Configuring Content Security Policy

Jenkins 1.641 / Jenkins 1.625.3 introduce the `Content-Security-Policy` header to static files served by Jenkins (specifically `DirectoryBrowserSupport`). This header is set to a very restrictive default set of permissions to protect Jenkins users from malicious HTML/JS files in workspaces, `/userContent`, or archived artifacts.

Unfortunately, several popular, useful plugins are affected by this and lose part of their functionality unless the default rules are relaxed.

Alternatives

Since Jenkins 2.200, it is possible to define a **Resource Root URL** in the Jenkins system configuration as an alternative to relaxing the Content Security Policy rules. See its inline help for details.

The Default Rule Set

The default rule is set to:

```
sandbox; default-src 'none'; img-src 'self'; style-src 'self';
```

This rule set results in the following:

- No JavaScript allowed at all
- No plugins (object/embed) allowed
- No inline CSS, or CSS from other sites allowed
- No images from other sites allowed
- No frames allowed
- No web fonts allowed
- No XHR/AJAX allowed
- etc.

In detail:

- `sandbox` limits a number of things of what the page can do, similar to the `sandbox` attribute set on iframes. For a full list of what is prohibited, see this site. This attribute is not widely supported.
- `default-src 'none'` prohibits loading scripts, URLs for AJAX/XHR/WebSockets/EventSources, fonts, plugin objects, media, and frames from anywhere (images and styles would also be prohibited, but are allowed by more specific rules described below).
- `img-src 'self'` allows loading images from other files served by Jenkins. Inline image definitions are prohibited.
- `style-src 'self'` allows loading style sheets from other files served by Jenkins. Inline style sheets are prohibited.

See content-security-policy.com for a reference on this header and its possible values.

Getting things working

The most expedient approach is to use Jenkins 2.200+ and set up a second domain pointing to the same Jenkins instance (Jenkins URL: build.example.com; Resource Root URL: build-artifacts.example.com). This will result in resources being served from the resource root URL instead of the Jenkins URL. The advantage of this is that there are no cookies associated with this domain, and file paths are hopefully sufficiently non predictable that people won't be able to exfiltrate content.

Considerations

The resource root URLs linked from Jenkins include individual secret keys which can be shared by users to people who don't otherwise have permission to access Jenkins. They have a site-wide configurable timeout.

Relaxing The Rules

This is highly discouraged. If `resource root URL` doesn't work for you, please reach out to the Jenkins team.

Considerations

It depends on the specific Jenkins setup whether relaxing these rules substantially is safe.

The following needs to be taken into consideration:

- **Are less trusted users allowed to create or modify files in Jenkins workspaces?** Jenkins builds pull requests sent by untrusted users, or employ a security model that limits trust in users allowed to configure one or more jobs, this also affects in what way the CSP rule set should be relaxed: Anything allowed there could be abused by users with the ability to change files in workspaces or archived artifacts.
- **Are some slaves not fully trusted?** Even when Slave To Master Access Control is used to limit what slaves can do on the master node, the entire build directory on the master node is writable by slaves, with the exception of the build.xml file itself. Therefore any file stored in a build directory and served by Jenkins should be considered potentially unsafe.

If either of these are true, you should be very careful when relaxing the CSP rule set. If neither is true, and all users with the ability to change files in workspaces are fully trusted, as are all slave machines, then it should be safe to relax or even disable the CSP rules.

Implementation

The CSP header sent by Jenkins can be modified by setting the system property `hudson.model.DirectoryBrowserSupport.CSP`:

If its value is the **empty string**, e.g. `java -Dhudson.model.DirectoryBrowserSupport.CSP= -jar jenkins.war` then the header will not be sent at all.

This is potentially very unsafe and should only be used after reviewing the overall security setup.

Any other value will be used as the header value, e.g. `java -Dhudson.model.DirectoryBrowserSupport.CSP="sandbox; default-src 'self';" -jar jenkins.war`. See content-security-policy.com for a reference on this header and its possible values.

Changes to the system property will be effective immediately, so it's possible to set this system property temporarily via the Jenkins Script Console, allowing you to experiment with different values:

Set a custom value for the header:

```
System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "sandbox; default-src 'self';")
```

Unset the header:

```
System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")
```

Set the header to the default:

```
System.clearProperty("hudson.model.DirectoryBrowserSupport.CSP")
```

Find out the current header value:

```
System.getProperty("hudson.model.DirectoryBrowserSupport.CSP")
```

How to interpret the output:

- No output below "Result" header: default protection
- Output `Result:` below "Result" header: protection disabled
- Output `Result: some text here` below "Result" header: custom protection

Forcing an uncached reload ("Shift-F5" or equivalent) of the affected web page may be necessary after changing the system property for the behavior to change.

Making Plugins Work

Maven Integration Plugin

Maven Integration Plugin has a feature that allows browsing generated Maven documentation sites (e.g. `site:site`) in Jenkins. When using this feature, it may be necessary to relax the CSP rule set to allow this to work. In limited testing, it was necessary to at least allow `style-src 'unsafe-inline'`. Depending on the site's content, more relaxed rules may be necessary.

```
sandbox; default-src 'none'; img-src 'self'; style-src 'self' 'unsafe-inline';
```

Javadoc Plugin

The Javadoc Plugin makes Javadoc available for browsing in Jenkins. The default rule set does not allow use of frames in pages served by Jenkins. To make this work again, the directives `frame-src 'self'` and `child-src 'self'` must be added to the CSP header. It appears Safari also requires the `sandbox` directive to be removed.

```
default-src 'none'; img-src 'self'; style-src 'self'; child-src 'self'; frame-src 'self';
```

To see the `ALL CLASSES` link when browsing Javadoc without frames, `script-src 'unsafe-inline'` must also be added to the CSP header.

HTML Publisher Plugin

Make sure to update HTML Publisher Plugin to version 1.10 to make it work with Content Security Policy

From version 1.10 on, the HTML Publisher Plugin is compatible with Content Security Policy. Before that, it executed inline JavaScript in a file served by `DirectoryBrowserSupport` to set up the frame wrapper around the published files and would fail unless `script-src 'unsafe-inline'` was allowed, which is a possible security issue.

If the published HTML files require JavaScript or other dynamic features prohibited by Content Security Policy to work properly, the `Content-Security-Policy` header will need to be adjusted accordingly. This applies to all versions of HTML Publisher Plugin.