django-elevate Documentation

Release 2.0.3

Justin Mayer

Jul 28, 2022

Contents

1 What is this for? 3

2 Installation 5

3 Compatibility 7

4 Contents 9

4.1 Getting Started . . . 9

4.2 Configuration . . . 10

4.3 Usage . . . 11

4.4 Contributing . . . 12

4.5 How does this work? . . . 13

4.6 Security . . . 13

4.7 Changelog . . . 14

Python Module Index 15

Index 17

Elevate, also known as `django-elevate`, is an implementation of GitHub's Sudo Mode for Django.

CHAPTER 1

What is this for?

Elevate provides an extra layer of security beyond initial user authentication. Views can be decorated with `@elevate_required`, and then users must re-authenticate to access that resource. This might be useful for deleting objects, canceling subscriptions, and other sensitive operations. After re-authentication, the user has elevated permissions for the duration of `ELEVATE_COOKIE_AGE`. This duration is independent of the normal session duration, allowing for short elevated permission durations while still retaining long user sessions.

CHAPTER 2

Installation

```
$ pip install django-elevate
```

CHAPTER 3

Compatibility

- Django 2.2, 3.2, and 4.0
- Python 3.7 - 3.10
- pypy3

CHAPTER 4

Contents

4.1 Getting Started

4.1.1 Installation

First, install the `django-elevate` library with pip.

```
$ pip install django-elevate
```

Next, we need to add the `elevate` application to our `INSTALLED_APPS`. Installing the application will automatically register the `user_logged_in` and `user_logged_out` signals that are needed.

```
INSTALLED_APPS = (
    # ...
    'elevate',
)
```

Now we need to add Elevate's middleware to the `MIDDLEWARE` setting:

```
MIDDLEWARE = (
    # ...
    'elevate.middleware.ElevateMiddleware',
)
```

Note: `elevate.middleware.ElevateMiddleware` **must** be installed after `django.contrib.session.middleware.SessionMiddleware`.

Proceed to the *Configuration* documentation.

4.2 Configuration

4.2.1 Settings

By default, all of the settings are optional and define sane and secure defaults.

ELEVATE_URL The url or view name for the Elevate view. *Default: elevate.views.elevate*

ELEVATE_REDIRECT_URL Default url to be redirected to after elevating permissions. *Default: /*

ELEVATE_REDIRECT_FIELD_NAME The querystring argument to be used for redirection. *Default: next*

ELEVATE_COOKIE_AGE How long should Elevate mode be active for? Duration in seconds. *Default: 10800*

ELEVATE_COOKIE_DOMAIN The domain to bind the Elevate cookie to. *Default: current exact domain.*

ELEVATE_COOKIE_HTTPONLY Should the cookie only be accessible via http requests? *Default: True*

> **Note:** If this is set to `False`, any JavaScript files have the ability to access this cookie, so this should only be changed if you have a good reason to do so.

ELEVATE_COOKIE_NAME The name of the cookie to be used for Elevate mode. *Default: elevate*

ELEVATE_COOKIE_PATH Restrict the Elevate cookie to a specific path. *Default: /*

ELEVATE_COOKIE_SECURE Only transmit the Elevate cookie over https if True. *Default: matches current protocol*

> **Note:** By default, we will match the protocol that made the request. So if your Elevate page is over https, we will set the `secure` flag on the cookie so it won't be transmitted over plain http. It is highly recommended that you only use `django-elevate` over https.

ELEVATE_COOKIE_SALT An extra salt to be added into the cookie signature. *Default: ‘’*

ELEVATE_REDIRECT_TO_FIELD_NAME The name of the session attribute used to preserve the redirect destination between the original page request and successful elevated login. *Default: elevate_redirect_to*

ELEVATE_TOKEN_LENGTH Length of the random string that is stored in the Elevate cookie. *Default: 12*

4.2.2 Set up URLs

We need to hook up one url to use `django-elevate` properly. At minimum, you need something like the following:

```
from elevate.views import elevate as elevate_view

(r'^elevate/$',  # Whatever path you want
    elevate_view,  # Required
    {'template_name': 'elevate/elevate.html'}  # Optionally change the template to be␣
↪used
)
```

4.2.3 Required Template

To get up and running, we last need to create a template for the Elevate page to render. By default, the package will look for `elevate/elevate.html` but can easily be overwritten by setting the `template_name` when defining the url definition as seen above.

elevate/elevate.html

This template gets rendered with the the following context:

form An instance of `ElevateForm`.

ELEVATE_REDIRECT_FIELD_NAME The value of `?next=/foo/`. If `ELEVATE_REDIRECT_FIELD_NAME` is `name`, then expect to find `{{ next }}` in the context, with the value of `/foo/`.

After configuring things, we can now *start securing pages*.

4.3 Usage

Once we have `django-elevate` *installed* and *configured*, we need to decide which views should be secured.

`elevate.decorators.`**`elevate_required`**`()`

The meat of `django-elevate` comes from decorating your views with `@elevate_required` much in the same way that `@login_required` works.

Let's pretend that we have a page on our site that has sensitive information that we want to make extra sure that a user is allowed to see it:

```
from elevate.decorators import elevate_required

@login_required  # Make sure they're at least logged in
@elevate_required  # On top of being logged in, are you in Elevate mode?
def super_secret_stuff(request):
    return HttpResponse('your social security number')
```

That's it! When a user visits this page and they don't have the correct permission, they'll be redirected to a page and prompted for their password. After entering their password, they'll be redirected back to this page to continue on what they were trying to do.

`class` `elevate.mixins.`**`ElevateMixin`**

`ElevateMixin` provides an easy way to elevate a class-based view. Any view that inherits from this mixin is automatically wrapped by the `@elevate_required` decorator.

This works well with the `LoginRequiredMixin` from django-braces:

```
from django.views import generic
from braces.views import LoginRequiredMixin
from elevate.mixins import ElevateMixin

class SuperSecretView(LoginRequiredMixin, ElevateMixin, generic.TemplateView):
    template_name = 'secret/super-secret.html'
```

`request.`**`is_elevated`**`()`

Returns a boolean to indicate if the current request is in Elevate mode or not. This gets added on by the *`ElevateMiddleware`*. This is an shortcut for calling *`has_elevated_privileges()`* directly.

`class` `elevate.middleware.`**`ElevateMiddleware`**

By default, you just need to add this to your `MIDDLEWARE` list.

`has_elevated_privileges``(`*self*, *request*`)`

Subclass and override `has_elevated_privileges()` if you'd like to override the default behavior of *`request.is_elevated()`*.

`process_request``(`*self*, *request*`)`

Adds *is_elevated()* to the request.

process_response(*self*, *request*, *response*)

Controls the behavior of setting and deleting the Elevate cookie for the browser.

elevate.utils.**grant_elevated_privileges**(*request*, *max_age=ELEVATE_COOKIE_AGE*)

Assigns a random token to the user's session that allows them to have elevated permissions.

```
from elevate.utils import grant_elevated_privileges
token = grant_elevated_privileges(request)
```

elevate.utils.**revoke_elevated_privileges**(*request*)

Revoke elevated privileges from a request explicitly

```
from elevate.utils import revoke_elevated_privileges
revoke_elevated_privileges(request)
```

elevate.utils.**has_elevated_privileges**(*request*)

Check if a request is allowed to perform elevated actions.

```
from elevate.utils import has_elevated_privileges
has_elevate = has_elevated_privileges(request)
```

4.4 Contributing

4.4.1 Getting the Source

You will first want to clone the source repository locally with `git`:

```
$ git clone git@github.com:justinmayer/django-elevate.git
```

4.4.2 Setting Up the Environment

I would recommend using virtualenv to set up a dev environment. After creating an environment, install all development dependencies with:

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

4.4.3 Running Tests

Tests are run using pytest and can be found inside `tests/*`.

Tests can simply be run using:

```
$ pytest
```

This will discover and run the test suite using your default Python interpreter. To run tests for all supported platforms, we use tox.

```
$ tox
```

4.4.4 Submitting Patches

Patches are accepted via pull requests on GitHub. Please be sure to add a `RELEASE.md` file in the root of the project that contains the release type (major, minor, patch) and a summary of the changes that will be used as the release changelog entry. For example:

```
Release type: patch

Remove vendored copy of is_safe_url
```

Note: If you are submitting a security patch, please see our *Security* page for special instructions.

Tests

All new code and changed code must come with **100%** test coverage to be considered for acceptance.

4.5 How does this work?

`django-elevate` works by setting an additional cookie that must match a secret value in your session. This cookie is ideally set to a shorter TTL than the normal session. When not in Elevate mode, any view that is decorated with `@elevate_required` will require the user to re-enter their password. Once in Elevate mode, they won't be prompted to enter their password for the next `ELEVATE_COOKIE_AGE` seconds.

In practice, we want to serve this Elevate cookie over https only to avoid a man-in-the-middle attack where someone hijacks this cookie. This can be utilized safely in situations where the sessionid cookie is being transmitted over http, but we want to make sure that secure areas of our site are not accessible with just the sessionid.

- When logging in, `django-elevate` automatically elevates your permission to `Elevate mode`.
- A second cookie is sent to your browser (by default this cookie is named `elevate` but can be set to anything with `ELEVATE_COOKIE_NAME`). This cookie contains a randomly generated string of characters.
- The same randomly generated string of characters is stored in the user's session.
- On subsequent requests, the cookie value must match the value that was stored in the session. If the values do not match, or the cookie is not sent at all, the user will be redirected to a page to re-enter their password.
- If they re-enter their password successfully, a new cookie is set and their permissions are again elevated.

Note: The best way to secure your site and your users is to use https. `django-elevate` won't be able to help you if it's being served over http.

4.6 Security

We take the security of `django-elevate` seriously. If you believe you've identified a security vulnerability, please report it to Justin Mayer.

Once you've submitted an issue via email, you should receive an acknowledgement within 48 hours, and depending on the action to be taken, you may receive further follow-up messages.

4.7 Changelog

4.7.1 2.0.3 - 2022-07-28

- Add Django 4.0 support - Add *ELEVATE_TOKEN_LENGTH* setting as *get_random_string* no longer has a default length
- Remove Django 3.1 from test matrix
- Remove Python 3.6 from test matrix
- No longer build wheel as universal as Python 2 is not supported

4.7.2 2.0.2 - 2021-06-02

Added *request* to the *authenticate()* call to prevent errors from authenication backends that require it.

4.7.3 2.0.1 - 2021-04-11

Add Django 3.2 support and remove Django 3.0 support

4.7.4 2.0.0 - 2020-10-25

- Drop support for Python 2 and Python 3.4
- Drop support for Django 1.8-2.1
- Add support for Python 3.8 and 3.9
- Add support for Django 3.0 and 3.1
- Removed code that was required to support older versions of Python and Django

4.7.5 1.0.1 - 2019-06-06

- Add support for Django 2.1 and 2.2
- Add Python 3.7, and drop Python 3.3, from test matrix

4.7.6 1.0.0 - 2018-06-25

- Add support for Django 2.0, 1.11, and 1.10
- Auto-focus input on password field
- Fork and rename project

Python Module Index

e

elevate.utils, 12

Index

E

elevate.decorators.elevate_required() (*built-in function*), 11

elevate.middleware.ElevateMiddleware (*built-in class*), 11

elevate.mixins.ElevateMixin (*built-in class*), 11

elevate.utils (*module*), 12

G

grant_elevated_privileges() (*in module elevate.utils*), 12

H

has_elevated_privileges() (*elevate.middleware.ElevateMiddleware method*), 11

has_elevated_privileges() (*in module elevate.utils*), 12

I

is_elevated() (*request method*), 11

P

process_request() (*elevate.middleware.ElevateMiddleware method*), 11

process_response() (*elevate.middleware.ElevateMiddleware method*), 12

R

revoke_elevated_privileges() (*in module elevate.utils*), 12