此页面由社区从英文翻译而来。了解更多并加入 MDN Web Docs 社区。

内容安全策略(CSP)

内容安全策略 (CSP) 是一个额外的安全层，用于检测并削弱某些特定类型的攻击，包括跨站脚本 (XSS (en-US)) 和数据注入攻击等。无论是数据盗取、网站内容污染还是散发恶意软件，这些攻击都是主要的手段。

CSP 被设计成完全向后兼容（除CSP2 在向后兼容有明确提及的不一致；更多细节查看这里 章节1.1）。不支持CSP的浏览器也能与实现了CSP的服务器正常合作，反之亦然：不支持 CSP 的浏览器只会忽略它，如常运行，默认为网页内容使用标准的同源策略。如果网站不提供 CSP 头部，浏览器也使用标准的同源策略。

为使CSP可用, 你需要配置你的网络服务器返回 `Content-Security-Policy` HTTP头部（有时你会看到一些关于 `X-Content-Security-Policy` 头部的提法, 那是旧版本，你无须再如此指定它)。

除此之外， `<meta>` 元素也可以被用来配置该策略, 例如

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; img-src https://*; child-src 'none';">
```

威胁

跨站脚本攻击

CSP 的主要目标是减少和报告 XSS 攻击 ，XSS 攻击利用了浏览器对于从服务器所获取的内容的信任。恶意脚本在受害者的浏览器中得以运行，因为浏览器信任其内容来源，即使有的时候这些脚本并非来自于它本该来的地方。

CSP通过指定有效域——即浏览器认可的可执行脚本的有效来源——使服务器管理者有能力减少或消除XSS攻击所依赖的载体。一个CSP兼容的浏览器将会仅执行从白名单域获取到的脚本文件，忽略所有的其他脚本 (包括内联脚本和HTML的事件处理属性)。

作为一种终极防护形式，始终不允许执行脚本的站点可以选择全面禁止脚本执行。

数据包嗅探攻击

除限制可以加载内容的域，服务器还可指明哪种协议允许使用；比如 (从理想化的安全角度来说)，服务器可指定所有内容必须通过HTTPS加载。一个完整的数据安全传输策略不仅强制使用HTTPS进行数据传输，也为所有的cookie标记安全标识 cookies with the secure flag，并且提供自动的重定向使得HTTP页面导向HTTPS版本。网站也可以使用 `Strict-Transport-Security` HTTP头部确保连接它的浏览器只使用加密通道。

使用 CSP

配置内容安全策略涉及到添加 `Content-Security-Policy` HTTP头部到一个页面，并配置相应的值，以控制用户代理（浏览器等）可以为该页面获取哪些资源。比如一个可以上传文件和显示图片页面，应该允许图片来自任何地方，但限制表单的action属性只可以赋值为指定的端点。一个经过恰当设计的内容安全策略应该可以有效的保护页面免受跨站脚本攻击。本文阐述如何恰当的构造这样的头部，并提供了一些例子。

制定策略

你可以使用 `Content-Security-Policy` HTTP头部 来指定你的策略，像这样:

```
Content-Security-Policy: policy
```

policy参数是一个包含了各种描述你的CSP策略指令的字符串。

描述策略

一个策略由一系列策略指令所组成，每个策略指令都描述了一个针对某个特定类型资源以及生效范围的策略。你的策略应当包含一个 default-src 策略指令，在其他资源类型没有符合自己的策略时应用该策略(有关完整列表查看 default-src)。一个策略可以包含 default-src 或者 script-src (en-US) 指令来防止内联脚本运行, 并杜绝 eval() 的使用。 一个策略也可包含一个 default-src 或 style-src (en-US) 指令去限制来自一个 <style> 元素或者style属性的内联样式。

示例：常见用例

这一部分提供了一些常用的安全策略方案示例。

示例 1

一个网站管理者想要所有内容均来自站点的同一个源 (不包括其子域名)

```
Content-Security-Policy: default-src 'self'
```

示例 2

一个网站管理者允许内容来自信任的域名及其子域名 (域名不必须与CSP设置所在的域名相同)

```
Content-Security-Policy: default-src 'self' *.trusted.com
```

示例 3

一个网站管理者允许网页应用的用户在他们自己的内容中包含来自任何源的图片, 但是限制音频或视频需从信任的资源提供者(获得)，所有脚本必须从特定主机服务器获取可信的代码.

```
Content-Security-Policy: default-src 'self'; img-src *; media-src media1.com media2.com; script-src userscripts.example.com
```

在这里，各种内容默认仅允许从文档所在的源获取, 但存在如下例外:

- 图片可以从任何地方加载(注意 "*" 通配符)。
- 多媒体文件仅允许从 media1.com 和 media2.com 加载(不允许从这些站点的子域名)。
- 可运行脚本仅允许来自于userscripts.example.com。

示例 4

一个线上银行网站的管理者想要确保网站的所有内容都要通过SSL方式获取，以避免攻击者窃听用户发出的请求。

```
Content-Security-Policy: default-src https://onlinebanking.jumbobank.com
```

该服务器仅允许通过HTTPS方式并仅从 onlinebanking.jumbobank.com 域名来访问文档。

示例 5

一个在线邮箱的管理者想要允许在邮件里包含HTML，同样图片允许从任何地方加载，但不允许JavaScript或者其他潜在的危险内容(从任意位置加载)。

```
Content-Security-Policy: default-src 'self' *.mailsite.com; img-src *
```

注意这个示例并未指定script-src (en-US)。在此CSP示例中，站点通过 default-src 指令的对其进行配置，这也同样意味着脚本文件仅允许从原始服务器获取。

对策略进行测试

为降低部署成本，CSP可以部署为*报告(report-only)*模式。在此模式下，CSP策略不是强制性的，但是任何违规行为将会报告给一个指定的URI地址。此外，一个报告模式的头部可以用来测试一个修订后的未来将应用的策略而不用实际部署它。

你可以用 Content-Security-Policy-Report-Only HTTP 头部来指定你的策略，像这样：

```
Content-Security-Policy-Report-Only: policy
```

如果 `Content-Security-Policy-Report-Only` 头部和 `Content-Security-Policy` 同时出现在一个响应中，两个策略均有效。在 `Content-Security-Policy` 头部中指定的策略有强制性 ，而 `Content-Security-Policy-Report-Only` 中的策略仅产生报告而不具有强制性。

支持CSP的浏览器将始终对于每个企图违反你所建立的策略都发送违规报告，如果策略里包含一个有效的report-uri (en-US) 指令。

启用违例报告

默认情况下，违规报告并不会发送。为启用发送违规报告，你需要指定 report-uri (en-US) 策略指令，并提供至少一个URI地址去递交报告：

```
Content-Security-Policy: default-src 'self'; report-uri http://reportcollector.example.com/collector.cgi
```

然后你需要设置你的服务器能够接收报告，使其能够以你认为恰当的方式存储并处理这些报告。

违例报告的语法

作为报告的JSON对象报告包含了以下数据：

`document-uri`

发生违规的文档的URI。

`referrer`

违规发生处的文档引用（地址）。

`blocked-uri`

被CSP阻止的资源URI。如果被阻止的URI来自不同的源而非文档URI，那么被阻止的资源URI会被删减，仅保留协议，主机和端口号。

`violated-directive`

违反的策略名称。

`original-policy`

在 `Content-Security-Policy` HTTP 头部中指明的原始策略。

违例报告样本

我们假设页面位于 `http://example.com/signup.html` 。它使用如下策略，该策略禁止任何资源的加载，除了来自 `cdn.example.com`的样式表。

```
Content-Security-Policy: default-src 'none'; style-src cdn.example.com; report-uri /_/csp-reports
```

`signup.html` 的HTML像这样：

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sign Up</title>
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
    ... Content ...
  </body>
</html>
```

你能看出其中错误吗？样式表仅允许加载自 `cdn.example.com`，然而该页面企图从自己的源（`http://example.com`）加载。当该文档被访问时，一个兼容CSP的浏览器将以POST请求的形式发送违规报告到 `http://example.com/_/csp-reports` ，内容如下：

```
{
  "csp-report": {
    "document-uri": "http://example.com/signup.html",
```

```
    "referrer": "",
    "blocked-uri": "http://example.com/css/style.css",
    "violated-directive": "style-src cdn.example.com",
    "original-policy": "default-src 'none'; style-src cdn.example.com; report-uri /_/csp-reports"
  }
}
```

如你所见，该报告在 `blocked-uri字段` 中包含了违规资源的完整路径 ，但情况并非总是如此。比如，当signup.html试图从 `http://anothercdn.example.com/stylesheet.css` 加载CSS时，浏览器将不会包含完整路径，而只会保留源路径（`http://anothercdn.example.com`）。CSP技术规范小组对此古怪行为给出了解释 。大体上说，这样是为了防止泄露跨域资源的敏感信息。

浏览器兼容性

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	WebView Android	Chrome Android
Content-Security-Policy	Chrome25	Edge14	Firefox23	Internet Explorer 10	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.base-uri	Chrome40	Edge79	Firefox35	Internet Explorer No	Opera27	Safari10	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.block-all-mixed-content	ChromeYes	Edge79	Firefox48	Internet Explorer No	OperaYes	Safari?	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.child-src	Chrome40	Edge15	Firefox45	Internet Explorer No	Opera27	Safari10	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.connect-src	Chrome25	Edge14	Firefox23	Internet Explorer No	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.default-src	Chrome25	Edge14	Firefox23	Internet Explorer No	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.font-src	Chrome25	Edge14	Firefox23	Internet Explorer No	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.form-action	Chrome40	Edge15	Firefox36	Internet Explorer No	Opera27	Safari10	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.frame-ancestors	Chrome40	Edge15	Firefox33	Internet Explorer No	Opera26	Safari10	WebView Android ?	Chrome Android Yes
Content-Security-Policy.frame-src	Chrome25	Edge14	Firefox23	Internet Explorer No	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.img-src	Chrome25	Edge14	Firefox23	Internet Explorer No	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.manifest-src	ChromeYes	Edge79	Firefox41	Internet Explorer No	OperaYes	SafariNo	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.media-src	Chrome25	Edge14	Firefox23	Internet Explorer No	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
`<meta>` element support	ChromeYes	Edge18	Firefox45	Internet Explorer No	OperaYes	SafariYes	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.navigate-to	ChromeNo	EdgeNo	FirefoxNo	Internet Explorer No	OperaNo	SafariNo	WebView Android No	Chrome Android No
Content-Security-Policy.object-src	Chrome25	Edge14	Firefox23	Internet Explorer No	Opera15	Safari7	WebView Android Yes	Chrome Android Yes

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	WebView Android	Chrome Android
Content-Security-Policy.plugin-types	Chrome 40–90	Edge 15–90	FirefoxNo	Internet Explorer No	Opera 27–76	Safari10	WebView Android ?–90	Chrome Android ?–90
Content-Security-Policy.prefetch-src	ChromeNo	EdgeNo	FirefoxNo	Internet Explorer No	OperaNo	SafariNo	WebViewNo Android	ChromeNo Android
Content-Security-Policy.referrer	Chrome 33–56	EdgeNo	Firefox 37–62	Internet Explorer No	Opera ?–43	SafariNo	WebView Android 4.4.3–56	Chrome Android 33–56
Content-Security-Policy.report-sample	Chrome59	Edge79	Firefox?	Internet Explorer ?	Opera46	Safari15.4	WebView Android 59	Chrome Android 59
Content-Security-Policy.report-to	Chrome70	Edge79	FirefoxNo	Internet Explorer No	OperaNo	SafariNo	WebView Android 70	Chrome Android 70
Content-Security-Policy.report-uri	Chrome25	Edge14	Firefox23	Internet Explorer No	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.require-sri-for	Chrome54	Edge79	Firefox 49–68	Internet Explorer No	Opera41	SafariNo	WebView Android 54	Chrome Android 54
Content-Security-Policy.require-trusted-types-for	Chrome83	Edge83	FirefoxNo	Internet Explorer No	Opera69	SafariNo	WebView Android 83	Chrome Android 83
Content-Security-Policy.sandbox	Chrome25	Edge14	Firefox50	Internet Explorer 10	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.script-src	Chrome25	Edge14	Firefox23	Internet Explorer No	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
With external scripts	Chrome59	Edge79	Firefox?	Internet Explorer No	Opera?	Safari?	WebView Android 59	Chrome Android 59
Content-Security-Policy.script-src-attr	Chrome75	Edge79	FirefoxNo	Internet Explorer No	Opera62	SafariTP	WebView Android 75	Chrome Android 75
Content-Security-Policy.script-src-elem	Chrome75	Edge79	FirefoxNo	Internet Explorer No	Opera62	SafariTP	WebView Android 75	Chrome Android 75
Content-Security-Policy.strict-dynamic	Chrome52	Edge79	Firefox52	Internet Explorer No	Opera39	Safari15.4	WebView Android 52	Chrome Android 52
Content-Security-Policy.style-src	Chrome25	Edge14	Firefox23	Internet Explorer No	Opera15	Safari7	WebView Android Yes	Chrome Android Yes
Content-Security-Policy.style-src-attr	Chrome75	Edge79	FirefoxNo	Internet Explorer No	Opera62	SafariTP	WebView Android 75	Chrome Android 75
Content-Security-Policy.style-src-elem	Chrome75	Edge79	FirefoxNo	Internet Explorer No	Opera62	SafariTP	WebView Android 75	Chrome Android 75
Content-Security-Policy.trusted-types	Chrome83	Edge83	FirefoxNo	Internet Explorer No	Opera69	SafariNo	WebView Android 83	Chrome Android 83
Content-Security-Policy.unsafe-hashes	Chrome69	Edge79	FirefoxNo	Internet Explorer No	Opera56	Safari15.4	WebView Android 69	Chrome Android 69

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	WebView Android	Chrome Android
`Content-Security-Policy.upgrade-insecure-requests`	Chrome 43	Edge 17	Firefox 42	Internet Explorer No	Opera 30	Safari 10.1	WebView Android 43	Chrome Android 43
Worker support	Chrome Yes	Edge 79	Firefox 50	Internet Explorer No	Opera ?	Safari 10	WebView Android Yes	Chrome Android Yes
`Content-Security-Policy.worker-src`	Chrome 59	Edge 79	Firefox 58	Internet Explorer No	Opera 48	Safari 15.5	WebView Android 59	Chrome Android 59

Full support In development. Supported in a pre-release version. No support Compatibility unknown Experimental. Expect behavior to change in the future. Non-standard. Check cross-browser support before using. Deprecated. Not for use in new websites. See implementation notes. User must explicitly enable this feature. Uses a non-standard name.

参见

- `Content-Security-Policy`
- `Content-Security-Policy-Report-Only`
- Content Security in WebExtensions
- Display security and privacy policies In Firefox Developer Tools

Last modified: 2022年4月4日, by MDN contributors