

Fetch Metadata Request Headers



lwe@google.com, W3C TPAC '19

Fetch Metadata – TL;DR

- Allow **servers** to **protect themselves** from canonical cross-origin attacks
 - CSRF, XSS, timing attacks, clickjacking, Spectre
- The **browser sends** information about the request's **context** in HTTP headers
 - ➡ The server makes security decisions based on the source and context of the request
- Fetch Metadata shipped in Chrome 76
 - ➡ Currently piloting resource isolation at Google

Fetch Metadata – Recap

New **HTTP request headers** sent by the browser:

- **Sec-Fetch-Site** Which website generated the request?
same-origin, same-site, cross-site, none
- **Sec-Fetch-Mode** The Request *mode*, denoting the type of the request
cors, no-cors, navigate, nested-navigate, same-origin
- **Sec-Fetch-User** Was the request caused by a user gesture?
?1 if a navigation is triggered by a click or keypress

Fetch Metadata - Example

`https://site.example`

```
fetch("https://site.example/foo.json")
```

```
GET /foo.json
Host: site.example
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
```

`https://evil.example`

```

```

```
GET /foo.json
Host: site.example
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: no-cors
```

Fetch Metadata – Resource Isolation

Basic idea

Block cross-site requests [Sec-Fetch-Site: cross-site]

Unless:

- It's a non state-changing [!POST] **navigational request**
Sec-Fetch-Mode: navigate *or* Sec-Fetch-Mode: nested-navigate
- The action/servlet is **whitelisted** for cross-site traffic (e.g. a CORS endpoint)
- **Prevents** attacks based on the attacker forcing the loading of the resource in an attacker-controlled context

```
# Resource Isolation
```

```
def allow_request(req):
```

```
    # Allow requests from browsers which don't send Fetch Metadata
```

```
    if not req['sec-fetch-site']:
```

```
        return True
```

```
    # Allow same-site, same-origin, and browser-initiated requests
```

```
    if req['sec-fetch-site'] in ('same-origin', 'same-site', 'none'):
```

```
        return True
```

```
    # Allow simple (non-state changing) navigations from anywhere
```

```
    if req['sec-fetch-mode'] in ['navigate', 'nested-navigate']
```

```
        and req.method in ['GET', 'HEAD']:
```

```
        return True
```

```
    return False
```


Fetch Metadata – Policy Configurations

Many common attacks can be mitigated by blocking certain cross-site requests

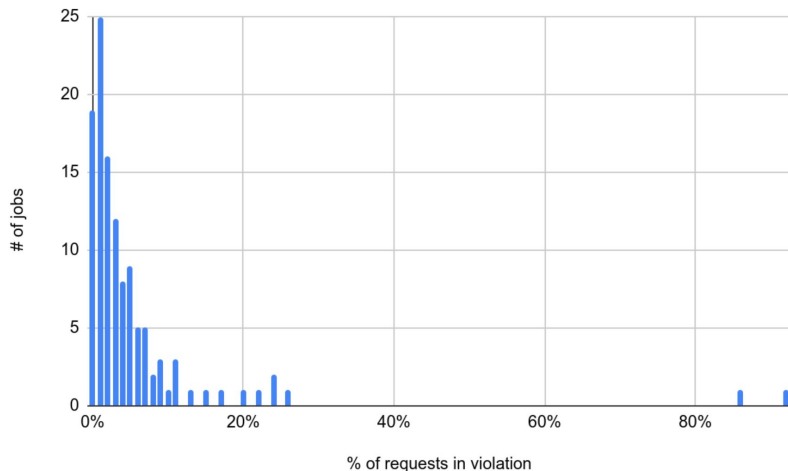
	site	mode	user	method
Resource Isolation	'cross-site'	NOT '(nested-)navigate'		
CSRF	'cross-site'			POST
XSSI / JSON hijacking	'cross-site'			
Timing attacks	'cross-site'	NOT '(nested-)navigate'		
Spectre	'cross-site'			
Navigation Isolation	'cross-site'	'(nested-)navigate'		
Clickjacking	'cross-site'	'(nested-)navigate' .		
XSS (reflected)	'cross-site'	'(nested-)navigate'		

Resource Isolation – Google Internal Pilot

- **Large scale** (400+ services/products) **Resource Isolation** experiment enabled
- **Report-only**: Apply, but don't act on the policy (don't reject requests)
- Log the policy result along with relevant metadata:
 - Sec-Fetch-* headers, HTTP method, response code, relevant response headers, ...
 - Presence of Access-Control-* headers means an endpoint has legitimate cross-site traffic

Resource Isolation – Pilot Results

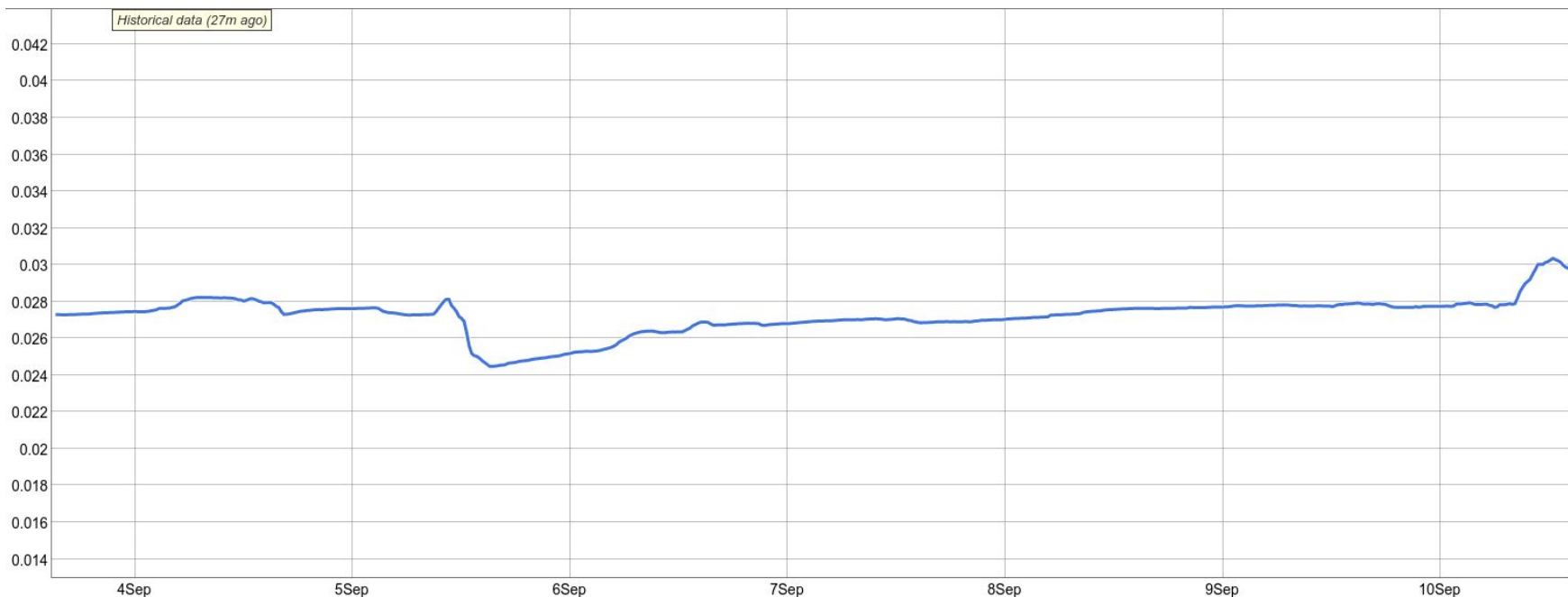
- Almost **70%** of ~400 services serving text/html **compatible without changes**
- **30%** need **whitelisting** of endpoints serving legitimate CORS / cross-site traffic



- Some policy violations across services due to browser bugs (mostly fixed)

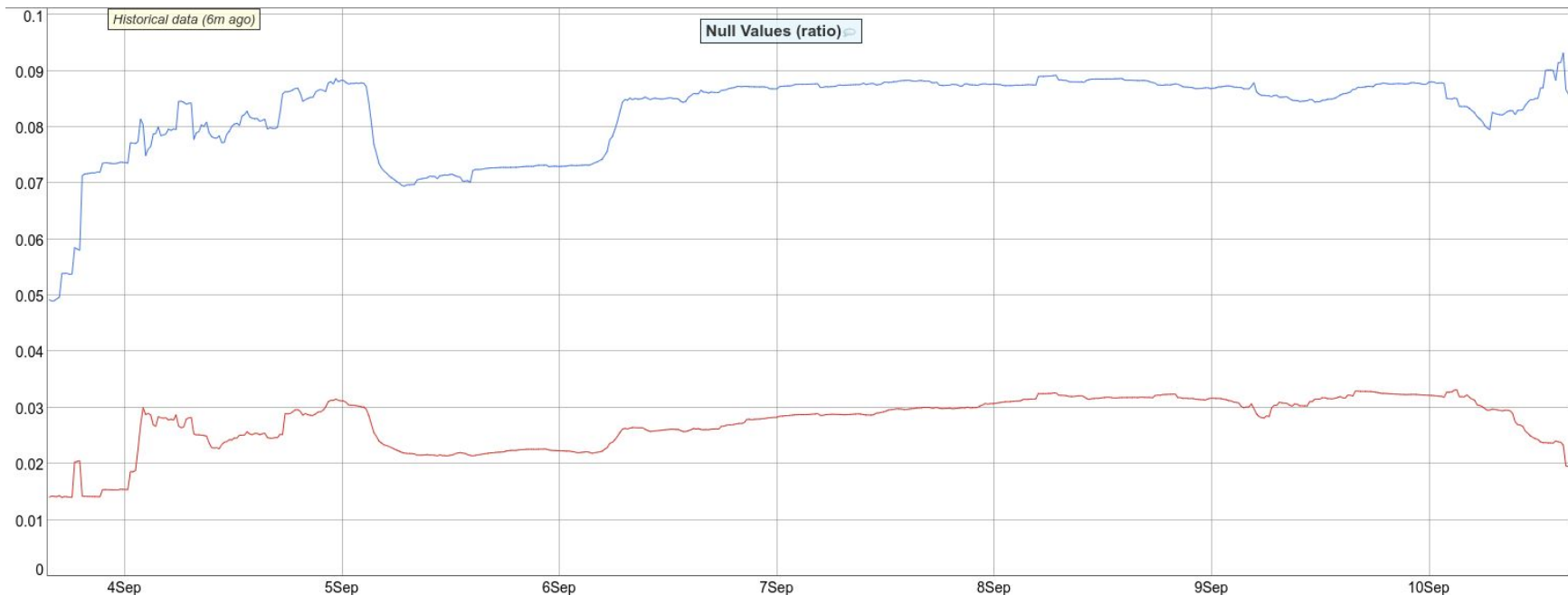
Resource Isolation – Pilot Results

Most valid cross-site endpoints whitelisted - violations occur for <3% of traffic



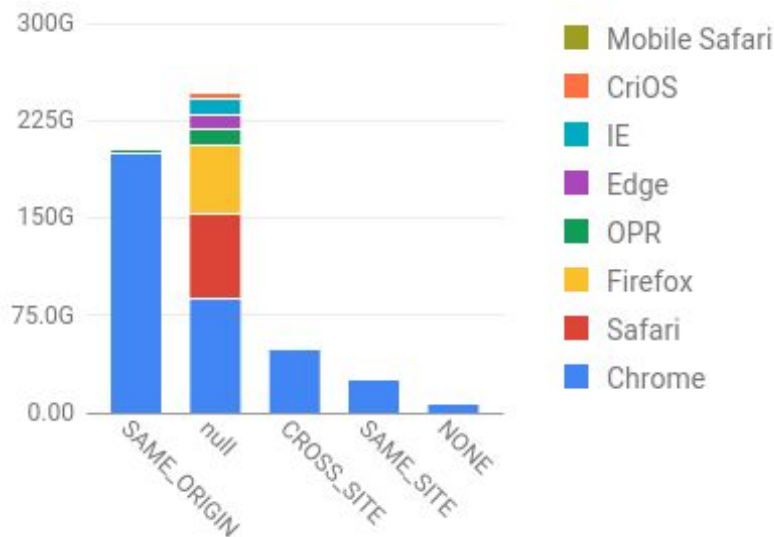
Resource Isolation – Pilot Results

Missing Sec-Fetch-[Site|Mode] headers can indicate browser bugs



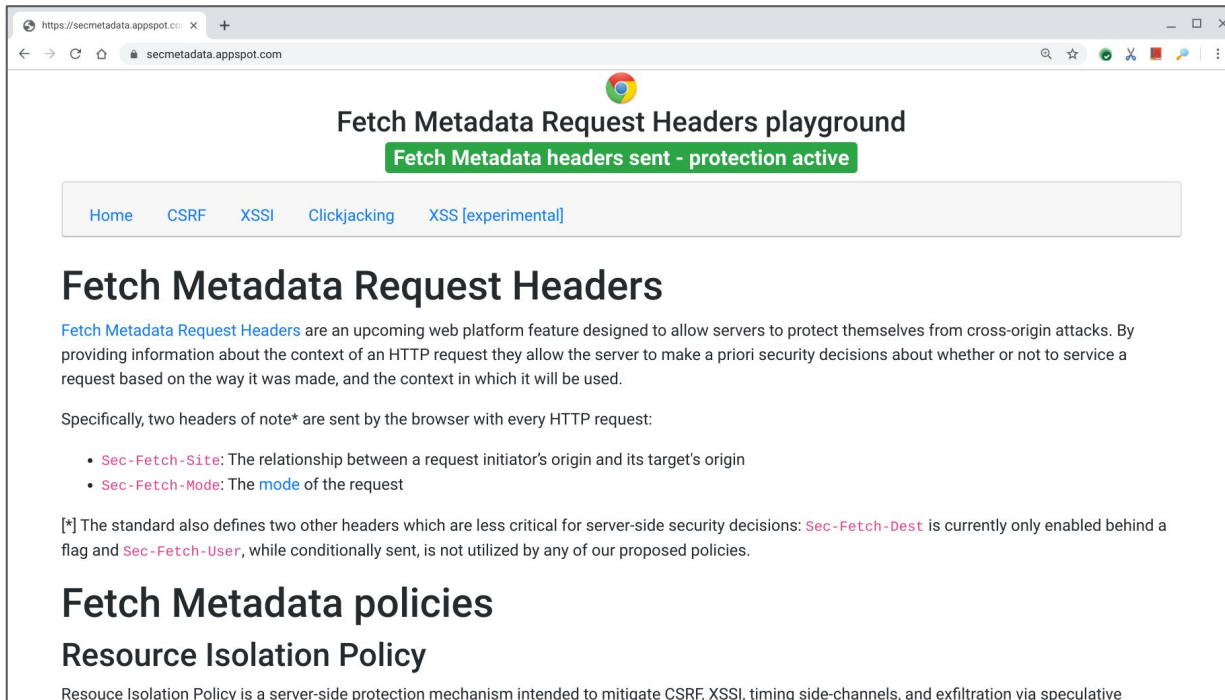
Next steps: what would be *awesome*?

- **Cross-browser support**
- Consistent / Complete implementation (Web Platform Tests)



Fetch Metadata – Playground

<https://secmetadata.appspot.com>



The screenshot shows a web browser window with the address bar displaying `https://secmetadata.appspot.com`. The page title is "Fetch Metadata Request Headers playground". Below the title, a green banner states "Fetch Metadata headers sent - protection active". A navigation bar contains links: "Home", "CSRF", "XSSI", "Clickjacking", and "XSS [experimental]". The main heading is "Fetch Metadata Request Headers". The text explains that Fetch Metadata Request Headers are an upcoming web platform feature designed to allow servers to protect themselves from cross-origin attacks by providing information about the context of an HTTP request. It specifies that two headers are sent by the browser with every HTTP request:

- **Sec-Fetch-Site**: The relationship between a request initiator's origin and its target's origin
- **Sec-Fetch-Mode**: The `mode` of the request

[*] The standard also defines two other headers which are less critical for server-side security decisions: **Sec-Fetch-Dest** is currently only enabled behind a flag and **Sec-Fetch-User**, while conditionally sent, is not utilized by any of our proposed policies.

Fetch Metadata policies

Resource Isolation Policy

Resource Isolation Policy is a server-side protection mechanism intended to mitigate CSRF, XSSI, timing side-channels, and exfiltration via speculative

Fetch Metadata – Discussion

- Overlap with Sec-Same-Origin
- Resource Isolation can be used to deploy CORP
- Vary headers on Sec-Fetch-Site force separate cache for same-origin traffic