Know Your Own Risks: Content Security Policy Report Aggregation and Analysis

Ksenia Dmitrieva
Senior Consultant
Cigital, Inc.
@KseniaDmitrieva
Agenda

- What is Content Security Policy (CSP) and why do we need it?
- CSP reporting functionality
- Extracting important information from reports at scale
- CSP support and adoption
- Real world examples of policies
Pop Quiz

Which web application vulnerability was

- #4 in OWASP* Top 10 in 2003
- #1 in OWASP Top 10 in 2007
- #2 in OWASP Top 10 in 2010
- #3 in OWASP Top 10 in 2013

*CROSS-SITE SCRIPTING*

*OWASP – Open Web Application Security Project (https://www.owasp.org/)*
Traditional Methods of XSS Protection

- Reflected
  - Server-side Output Encoding
- Stored
  - Server-side Input Validation

DOM-based
Execution of DOM-based XSS

Traditional XSS

DOM-based XSS
What is Content Security Policy?

- Content Security Policy:
  - Restricts ad-hoc XSS vectors: inline scripts and eval-constructs
  - Imposes restrictions on resources based on their origin

- CSP defines a list of resource directives:
  - script-src
  - object-src
  - frame-src
  - img-src
  - media-src
  - etc.
Sample CSP Policies

- Policy is sent by the server as an HTTP header:
  

- Any malicious inline scripts or scripts hosted elsewhere will not be executed.

Can a page with the following policy load a CSS style sheet from http://wordpress.org?

`Content-Security-Policy: script-src 'self'; frame-src 'none'; object-src 'none'`
CSP Reporting

- Browsers supporting CSP send policy violation reports in JSON format to the server
- Violation report may contain data about the attack

```
{
    "csp-report": {
        "document-uri": "http://example.com/page.html",
        "referrer": "http://evil.example.com/",
        "blocked-uri": "http://evil.example.com/evil.js",
        "violated-directive": "script-src 'self' https://apis.google.com",
        "original-policy": "default-src 'self'; script-src 'self'
    }
}
```

Where the violation occurred
Where the attack is coming from
What the attacker is trying to do
Directive controlling the resource
CSP Reporting and Enforcing

- **Content-Security-Policy** header with report-uri enforces the policy.
- **Content-Security-Policy-Report-Only** header reports policy violations, but does not enforce the policy.

```
Content-Security-Policy-Report-Only: default-src 'self'; script-src 'self'
```

- Use both headers: one to enforce the old policy and another to test out the new policy.

```
Content-Security-Policy: default-src 'self' https://*.google.com;

Content-Security-Policy-Report-Only: default-src 'self' *.google.com;
script-src 'self' https://apis.google.com; frame-src 'self';
report-uri http://example.com/reporting/parser.php
```
Browser Dependent Report Formats

- Browsers use slightly different formats for the reports

<table>
<thead>
<tr>
<th>Google Chrome</th>
<th>Mozilla Firefox</th>
<th>Windows Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>document-uri</td>
<td>document-uri (with path)</td>
<td>?</td>
</tr>
<tr>
<td>referrer</td>
<td>referrer</td>
<td>Coming soon</td>
</tr>
<tr>
<td>blocked-uri: &quot;&quot;</td>
<td>blocked-uri (with path): 'self'</td>
<td>July 29, 2015</td>
</tr>
<tr>
<td>violated-directive</td>
<td>violated-directive</td>
<td></td>
</tr>
<tr>
<td>original-policy</td>
<td>original-policy</td>
<td></td>
</tr>
<tr>
<td>effective-directive</td>
<td>effective-directive</td>
<td></td>
</tr>
<tr>
<td>status-code</td>
<td>status-code</td>
<td></td>
</tr>
</tbody>
</table>

- Special tools are needed to process CSP reports at scale
Report Aggregation: Tools

- **Aggregate Data**
  - Normalize data coming from different browsers
  - Add extra fields: application name, user agent

- **Filter**
  - According to Twitter, 80% is noise
  - Filter out reports from browser plugins, proxy sites, ISP cache servers

- **Graph**
  - Twitter created custom CSP report aggregation tool – “highly proprietary and will never be open sourced”
  - Yelp uses Elasticsearch/Logstash/Kibana trio
  - Open source tool Caspr (http://www.caspr.io/)
Yelp’s Report Aggregation

Source: http://engineeringblog.yelp.com/2014/09/csp_reports_at_scale.html
Content Security Policy 1.0

Mitigate cross-site scripting attacks by whitelisting allowed sources of script, style, and other resources.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 10</td>
<td>8</td>
<td>31</td>
<td>36</td>
</tr>
<tr>
<td>Windows 11</td>
<td>9</td>
<td>38</td>
<td>40</td>
</tr>
<tr>
<td>Windows 10</td>
<td>10</td>
<td>39</td>
<td>40</td>
</tr>
<tr>
<td>Windows 11</td>
<td>11</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Edge</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>Chrome</td>
<td>31</td>
<td>36</td>
<td>37</td>
</tr>
<tr>
<td>Safari</td>
<td>39</td>
<td>40</td>
<td>42</td>
</tr>
<tr>
<td>Opera</td>
<td>37</td>
<td>39</td>
<td>41</td>
</tr>
<tr>
<td>iOS Safari</td>
<td>38</td>
<td>40</td>
<td>42</td>
</tr>
<tr>
<td>Opera Mini</td>
<td>39</td>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td>Android Browser</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>Chrome for Android</td>
<td>42</td>
<td>43</td>
<td>44</td>
</tr>
</tbody>
</table>

Source: http://caniuse.com/#feat=contentsecuritypolicy

Browser Support of CSP

Source: http://caniuse.com/#feat=contentsecuritypolicy
CSP Adoption

Number of sites using CSP out of Alexa Top 1 Million

2013

2014

https://www.veracode.com/blog/2014/03/guidelines-for-setting-security-headers
CSP Level 2

- Problem with CSP 1
  - Using CSP requires re-writing a whole application
  - Frameworks still use inline JavaScript and eval-constructs

- CSP Level 2 (W3C Candidate Recommendation, July 21, 2015)
  - Protecting inline JavaScript without re-writing an application:
    - Nonce-source directive
    - Hash-source directive
  - Replacing X-Frame-Options with frame-ancestors
Real World CSP Adoption Example: Facebook

Facebook uses CSP on www.facebook.com

Real World CSP Adoption Example: Yelp

Yelp uses CSP on www.yelp.com

Content-Security-Policy-Report-Only: report-uri
Real World CSP Adoption Example: Twitter

Twitter uses CSP on all their services

Apply What You Have Learned Today

- If you want to use reporting functionality of CSP, next week answer the two questions:
  - Do any of your applications use CSP?
  - If they do, do you collect and analyze violation reports?
- In the next three months do the following:

<table>
<thead>
<tr>
<th>Not Using CSP Yet</th>
<th>Already Using CSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run a pilot:</td>
<td>Start collecting and analyzing reports:</td>
</tr>
<tr>
<td>• Select an application developed from scratch.</td>
<td>• Add the <code>report-uri</code> directive to the policy.</td>
</tr>
<tr>
<td>• Understand technologies used by the application.</td>
<td>• Filter out the noise.</td>
</tr>
<tr>
<td>Are there any that don’t support CSP?</td>
<td>• Add extra information to easily identify the application/module.</td>
</tr>
<tr>
<td>• Use technologies that support CSP (Angular JS, Django).</td>
<td>• Provide analysis results to development teams and to the security team.</td>
</tr>
<tr>
<td>• Do not use ‘unsafe-eval’ and ‘unsafe-inline’.</td>
<td>• Use Report-Only option to test a new policy.</td>
</tr>
<tr>
<td>• Use Report-Only option to test the policy.</td>
<td></td>
</tr>
</tbody>
</table>
Summary

- CSP protects applications from all types of XSS, including DOM-based XSS.
- CSP 1 is hard to implement on existing applications, but can be added to the newly developed applications. CSP 2 is on its way!
- CSP violation reports provide essential data about the application running in the users’ browsers. Use it!
- Normalize and filter CSP reports before analyzing and plotting data.
Questions?

@KseniaDmitrieva
kdmitrieva@cigital.com
https://www.cigital.com/blog/