CSP

Practical issues

Background

- Previously at Microsoft, Cloudflare & Vercel
- Cside
  - 26 staff of which 21 engineers
  - 2 rounds of funding, seed stage startup
  - Analyze client-side fetched dependencies through a hybrid proxy
  - Analyzing 30 million scripts per day
  - Customers: banks, insurrers, e-commerce, payment providers…

Personal motivation:

- Safer web, for everyone, not only the resource rich, not only those with full-time security teams.
- Prevent bad actors from destabilizing the free world we all rely on.
  We can not be safe as humans in a digital age unless we think like the bad actor and push for what is right (and actionable). Hence I'm here…

Oh and lol, it's my birthday today.

I spoke to a few customers to prep for this presentation

- E-commerce brands x3
- Payment providers x2
- SaaS x2

But many more years of practical experience and hair loss here…

Who pushes for CSP adoption in companies?

- GRC teams (Governance Risk Compliance)
- Security teams
- Not the front-end engineers…
- Ofcourse ***never*** the marketeers in Google Tag Manager

=> CSP adoption is low and that is not because people didn't try, those that want it have to pick their fights. Those that had it rebound back to removing it.

The truth about security teams

- The general rule of thumb, per 50 employees (maybe) 1 security person
- Often do not touch code themselves
  - Or at least need to work with another team to get something done
- Have to beg for engineering resources
- Pushback is the norm
  - not considered core team because hard to route back revenue or business wins to them
- “Wait, this things need constant maintenance? We don’t have capacity for that. Not doing it”
- Often chasing compliance and internal existential risk projects.

Any security engineer that brings CSP into a company gets fried.

The business after security imposing CSP blocking mode

CSP management is a disaster

1. Chatbots => constantly changes dependencies
2. Analytics tools => constantly changes dependencies
3. Ad networks => constantly changes dependencies
4. Marketing teams like to use Tag Manager => but not updating CSP

- No notice is given on changes
- Problem is noticed when it is already too late
- Difficulty to make fast changes

How fast can a company ship a CSP change?

- Normal release cycles 2-5 days
- Emergency changes ‘couple of hours’
- What if it’s my payment provider’s client-side script?

If they use a proxy, changes can* happen a lot faster. But…

=> If a proxy is required for a specification to be actionable, the specification is wrong.
A proxy should NEVER be the standard, it’s a patch to imperfection at the spec level.

*IaC is an exception to this rule.

How do I even know what I should put in a CSP?

- Report-only => noisy ugly and alarming badly worded alerts
  - [Report Only] Refused to load => still loads ?!?!?!?!
- By the time I have my reports, it is already outdated

How CSP is implemented

- Who built a reporting endpoint from scratch recently?
- The spec is interpreted as broadly as the recipe of Bolognese sauce

```
export const SINGLE_REPORT_TO_SCHEMA = z.object({
  // Some null cases below as in union with undefined, because the chrome doesn't send them at all if null.
```

```
  // IMPLEMENTATION NOTE: THE FOLLOWING IS MADE OPTIONAL BECAUSE SAFARI DOES NOT SEND IT, AS IT HAS NOT CORRECTLY
  // IMPLEMENTED THE REPORTING API. THIS IS A SAFARI BUG, NOT A SPEC BUG.
  age: z
    .number()
    .min(0)
    .or(z.undefined().transform(() => 0)), // Number of milliseconds since the violation occurred and this request was

  type: z.literal("csp-violation"), // The reporting system is multi-purpose, we only are ingesting csp-violations.
  url: z.string().url(), // Where the report was generated from

  // IMPLEMENTATION NOTE: THE FOLLOWING IS MADE OPTIONAL BECAUSE SAFARI DOES NOT SEND IT, AS IT HAS NOT CORRECTLY
  // IMPLEMENTED THE REPORTING API. THIS IS A SAFARI BUG, NOT A SPEC BUG.
  user_agent: z.string().or(z.undefined()), // User agent string
});
```

Types of Headaches

Migraine

Hypertension

Stress

CSP

Strict-dynamic

Feedback:

- Wow, wait? So I am now explicitly allowing any subrequest from a script? So I have the trust the entire supply chain based on the parent? Why use CSP in the first place? I am closing the gate to then open it again?

On sharing you can still define an allowlist:

```
Content-Security-Policy: script-src 'nonce-abcdefg' 'strict-dynamic', script-src https://my.amazing.cdn
```

Yea but what is the point then in the first place?

- Wait so this would only work for dynamic sites? (nonce)
- Literal quote “a very magento take on things”
- Internal pushback: Bad actors inject client-side fetches inside NPM scripts. 1st party scripts will need strict-dynamic. This proposal will result in more negative impact.
- Internal pushback: Bad-actors already use googletagmanager.com to bypass CSP. It is a matter of time before Google recommends Strict-dynamic on GTM and at that point we are much worse off.

On a different note, making sense of Strict-dynamic was a lot in the first place.

The point:

- CSP was not built with
  - dynamicness in mind
  - CSR in mind
  - NPM in mind
  - how it would actually go down in an average company (not google, not meta…)
- Script-dynamic - by all means, support it. But this is **not** adding security.

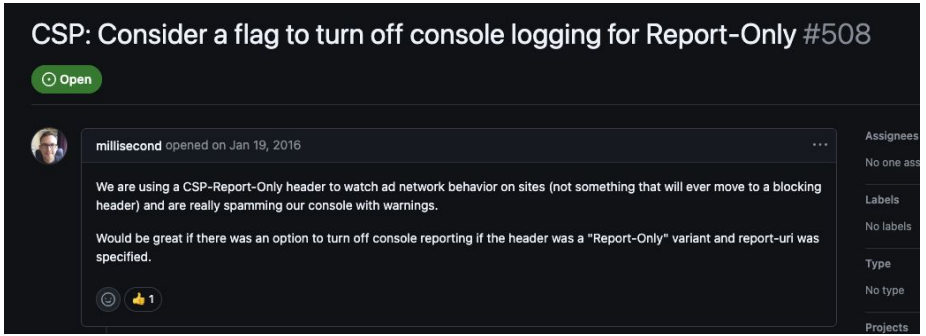
  It sure adds convenience in some cases but it is not a real solution to the problem at hand.

What people really want and asked for

Let’s not ignore our responsibility here.

- Ability to [silence report-only console logs]
- Fix the shouty wording of a non-blocking report-only console logs
- Allow to set CSP’s elsewhere, not hard coded, not in a proxy, not in a meta tag and a janky server side dynamic element

```
[Report Only] Refused to load the script '<URL>' because it violates the following Content Security Policy directive: "script-src 'none'". Note that 'script-src-elem' was not explicitly set, so 'script-src' is used as a fallback.
```



Questions I was asked before

- But what about CSP in <meta> elements?
  - Logical separation between code and headers is a feature, not a bug.
  - Limitations of features is a thing.
  - Does not solve most of the problems I raised before.
- Sure but can’t you use <meta> and make a customer server-side component?
  - Yes, but I could also build Doom in a PDF or build a back-end in CSS.
    Is a specification good if you need to get creative to achieve basic daily usability?

The objections I expect

- But the added latency of doing a render blocking 3rd party fetch
  - People add literal proxies in front of their website to make CSP work => performance issue too
  - It is not because something *could* be slow, it should not be an option
    => it can be done fast too
    => security features often add latency - we all want SSL right?
    If we let this argument carry any weight, we might as well all go home and revert everything we did in the last 25 years.
- Your sample size is not big enough.
  - CSP adoption is low. We're a vendor with meaningful customers. Customers that come to us because CSP failed them. We are not the only one. We need to do the creativity for our customers because it is such a burden. Sorry but this is not a helpful take. What are we really trying to achieve here? For me it is simple: **make good security practices more accessible to everyone**.

Q&A