Christian Wenz

The Final Frontier: Security APIs in Modern Browsers

Same-Origin Policy (SOP)

Origin: cf. http://tools.ietf.org/html/rfc6454

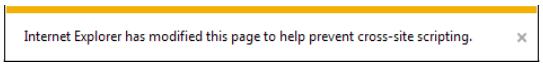
Protocol, domain, port

Does not apply to the src attribute – e.g <script>!

Browser Protection

Previously: XSS filters in IE8+, Chrome, Safari, Edge

Works for many attacks (but not all)
Can be deactivated using the *X-XSS-Protection: 0* HTTP header
Nowadays: IE only ;-)



Content Security Policy (CSP)

Version 1: W3C Candidate Recommendation back in 2012, but now „only" a Working Note (02/19/2012): https://www.w3.org/TR/CSP1/

Version 2: W3C Recommendation (12/15/2016): http://www.w3.org/TR/CSP2/

Version 3: W3C Working Draft (10/14/2022)/Editor's Draft (10/14/2022): https://www.w3.org/TR/CSP3/ / https://w3c.github.io/webappsec-csp/

Implementing Content Security Policy

Server sends HTTP header with policy

```
Content-Security-Policy
```

From CSP2: <meta> tag

```
<meta http-equiv="Content-Security-Policy"
      content="default src: 'self';">
```


CSP Structure

Content-Security-Policy:

```
default-src 'self';

img-src 'self' https://static.example.com;
```

Directives: Loading Ressources

default-src
child-src *
connect-src
font-src
frame-src **/***

img-src
media-src
object-src
script-src
style-src

* New in CSP 2
** Removed from CSP 2
*** Not covered by default-src

Values

*-src:
- *
- Origin
- URI (CSP 2+)
- 'self'
- 'none'
- 'unsafe-inline'
- 'unsafe-eval'

Re-Enabling Inline

Using a „number used once“
```
'nonce-abc123'
<script nonce="abc123">
```

Using a hash
```
'sha256-DZFspiNBPeeXMTdu4d8o5OdstdqahapOIKI6GxDqg3E='
```


Reporting

report-uri directive

Provides URI which receives reporting information in JSON format

Header *Content-Security-Policy-Report-Only* for reporting only (no blocking)

Reporting service: https://report-uri.com/

Subresource Integrity (SRI)

integrity attribute contains cryptographic digest of style sheet or JavaScript file

Browser verifies wiether the content matches that value (to avoid content injection via CDNs)

CSP directive *require-sri-for* enforces presence of a hash (directive values: *script*, *style*)

https://www.w3.org/TR/SRI/

Threat: Cookie Hijacking

Cookie Hijacking

Problem: Cookies are stolen

Cookie contains sensitive data or (more common) a session ID

Secure Cookies

Netscape´s original specification
http://curl.haxx.se/rfc/cookie_spec.html

Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME; **secure**
- Only send cookie via HTTPS

"New" option: **HttpOnly**
- *Almost* no access for JavaScript code

Encrypted Connections

HTTP Strict Transport Security (HSTS)
https://tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-14#section-6.1
Strict-Transport-Security HTTP header
Options:
- *max-age*
- *includeSubDomains*

Switches communication to HTTPS
Browsers come with preload list

HSTS Preload List

To put your site on the HSTS preload list:

Redirect from HTTP to HTTPS

Use HTTPS for all subdomains

Set *preload* option in header:
- *Strict-Transport-Security: max-age=63072000; includeSubDomains; preload*

Enforcing HTTPS with CSP

Content-Security-Policy update-insecure-requests

Automatically upgrade „insecure" requests (like http://) to secure ones (like https://)

Browser will send *Upgrade-Insecure-Requests* header to indicate its support for this feature

```
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36
```


Same-Site Cookies

Also: first party cookies

Protection levels for cross-site requests
- Lax: sends cookies for GET requests (and other methods that do not change app state)
- Strict: no cookies are sent
- None: no protection

Site != origin!

Demo

CSRF with user interaction: Clickjacking

Browser Protection

X-FRAME-OPTIONS HTTP header

Introduced in IE8 (!), also supported by all other browsers

Possible values:
- DENY: Page cannot be displayed in an iframe
- SAMEORIGIN: Page can only be displayed in an iframe from the same origin

CSP 2+: frame-ancestors directive

Threat: Data Leakage via Referrer

Referrer Policy

W3C Editor's Draft - https://w3c.github.io/webappsec-referrer-policy/

Referrer-Policy: origin

Other values: no-referrer, no-referrer-when-downgrade, same-origin, strict-origin, origin-when-cross-origin, strict-origin-when-cross-origin, unsafe-url

More Security Headers

Cross-Origin-{Embedder|Opener|Resource}-Policy

Expect-CT

Feature-Policy/Permissions-Policy

NEL

...

Feature Policy/Permissions Policy

Feature-Policy: geolocation 'none'; camera 'self' https://net.developerdays.pl

Permissions-Policy: geolocation=(), camera=(self "https://net.developerdays.pl")

One day: Document Policy

WebAuthn

Standardized API to authenticate web applications with public-key credentials

Including hardware support (e.g. Yubikey)

https://www.w3.org/TR/webauthn/

Thank you!

Questions?

Twitter: @chwenz

info@christianwenz.de

https://is.gd/MIhxaZ