounts

accounts.google.com/signin/oauth/oauthchooseaccount?client_id=10...

Sign in with Google

Choose an account

to continue to TripAdvisor

Lukas Weichselbaum
lwe@google.com

Use another account

To continue, Google will share your name, email address, language preference, and profile picture with TripAdvisor. Before using this app, you can review TripAdvisor's privacy policy and terms of service.

Elements Console Sources Network Performance Memory Application Security Audits COOP

1 22

Count	From	Action	Opener	Openee	Origin	Site	Stack Trace
	fromOpener	open	https://www.tripadvisor.com/R...	https://accounts.google.com/o...	cross origin	cross site	window.open (<anonymous>:4:...
	fromOpener	focus	https://www.tripadvisor.com/R...	https://accounts.google.com/o...	cross origin	cross site	Object.focus (<anonymous>:3:...
4	fromOpener	GET closed	https://www.tripadvisor.com/R...	https://accounts.google.com/o...	cross origin	cross site	Object.get (<anonymous>:1:42...
184	fromOpener	GET closed	https://www.tripadvisor.com/R...	https://accounts.google.com/s...	cross origin	cross site	Object.get (<anonymous>:1:42...

COOP – Case Study Results

Breakdown of instances where window.open was called

- **33 instances** had no cross window interaction [COOP safe]
  - Gmail <-> Hangouts used Broadcast Channel API instead of window references
- **3 instances** were cross window interactions were same-origin [COOP safe]
- **15 instances** were cross window interactions were cross/same-site [COOP interferes]
  - Paypal integration on Google Play - uses .closed [fails]
  - Federated login when using pop-ups - implementation specific [some fail gracefully]
  - "Share on Facebook" - uses .focus() [fails gracefully]
  - Open booking.com from Google Maps - booking counts frame of opener [fails gracefully]
  - Google Script Editor - uses postMessage [fails gracefully] (more investigation needed)

COOP - Case Study Conclusion

- Majority of sites could enable COOP **without changes**
  - Most sites don't use cross-site cross window communication
  - Usages of .focus() after opening would be blocked by COOP, but is a no-op.

- The rest could enable COOP with
  - **refactoring** e.g. use of Broadcast Channel API and/or
  - COOP **policy adjustments** e.g. `'unsafe-inherit'`/`'unsafe-allow-outgoing'`
  - Most common case was federated login
    - redirect based oauth was not affected though!
    - Issues implementation specific

- Policy adjustment examples:
  - Federated login providers (e.g. Google Oauth) could set COOP to `'unsafe-inherit'`
  - Google Play could set `'unsafe-allow-outgoing'` in COOP for their Paypal integration

COOP - Deployment Hurdles

- Rollouts need to be coordinated across **multiple** origins/services
  - Hard to rollout header across different services simultaneously
  - Even harder for percent rollouts (experiments)
  - No rollback safety, if only one service rolls back

➡️ Enable COOP `'unsafe-inherit'` **before** rolling out actual COOP enforcement

- Sites involved in cross window communication are often unknown upfront
  - COOP header needs to be set on **all** sites involved in cross window communication!

➡️ Reporting **[Feature Request]**

- COOP setup cannot be tested without enforcing (breakages erode trust)
- If COOP breaks (niche) functionality, discovery could be slow

➡️ "report-only" mode **[Feature Request]**

COOP - Discussion

- Chrome extension DEMO, if there's interest
- Which form of reporting can we support?
  - full report-only mode
  - reporting only when COOP kicks in (window.open / child navigation)
  - no reporting
- What to use instead of reference based cross window interaction?
  → Broadcast Channel API for framed A on B -> child A?
- Allowing SharedArrayBuffer access in presence of COOP & COEP?
- Is there an easier mechanism than COOP to achieve the same?
  - allow postMessage (via IPC) via a flag could potentially simplify adoption