Django-Cryptolock Documentation

Release 0.1.0

Gonçalo Valério

Mar 31, 2020

Contents

1 Django-Cryptolock **3**
- 1.1 Documentation . . . 3
- 1.2 Quickstart . . . 3
- 1.3 Credits . . . 4

2 Installation **5**

3 Usage **7**

4 Features and Roadmap **9**
- 4.1 Features . . . 9
- 4.2 Roadmap . . . 9

5 Contributing **11**
- 5.1 Types of Contributions . . . 11
- 5.2 Get Started! . . . 12
- 5.3 Running tests for specific environments . . . 13
- 5.4 Pull Request Guidelines . . . 13
- 5.5 Tips . . . 13

6 Credits **15**
- 6.1 Development Lead . . . 15
- 6.2 Contributors . . . 15

7 History **17**
- 7.1 0.1.0 (2020-03-31) . . . 17
- 7.2 0.0.2 (2020-01-08) . . . 17
- 7.3 0.0.1 (2019-11-25) . . . 17

Contents:

CHAPTER 1

Django-Cryptolock

Authenticatio using cryptocurrency wallets for Django projects

This package provided a django app containing a set of utilities to make easier to implement the BitId and Monero Cryptolock authentication "protocols".

Future releases might include other cryptocurrencies but for the being (until we reach some stability) the focus will continue BTC and XMR.

DISCLAIMER: This package is still in an early stage of development. It isn't meant to be used on any production scenario yet (in other words, only test projects for now).

1.1 Documentation

The full documentation is at https://django-cryptolock.readthedocs.io.

1.2 Quickstart

1. Install Django-Cryptolock:

```
pip install django-cryptolock
```

2. Add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'django_cryptolock.apps.DjangoCryptolockConfig',
```

(continues on next page)

(continued from previous page)

```
    ...
)
```

3. Migrate your database:

```
python manage.py migrate
```

4. Add the following settings to your project for the Monero Backend:

```
AUTHENTICATION_BACKENDS = [
    "django_cryptolock.backends.MoneroAddressBackend",
    ...
]
DJCL_MONERO_NETWORK = "mainnet"
DJCL_MONERO_WALLET_RPC_PROTOCOL = "<http_or_https>"
DJCL_MONERO_WALLET_RPC_HOST = "<wallet_rpc_host>:<port>"
DJCL_MONERO_WALLET_RPC_USER = "<user>"
DJCL_MONERO_WALLET_RPC_PASS = "<password>"
```

5. Add Django-Cryptolock's URL patterns:

```
from django.conf.urls import url


urlpatterns = [
    ...
    url(r"^auth/", include("django_cryptolock.urls", namespace="django_cryptolock")),
    ...
]
```

More detailed information can be found in the documentation.

1.3 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

At the command line:

```
$ pip install django-cryptolock
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-cryptolock
$ pip install django-cryptolock
```

CHAPTER 3

Usage

To use Django-Cryptolock in a project, add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'django_cryptolock.apps.DjangoCryptolockConfig',
    ...
)
```

Now you should add the auth backend you wish to use on your project. You can use one or more:

```
AUTHENTICATION_BACKENDS = [
    "django_cryptolock.backends.BitcoinAddressBackend",
    "django_cryptolock.backends.MoneroAddressBackend",
]
```

If you use Monero, currently the following extra settings are required:

```
DJCL_MONERO_NETWORK = "mainnet"  # mainnet, stagenet or testnet
DJCL_MONERO_WALLET_RPC_PROTOCOL = "<http_or_https>"
DJCL_MONERO_WALLET_RPC_HOST = "<wallet_rpc_host>:<port>"
DJCL_MONERO_WALLET_RPC_USER = "<user>"
DJCL_MONERO_WALLET_RPC_PASS = "<password>"
```

For Bitcoin, you only need to set the `DJCL_BITCOIN_NETWORK`:

```
DJCL_BITCOIN_NETWORK = "mainnet"  # mainnet or testnet
```

Add Django-Cryptolock's URL patterns:

```
from django.conf.urls import url


urlpatterns = [
    ...
```

(continues on next page)

(continued from previous page)

```
    url(r"^auth/", include("django_cryptolock.urls", namespace="django_cryptolock")),
    ...
]
```

This will add 2 routes :

- `django_cryptolock:signup`
- `django_cryptolock:login`

For usega within you templates. For specific auth pages you can create the template files (`login.html` and `signup.html`) under a `django_cryptolock` subfolder.

Both of these templates will have access to a ‘‘form‘‘‘containing the required fields for the authentication.

CHAPTER 4

Features and Roadmap

4.1 Features

Below are some of the features the package already supports:

- Authentication using BitId
- Authentication using Monero-Cryptolock
- Supports custom user models

4.2 Roadmap

- QR code generation
- Multiple login addresses per-user

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/dethos/django-cryptolock/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

Django-Cryptolock could always use more documentation, whether as part of the official Django-Cryptolock docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/dethos/django-cryptolock/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *django-cryptolock* for local development.

1. Fork the *django-cryptolock* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-cryptolock.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-cryptolock
$ cd django-cryptolock/
$ pip install -r requirements_dev.txt -r requirements_test.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass `black` and the tests, including testing other Python versions with `tox`:

   $ black –check django_cryptolock $ make test $ make test-all

   To get black and tox, just pip install them into your virtualenv.

6. If your changes are visible to the user, you can add a demo for them to the example project.
7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

5.3 Running tests for specific environments

Do want to test only a specific python version / django version locally?

You can use tox directly:

```
: :
```

source <YOURVIRTUALENV>/bin/activate (myenv) $ pip install tox (myenv) $ tox -e <your-python-version>-django-<22_or_30>

5.4 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6, 3.7 and 3.8. Check https://travis-ci.org/dethos/django-cryptolock/pull_requests and make sure that the tests pass for all supported Python versions.

5.5 Tips

To run a subset of tests:

```
$ pytest tests/test_models.py
```

CHAPTER 6

Credits

6.1 Development Lead

- Gonçalo Valério <gon@ovalerio.net>

6.2 Contributors

- Guy Willett - https://github.com/guywillett

CHAPTER 7

History

7.1 0.1.0 (2020-03-31)

- Add validation for existing addresses on the signup form
- Add rudimentary BitId support
- Renamed the base auth views to generic names

7.2 0.0.2 (2020-01-08)

- A default `urls.py` is provided by the package so can work "out-of-the-box".
- Default location for templates moved to `django_cryptolock` folder.
- Update quickstart guide.
- Update instructions to contribute to the project.
- Add `DJCL` namespace to all related settings.
- MoneroAddressBackend is now executed when more parameters are added to the `authenticate` function.

7.3 0.0.1 (2019-11-25)

- First release on PyPI.