Exploring Content Security Policies

December 8th 2023
What even is it?

- Added layer of security for websites
- Allowlist of website resources
- A W3C standard for browser implementations
How it works

Browser

Please load font from fonts.google.com

CSP

Browser loads fonts from fonts.google.com

Resource
How it works

Browser

Please load script from malicious.domain.net

CSP

It depends...

Resource
How it works

In short… A CSP is a bouncer

Please load script from malicious.domain.net

It depends…

Browser

CSP

Resource
Pros & Cons

- **Benefits**
  - Blocks malicious resources
  - Blocks browser extension modifications
  - Identifies external resources being used
  - Improves our security score

- **Cons**
  - Allowlist approach is a big ask
  - Potentially slow to roll out
  - Somewhat complicated for developers to understand
  - Can break your website!
The list... mostly

- Directives: Types of resources the website might load
  - `script-src`: Scripts (Javascript)
  - `style-src`: Styles (CSS)
  - Etc...

- Directive Values: What is allowed for the given directive
  - `URL`: [https://fonts.google.com](https://fonts.google.com) or [https://fonts.google.com/family/font/style.css](https://fonts.google.com/family/font/style.css)
  - `'self'`: Tells the browser the website is allowed to load resources from the website
  - `data::`: Tells the browser embedded data is allowed (Typically used for `img-src` directive)
  - `blob::`: Tells the browser blob data is allowed (Typically used for `img-src` directive)
  - `'none'`: Tells the browser there are no allowed resources for the directive
  - `'unsafe-inline'`: Tells the browser it’s okay to render resources that are embedded in the html
  - `'unsafe-eval'`: Tells the browser it’s okay to perform script evals
  - `'nonce-'` or `'sha256-'`: Tells the browser it’s okay to trust files or inline items if they match hashes

- `report-uri` & `report-to`: Where should the browser send violation reports?
The list... mostly

- Directives: Types of resources the website might load
  - script-src: Scripts (Javascript)
  - style-src: Styles (CSS)
  - Etc...

- Directive Values: What is allowed for the given directive
  - URL: [https://fonts.google.com](https://fonts.google.com) or [https://fonts.google.com/family/font/style.css](https://fonts.google.com/family/font/style.css)
  - 'self': Tells the browser the website is allowed to load resources from the website
  - data:: Tells the browser embedded data is allowed (Typically used for img-src directive)
  - blob:: Tells the browser blob data is allowed (Typically used for img-src directive)
  - 'none': Tells the browser there are no allowed resources for the directive
  - ‘unsafe-inline’: Tells the browser it’s okay to render resources that are embedded in the html
  - ‘unsafe-eval’: Tells the browser it’s okay to perform script evals
  - ‘nonce-’ or ‘sha256-’: Tells the browser it’s okay to trust files or inline items if they match hashes

- report-uri & report-to: Where should the browser send violation reports?

This was intentional

Maybe use this sometimes...

Avoid this like the plague...
Let’s make a CSP

- Options
  - Content-Security-Policy Header
  - Content-Security-Policy-Report-Only Header

- IT’S GOOD. IT’S GOING TO BE GOOD. -YEAH? GREAT.
Let's make a CSP

- Options
  - Content-Security-Policy Header
  - Content-Security-Policy-Report-Only Header
Let’s make a CSP

- Options
  - Content-Security-Policy Header
  - Content-Security-Policy-Report-Only Header
Let's make a CSP

- Options
  - Content-Security-Policy Header
  - Content-Security-Policy-Report-Only Header

```json
"helmet": {
  "csp": {
    "directives": {
      "default-src": [
        "'self'
      ],
      "report-uri": [
        "https://local-dev.robotti.private/csp"
      ]
    },
    "reportOnly": false
  }
}
```
Let’s make a CSP

- Options
  - Content-Security-Policy Header
  - Content-Security-Policy-Report-Only Header

Hey… I tried to do that thing, but the bouncer said NO!
Reporting
Reporting
Reporting
So lets add this
Wait... what?

So now, because we have a script-src-elem directive all our scripts are blocked?

Because we didn’t include ‘self’ for that directive.
Wait... what?

We also have this new script that is being blocked?

Because the script we allowed pulled it in.
Lets be smarter about it
Let's be smarter about it

The Content Security Policy directive "upgrade-insecure-requests" is ignored when delivered in a report-only policy.

The Content Security Policy directive "upgrade-insecure-requests" is ignored when delivered in a report-only policy.

The Content Security Policy directive "upgrade-insecure-requests" is ignored when delivered in a report-only policy.

The Content Security Policy directive "upgrade-insecure-requests" is ignored when delivered in a report-only policy.

The Content Security Policy directive "upgrade-insecure-requests" is ignored when delivered in a report-only policy.

The Content Security Policy directive "upgrade-insecure-requests" is ignored when delivered in a report-only policy.

The key "user-scalable" is not recognized and ignored.

[Report Only] Refused to load media from "https://local-dev.robotte_private/images/cincinnati-skyline.png" because it violates the following Content Security Policy directive: "default-src none". Note that 'media-src' was not explicitly set, so 'default-src' is used as a fallback.

[Report Only] Refused to load media from "https://local-dev.robotte_private/images/cincinnati-skyline.png" because it violates the following Content Security Policy directive: "default-src none". Note that 'media-src' was not explicitly set, so 'default-src' is used as a fallback.

[Report Only] Refused to load the script "https://www.google.com/recaptcha/api.js" because it violates the following Content Security Policy directive: "script-src 'self'". Note that 'script-src-elem' was not explicitly set, so 'script-src' is used as a fallback.

[Report Only] Refused to load the script "https://www.localhost.com/recaptcha/api.js" because it violates the following Content Security Policy directive: "script-src 'self'". Note that 'script-src-elem' was not explicitly set, so 'script-src' is used as a fallback.

[Report Only] Refused to frame 'https://www.google.com/' because it violates the following Content Security Policy directive: "default-src none". Note that 'frame-src' was not explicitly set, so 'default-src' is used as a fallback.

[Report Only] Refused to frame 'https://www.googleapis.com/' because it violates the following Content Security Policy directive: "default-src none". Note that 'frame-src' was not explicitly set, so 'default-src' is used as a fallback.

[Report Only] Refused to frame 'https://www.googleapis.com/' because it violates the following Content Security Policy directive: "default-src none". Note that 'frame-src' was not explicitly set, so 'default-src' is used as a fallback.

Because we used the content-security-policy-report-only we reported, but didn’t block

Which means this script was loaded and also reported
But there's something off about this...

I never added script-src 'self'
Well we assumed...

Some libraries that handle these headers assume you need certain directives set.

Directives:
- base-uri
- font-src
- form-action
- frame-ancestors
- img-src
- script-src
- style-src

What’s worse is they sometimes assume bad not great things
Okay... Big brain smart time

Let’s be super explicit with our policy.
And set everything to ‘none’
Okay... Big brain smart time
Oh wait... We have report-uri

- Violation reporting services exist
  - Report-uri.com
  - Csper.com
Oh wait... We have report-uri
Oh wait... We have report-uri
Oh wait... We have `report-uri`

Why all the validation?
1. Because this is open to all
2. I'm not hypocrite
3. It's javascript
Scaling Across Organization

- Training Developers
- Communication
- Pilot Groups
- Provide Tooling Access
- Phased Approach
- Progress Over Perfection
For science

- What do you do with the reports?
- Bandwidth?
- Other Pages?
- Recommended?
- Was it worth the effort?