Reining in the Web with Content Security Policy

Sid Stamm
Brandon Sterne
Gervase Markham

Mozilla
Mash-ups Anyone?

But how do I stop malicious content?
DOM attacks and Defacement

Content Injection
XSS

All your page is belong to us!
Filtering is Hard!
Mutual Approval can be Expensive!

He said no.

Really?

NO!

may I?

ask bob

policy

may I?
In-Band Policies are Dangerous!

Javascript that polices itself?
Is that like an application that tells you if it is a virus?
Goals

- Control of Site Content
- Protection against XSS
- Clickjacking Avoidance
- Increased Security
- Feasible Use
Control of Site Content

Document “Good” behavior...
Suppress the “Bad”
Grabbing the Reins

- Content Rules & Regulations
- Specify a “Normal Behavior” Policy
- Catch and Block Violations
Part 1: Smooth Edges

• Scripts served in files (not inline)
  - “javascript:” URIs
  - `<tag on*=...>` event registration
  - text nodes in `<script>` tags

• Establish Code / Data Separation
  - `eval("foo")` and friends
Part 2: Content Restrictions

• Block requests for all resources
  … unless explicitly allowed by a policy!
CSP: Policies

HTTP Response Header
X-CONTENT-SECURITY-POLICY

Directives to enforce listed within
Speed Bump

<meta http-equiv=....>?

- Designers may not have access to HTTP
- Two entities want restrictions
- Multiple policies?
Speed Bump

Intersecting Policies

Given Policies P1 and P2:
\[ P_e = \{ u \mid P_1 \text{ allows } u \text{ AND } P_2 \text{ allows } u \} \]
Speed Bump

`<meta http-equiv=....>`?

- policy in-band is too dangerous
- Multiple header instances!
CSP: Directives

report-uri
policy-uri
options

source directives
CSP: Source Directives

`allow (default for these)`

- `img-src`
- `media-src`
- `script-src`
- `object-src`
- `frame-src`
- `font-src`
- `xhr-src`
- `frame-ancestors`
- `style-src`
Speed Bump

- ‘self’ ... in pieces?

https://‘self’:443
‘self’://foo.com
foo.com:‘self’
‘self’

bar.com:8080 -> http://bar.com:8080
bar.com -> http://bar.com:80
Speed Bump

- Redirects

- http://foo.com
- http://bar.com
- http://duh.com
Goals (revisited)

- Control of Site Content
- Protection against XSS
- Clickjacking Avoidance
- Only Increased Security
- Feasible Use
Goals (revisited)

• Control of Site Content

Expressive white-list policy language
Goals (revisited)

• Protection against XSS

Only load scripts in external (whitelisted) files
Goals (revisited)

- Clickjacking Avoidance

frame-ancestors
Goals (revisited)

• Only Increased Security

Declarative syntax that can only reduce capabilities
Goals (revisited)

• Feasible Use

(1) Built into Firefox nightlies
(2) Deployed as patch for for Mozilla Add-Ons site
(3) In progress for Wordpress

http://core.trac.wordpress.org/ticket/10237
Beneficial Effects

• Content homogenization (mixed content control)
• Data exfiltration (and CSRF) reduction
• Violation reports = early alert
CSP: Use Case 1

allow ‘self’

• Site wants all content to come from the same source (scheme, host, port)
CSP: Use Case 2

allow ‘self’; frame-src ads.net

- Site wants all content to come from the same source (scheme, host, port), except content in iframes may be served by a third-party advertising network.
CSP: Use Case 3

```
allow 'self'; img-src *;
object-src *.teevee.com;
script-src myscripts.com
```

- Auction site wants to allow images from anywhere, plugin content from a trusted media provider network, and scripts only from its server hosting sanitized JavaScript.
CSP: Use Case 4

allow https://*.x.com;

- Example site wants to force all content to be served via HTTPS on port 443, from any subdomain of example.com
Wait!
That breaks my site!

- Good Option: convert your site
- Less Good Option: disable parts of CSP
Ramping Up

• Disable some restrictions via options
• Report-Only mode
• “Writing a Policy” guide
• “Converting your Site” guide
• Maybe a policy recommendation tool?
Wordpress
Content Security Policy is a mechanism that prevents cross-site scripting and other content-injection attacks. It works by informing you of the sites you trust to serve content in your pages, and what specific types of content are expected. To provide protection in this way, CSP can be embedded in web pages, so be aware that WordPress plugins may experience broken functionality when CSP is enabled.

Enable CSP: [ ]

Trusted Sites:
- **Images**
  - Everyone
  - yoursitename.com
- **Script**
  - yoursitename.com
  - code.google.com
  - hackmill.com
- **Object**
  - yoursitename.com
  - Youtube
  - www.youtube.com
- **Style**
  - yoursitename.com

Verify that you trust the sites in the list above to serve content in your site. Remove any sites that you do not want to be trusted. Changes will not be saved until you click **Save Changes** below.
More Stuff


• Nightly Firefox http://nightly.mozilla.org

• Progress: https://bugzilla.mozilla.org/show_bug.cgi?id=csp

Now With CSP!!!