Content Security Policy (CSP)
Quick Reference Guide

Content Security Policy Reference

The *new* `Content-Security-Policy` HTTP response header helps you reduce XSS risks on modern browsers by declaring what dynamic resources are allowed to load via a HTTP Header.

Tweet (https://twitter.com/intent/tweet?url=http%3A%2F%2Fcontent-security-policy.com%2F)

Edit on Github (https://github.com/foundeo/content-security-policy.com/)

Browser Support

Header					
`Content-Security-Policy` CSP Level 2	40+ Full January 2015	31+ *Partial* July 2014	10+	-	Edge 15 build 15002+
`Content-Security-Policy` CSP 1.0	25+	23+	7+	-	Edge 12 build 10240+
`X-Content-Security-Policy` Deprecated	-	4+	-	10+ *Limited*	12+ *Limited*

Header					
`X-Webkit-CSP` Deprecated	14+	-	6+	-	-

Sources: caniuse.com/contentsecuritypolicy (http://caniuse.com/contentsecuritypolicy), caniuse.com/contentsecuritypolicy2 (http://caniuse.com/contentsecuritypolicy2) & Mozilla (https://hacks.mozilla.org/2013/05/content-security-policy-1-0-lands-in-firefox-aurora/)

Try our CSP Browser Test (browser-test/) to test your browser.

Note: It is known that having both `Content-Security-Policy` and `X-Content-Security-Policy` or `X-Webkit-CSP` causes unexpected behaviours on certain versions of browsers. Please avoid using deprecated `X-*` headers.

Directive Reference

The `Content-Security-Policy` header value is made up of one or more directives (defined below), multiple directives are separated with a semicolon `;`

This documentation is provided based on the Content Security Policy 1.0 W3C Candidate Recommendation (http://www.w3.org/TR/2012/CR-CSP-20121115/)

Directive	Example Value	Description
`default-src`	`'self' cdn.example.com`	The `default-src` is the default policy for loading content such as JavaScript, Images, CSS, Fonts, AJAX requests, Frames, HTML5 Media. See the Source List Reference for possible values. CSP Level 1 25+ 23+ 7+ 12+
`script-src`	`'self' js.example.com`	Defines valid sources of JavaScript. CSP Level 1 25+ 23+ 7+ 12+
`style-src`	`'self' css.example.com`	Defines valid sources of stylesheets. CSP Level 1 25+ 23+ 7+ 12+
`img-src`	`'self' img.example.com`	Defines valid sources of images. CSP Level 1 25+ 23+ 7+ 12+
`connect-src`	`'self'`	Applies to `XMLHttpRequest` (AJAX), `WebSocket` or `EventSource`. If not allowed the browser emulates a `400` HTTP status code. CSP Level 1 25+ 23+ 7+ 12+

Directive	**Example Value**	**Description**
`font-src`	`font.example.com`	Defines valid sources of fonts. CSP Level 1 · 25+ · 23+ · 7+ · 12+
`object-src`	`'self'`	Defines valid sources of plugins, eg `<object>`, `<embed>` or `<applet>`. CSP Level 1 · 25+ · 23+ · 7+ · 12+
`media-src`	`media.example.com`	Defines valid sources of audio and video, eg HTML5 `<audio>`, `<video>` elements. CSP Level 1 · 25+ · 23+ · 7+ · 12+
`frame-src`	`'self'`	Defines valid sources for loading frames. `child-src` is preferred over this deprecated directive. Deprecated
`sandbox`	`allow-forms allow-scripts`	Enables a sandbox for the requested resource similar to the `iframe` `sandbox` attribute. The sandbox applies a same origin policy, prevents popups, plugins and script execution is blocked. You can keep the sandbox value empty to keep all restrictions in place, or add values: `allow-forms` `allow-same-origin` `allow-scripts` `allow-popups`, `allow-modals`, `allow-orientation-lock`, `allow-pointer-lock`, `allow-presentation`, `allow-popups-to-escape-sandbox`, and `allow-top-navigation` CSP Level 1 · 25+ · 50+ · 7+ · 12+
`report-uri`	`/some-report-uri`	Instructs the browser to POST reports of policy failures to this URI. You can also append `-Report-Only` to the HTTP header name to instruct the browser to only send reports (does not block anything). CSP Level 1 · 25+ · 23+ · 7+ · 12+
`child-src`	`'self'`	Defines valid sources for web workers and nested browsing contexts loaded using elements such as `<frame>` and `<iframe>` CSP Level 2 · 40+ · 45+ · 15+
`form-action`	`'self'`	Defines valid sources that can be used as a HTML `<form>` action. CSP Level 2 · 40+ · 36+ · 15+

Directive	**Example Value**	**Description**
`frame-ancestors`	`'none'`	Defines valid sources for embedding the resource using `<frame>` `<iframe>` `<object>` `<embed>` `<applet>`. Setting this directive to `'none'` should be roughly equivalent to `X-Frame-Options: DENY` CSP Level 2 39+ 33+ 15+
`plugin-types`	`application/pdf`	Defines valid MIME types for plugins invoked via `<object>` and `<embed>`. To load an `<applet>` you must specify `application/x-java-applet`. CSP Level 2 40+ 15+

Source List Reference

All of the directives that end with `-src` support similar values known as a source list. Multiple source list values can be space separated with the exception of `'none'` which should be the only value..

Source Value	**Example**	**Description**
`*`	`img-src *`	Wildcard, allows any URL except data: blob: filesystem: schemes.
`'none'`	`object-src 'none'`	Prevents loading resources from any source.
`'self'`	`script-src 'self'`	Allows loading resources from the same origin (same scheme, host and port).
`data:`	`img-src 'self' data:`	Allows loading resources via the data scheme (eg Base64 encoded images).
`example.com`	`img-src example.com`	Allows loading resources from the specified domain name.
`*.example.com`	`img-src *.example.com`	Allows loading resources from any subdomain under *`example.com`*.
`https://cdn.com`	`img-src https://cdn.com`	Allows loading resources only over HTTPS matching the given domain.
`https:`	`img-src https:`	Allows loading resources only over HTTPS on any domain.
`'unsafe-inline'`	`script-src 'unsafe-inline'`	Allows use of inline source elements such as style attribute, onclick, or script tag bodies (depends on the context of the source it is applied to) and `javascript:` URIs
`'unsafe-eval'`	`script-src 'unsafe-eval'`	Allows unsafe dynamic code evaluation such as JavaScript `eval()`

Source Value	Example	Description
`'nonce-'`	`script-src 'nonce-2726c7f26c'`	Allows `script` or `style` tag to execute if the `nonce` attribute value matches the header value. For example: `<script nonce="2726c7f26c">alert("hello");</script>`
`'sha256-'`	`script-src 'sha256-qzn...ng='`	Allow a specific script or style to execute if it matches the hash. Doesn't work for `javascript:` URIs. For example: `sha256-qznLcsROx4GACP2dm0UCKCzCG+HiZ1guq6ZZDob/Tng=` will allow `alert('Hello, world.');`

Content-Security-Policy Examples

Here a few common scenarios for content security policies:

Allow everything but only from the same origin

```
default-src 'self';
```

Only Allow Scripts from the same origin

```
script-src 'self';
```

Allow Google Analytics, Google AJAX CDN and Same Origin

```
script-src 'self' www.google-analytics.com ajax.googleapis.com;
```

Starter Policy

This policy allows images, scripts, AJAX, and CSS from the same origin, and does not allow any other resources to load (eg object, frame, media, etc). It is a good starting point for many sites.

```
default-src 'none'; script-src 'self'; connect-src 'self'; img-src 'self'; style-src 'self';
```

Content-Security-Policy Error Messages

In Chrome when a Content Security Policy Script Violation happens you get a message like this one in the *Chrome Developer Tools*:

```
Refused to load the script 'script-uri' because it violates the following Content Security Policy
```

In Firefox you might see messages like this in the *Web Developer Tools*:

```
Content Security Policy: A violation occurred for a report-only CSP policy ("An attempt to execute
```

In addition to a console message, a `securitypolicyviolation` event is fired on the window. See https://www.w3.org/TR/CSP2/#firing-securitypolicyviolationevent-events (https://www.w3.org/TR/CSP2/#firing-securitypolicyviolationevent-events).

Server Side Configuration

Any server side programming environment should allow you to send back a custom HTTP response header. You can also use your web server to send back the header.

Apache Content-Security-Policy Header

Add the following to your `httpd.conf` in your `VirtualHost` or in an `.htaccess` file:

```
Header set Content-Security-Policy "default-src 'self';"
```

Nginx Content-Security-Policy Header

In your `server {}` block add:

```
add_header Content-Security-Policy "default-src 'self';";
```

You can also append `always` to the end to ensure that nginx sends the header reguardless of response code.

IIS Content-Security-Policy Header

You can use the HTTP Response Headers GUI in IIS Manager or add the following to your web.config:

```
<system.webServer>
  <httpProtocol>
    <customHeaders>
      <add name="Content-Security-Policy" value="default-src 'self';" />
    </customHeaders>
  </httpProtocol>
</system.webServer>
```

CSP Resources

Want more info on CSP, checkout these links:

- CSP 1.0 Spec (http://www.w3.org/TR/CSP1/)
- CSP Level 2 Spec *(W3C Candidate Recommendation, 21 July 2015)* (http://www.w3.org/TR/CSP2/)
- CSP Presentations / Slides (presentations/)
- Mozilla Tutorial: Implementing Content Security Policy (https://hacks.mozilla.org/2016/02/implementing-content-security-policy/)
- Mozilla MDN Docs (https://developer.mozilla.org/en-US/docs/Web/Security/CSP/CSP_policy_directives)