Django-CSP Documentation

Release 2.0

James Socol, Mozilla

Feb 24, 2020

Contents

django-csp adds Content-Security-Policy headers to Django applications.

Version 2.0

Code https://github.com/mozilla/django-csp

License BSD; see LICENSE file

Issues https://github.com/mozilla/django-csp/issues

Contents:

CHAPTER 1

Installing django-csp

First, install django-csp via pip or from source:

```
# pip
$ pip install django-csp
```

```
# source
$ git clone https://github.com/mozilla/django-csp.git
$ cd django-csp
$ python setup.py install
```

Now edit your project's `settings` module. If you are not using the built in report processor, all you need to do is:

```
MIDDLEWARE_CLASSES = (
    # ...
    'csp.middleware.CSPMiddleware',
    # ...
)
```

That should do it! Go on to configuring CSP.

CHAPTER 2

Configuring django-csp

Content-Security-Policy is a complicated header. There are many values you may need to tweak here.

Note: Note when a setting requires a tuple or list. Since Python strings are iterable, you may get very strange policies and errors.

It's worth reading the latest CSP spec and making sure you understand it before configuring django-csp.

2.1 Policy Settings

These settings affect the policy in the header. The defaults are in *italics*.

Note: The "special" source values of `'self'`, `'unsafe-inline'`, `'unsafe-eval'`, and `'none'` must be quoted! e.g.: `CSP_DEFAULT_SRC = ("'self'",)`. Without quotes they will not work as intended.

CSP_DEFAULT_SRC Set the `default-src` directive. A tuple or list of values, e.g. `("'self'", 'cdn.example.net')`. *'self'*

CSP_IMG_SRC Set the `img-src` directive. A tuple or list. *None*

CSP_OBJECT_SRC Set the `object-src` directive. A tuple or list. *None*

CSP_MEDIA_SRC Set the `media-src` directive. A tuple or list. *None*

CSP_FRAME_SRC Set the `frame-src` directive. A tuple or list. *None*

CSP_FONT_SRC Set the `font-src` directive. A tuple or list. *None*

CSP_CONNECT_SRC Set the `connect-src` directive. A tuple or list. *None*

CSP_STYLE_SRC Set the `style-src` directive. A tuple or list. *None*

CSP_SANDBOX Set the `sandbox` directive. A tuple or list. *None*

CSP_REPORT_URI Set the `report-uri` directive. A **string** with a full or relative URI.

2.1.1 Changing the Policy

The policy can be changed on a per-view (or even per-request) basis. See the decorator documentation for more details.

2.2 Other Settings

These settings control the behavior of django-csp. Defaults are in *italics*.

CSP_REPORT_ONLY Send "report-only" headers instead of real headers. See the spec and the chapter on reports for more info. A boolean. *False*

CSP_EXCLUDE_URL_PREFIXES A **tuple** of URL prefixes. URLs beginning with any of these will not get the CSP headers. *('/admin',)*

> **Warning:** Excluding any path on your site will eliminate the benefits of CSP everywhere on your site. The typical browser security model for JavaScript considers all paths alike. A Cross-Site Scripting flaw on, e.g., *admin/* can therefore be leveraged to access everything on the same origin.

CHAPTER 3

Modifying the Policy with Decorators

Content Security Policies should be restricted and paranoid by default. You may, on some views, need to expand or change the policy. django-csp includes four decorators to help.

3.1 @csp_exempt

Using the `@csp_exempt` decorator disables the CSP header on a given view.

```
from csp.decorators import csp_exempt

# Will not have a CSP header.
@csp_exempt
def myview(request):
    return render(...)
```

You can manually set this on a per-response basis by setting the `_csp_exempt` attribute on the response to `True`:

```
# Also will not have a CSP header.
def myview(request):
    response = render(...)
    response._csp_exempt = True
    return response
```

3.2 @csp_update

The `@csp_update` header allows you to **append** values to the source lists specified in the settings. If there is no setting, the value passed to the decorator will be used verbatim.

Note: To quote the CSP spec: "There's no inheritance; ... the default list is not used for that resource type" if it is set. E.g., the following will not allow images from 'self':

```
default-src 'self'; img-src imgsrv.com
```

The arguments to the decorator the same as the settings without the `CSP_` prefix, e.g. `IMG_SRC`. (They are also case-insensitive.) The values are either strings, lists or tuples.

```
from csp.decorators import csp_update

# Will allow images from imgsrv.com.
@csp_update(IMG_SRC='imgsrv.com')
def myview(request):
    return render(...)
```

3.3 @csp_replace

The `@csp_replace` decorator allows you to **replace** a source list specified in settings. If there is no setting, the value passed to the decorator will be used verbatim. (See the note under `@csp_update`.)

The arguments and values are the same as `@csp_update`:

```
from csp.decorators import csp_replace

# settings.CSP_IMG_SRC = ['imgsrv.com']
# Will allow images from imgsrv2.com, but not imgsrv.com.
@csp_replace(IMG_SRC='imgsrv2.com')
def myview(request):
    return render(...)
```

3.4 @csp

If you need to set the entire policy on a view, ignoring all the settings, you can use the `@csp` decorator. The arguments and values are as above:

```
from csp.decorators import csp

@csp(DEFAULT_SRC="'self'", IMG_SRC='imgsrv.com',
     SCRIPT_SRC=['scriptsrv.com', 'googleanalytics.com'])
def myview(request):
    return render(...)
```

CHAPTER 4

CSP Violation Reports

When something on a page violates the Content-Security-Policy, and the policy defines a `report-uri` directive, the user agent may POST a report. Reports are JSON blobs containing information about how the policy was violated.

CHAPTER 5

Contributing

Patches are more than welcome! You can find the issue tracker on GitHub and we'd love pull requests.

5.1 Style

Patches should follow PEP8 and should not introduce any new violations as detected by the flake8 tool.

5.2 Tests

Patches fixing bugs should include regression tests (ideally tests that fail without the rest of the patch). Patches adding new features should test those features thoroughly.

To run the tests, install the requirements from `requirements.txt` (probably into a virtualenv):

```
pip install -r requirements.txt
```

Then just use the test-running shell script:

```
./run.sh test
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search