django-sesame Documentation

Release 1.2

Aymeric Augustin

December 12, 2016

Contents

CHAPTER 1

Introduction

django-sesame provides one-click login for your Django project. It uses specially crafted URLs containing an authentication token, for example: https://example.com/?url_auth_token=AAAAARchl18CIQUlImmbV9q7PZk%3A89AEU34b0JLSrkT8Ty2RPISio5

It's useful if you want to share private content without requiring your visitors to remember a username and a password or to go through an authentication process involving a third-party.

django-sesame is tested with:

- Django 1.8 (LTS), 1.9 and 1.10.
- all supported Python versions.

It requires `django.contrib.auth`. It's compatible with custom user models. It uses `django.contrib.session` when it's available.

Technically, django-sesame can provide stateless authenticated navigation without `django.contrib.sessions`, provided all internal links include the authentication token, but that increases the security issues explained below.

django-sesame is released under the BSD license, like Django itself.

CHAPTER 2

A few words about security

Before using django-sesame in your project, you should review the following advice carefully.

The major security weakness in django-sesame is a direct consequence of the feature it implements: **whoever obtains an authentication token will be able to log in to your website.** URLs end up in countless insecure places: emails, referer headers, proxy logs, browser history, etc. You can't avoid that. So use django-sesame only for mundane things.

If a data leak would affect you in non-trivial ways, don't use this library. You have been warned.

Otherwise, a reasonable attempt has been made to provide a secure solution. django-sesame uses Django's signing framework to create signed tokens.

Tokens are tied to the primary key and the password of the corresponding user. Changing the password invalidates the token. When the authentication backend uses salted passwords — that's been the default in Django for a long time — the token is invalidated even if the new password is identical to the old one.

By default, tokens never expire. If you want them to expire after a given amount of time, set the `SESAME_MAX_AGE` setting to a duration in seconds. Then each token will contain the time it was generated at and django-sesame will check if it's still valid at each login attempt.

CHAPTER 3

How to

1. Add `sesame.backends.ModelBackend` to `AUTHENTICATION_BACKENDS`:

```
AUTHENTICATION_BACKENDS += ['sesame.backends.ModelBackend']
```

2. Add `sesame.middleware.AuthenticationMiddleware` to `MIDDLEWARE`:

```
MIDDLEWARE += ['sesame.middleware.AuthenticationMiddleware']
```

   NB: in Django < 1.10, that setting was called `MIDDLEWARE_CLASSES`.

3. Generate authentication tokens with `sesame.get_query_string(user)`.

That's all!

CHAPTER 4

Generating URLs

django-sesame provides two functions to generate authenticated URLs.

1. `sesame.utils.get_query_string(user)` returns a complete query string that you can append to any URL to enable one-click login.
2. `sesame.utils.get_parameters(user)` returns a dictionary of GET parameters to add to the query string, if you're already building one.

Share resulting URLs with your users while ensuring adequate confidentiality.

By default, the URL parameter is called `url_auth_token`. You can set the `SESAME_TOKEN_NAME` setting to a shorter name that doesn't conflict with query string parameters used by your application.