Content Security Policy for Privacy

Devin Lundberg

PEPR 2023
Third party domains and privacy

Identity Providers

Bot & fraud detection

Analytics

Marketing Pixels
User data received by client side third parties

- Data sent by your application
- Cookies
- IP
- User agent
- Passive HTTP, TLS, or TCP fingerprinting
What is Content Security Policy (CSP)?

- HTTP header or meta tag
- Tells browser what domains are expected to be used by the website
- Commonly used to mitigate cross site scripting (XSS)

**Example**

```
Content-Security-Policy: default-src 'self'
blob: s.pinimg.com; script-src 'self'
'nonce-0260cb' 'strict-dynamic'
*.example-analytics.com; img-src 'self'
i.pinimg.com; report-uri /__/csp_report/
```
What is Content Security Policy (CSP)?

- HTTP header or meta tag
- Tells browser what domains are expected to be used by the website
- Commonly used to mitigate cross site scripting (XSS)

Example

Content-Security-Policy: `default-src 'self'
 blob: s.pinimg.com; `script-src 'self'
 'nonce-0260cb' 'strict-dynamic'
 '*.example-analytics.com; `img-src 'self'
i.pinimg.com; `report-uri /__/csp_report/

<img src=i.pinimg.com/pin.png />
<img src=third-party.com/pin.png />
Benefits of CSP
Central inventory of third parties used client side

- Important for responding to requests or ensuring compliance with new requirements
- More comprehensive than scanning based approaches
Gating function for onboarding new third parties

- Ensure appropriate legal and security stakeholders are involved before changes happen
Different policies based on the user consent and type

User opted-out of analytics

User opted-in to analytics

Content-Security-Policy: default-src 'self' blob: s.pinimg.com; script-src 'self'; report-uri /__/csp_report/

Content-Security-Policy: default-src 'self' blob: s.pinimg.com; script-src 'self'; report-uri /__/csp_report/

*example-analytics.com
Deployment
Deploying a CSP for privacy

Come up with an initial policy
If you know your website you may be able to do this manually.

You can alternatively open developer tools and click around your website to see what is loaded or install a browser extension to do this automatically.

Ensure you have a default-src and don’t use * in any of your directives.

Report only mode
Put your policy inside a header like:

\[
\text{Content-Security-Policy-Report-Only:}
\text{default-src 'self' blob: s.pinimg.com; script-src 'self' *.pinterest.com *.example-analytics.com; img-src 'self' i.pinimg.com; report-uri /_/csp_report/}
\]

This will send reports to the url listed in report-uri of any violations.

Enforce the policy
Put your policy inside a header like:

\[
\text{Content-Security-Policy: default-src 'self' blob: s.pinimg.com; script-src 'self' *.pinterest.com *.example-analytics.com; img-src 'self' i.pinimg.com; report-uri /_/csp_report/?enforce}
\]

Continue to monitor your reports to detect any issues. You can add query string parameters or custom fields to your report uri to help differentiate these reports if needed.
Nonces or hashes in your CSP script-src are better for securing against XSS.

When you use nonces, you can’t use an allowlist in the same policy.

Solution: send multiple CSPs comma separated (does not work for safari <15.6)
Specific high risk integrations (and directives)

**Javascript (script-src)** - Full access to everything on the page and any actions your user can perform

**Iframes (frame-src)** - Ability to add other third parties, can use local storage and client side fingerprinting
Limitations

- Same domain used for multiple purposes
- Iframes can include third parties outside the CSP
Summary

- Central inventory of third parties used client side
- Gating function for onboarding new third parties
- Allows central blocking of certain assets per consent option or user type
Thank you!