django-pwned-passwords Documentation

Release 4.1.0

Jamie Counsell

Aug 08, 2019

Contents

Contents:

CHAPTER 1

Django PWNED Passwords

django-pwned-passwords is a Django password validator that checks Troy Hunt's PWNED Passwords API to see if a password has been involved in a major security breach before.

Note: This app currently sends a portion of a user's hashed password to a third party. Before using this application, you should understand how that impacts you.

1.1 Documentation

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

1.2 Requirements

- Django [1.9, 2.1]
- Python 2.7, [3.5, 3.6, 3.7]

1.3 Quickstart

Install django-pwned-passwords:

```
pip install django-pwned-passwords
```

Add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'django_pwned_passwords',
    ...
)
```

Add django-pwned-passwords's PWNEDPasswordValidator:

```
AUTH_PASSWORD_VALIDATORS = [
    ...
    {
        'NAME': 'django_pwned_passwords.password_validation.PWNEDPasswordValidator'
    }
]
```

1.4 Features

This password validator returns a ValidationError if the PWNED Passwords API detects the password in its data set. Note that the API is heavily rate-limited, so there is a timeout (`PWNED_VALIDATOR_TIMEOUT`).

If `PWNED_VALIDATOR_FAIL_SAFE` is True, anything besides an API-identified bad password will pass, including a timeout. If `PWNED_VALIDATOR_FAIL_SAFE` is False, anything besides a good password will fail and raise a ValidationError.

1.5 Settings

Setting	Description	Default
`PWNED_VALIDATOR_TIMEOUT`	The timeout in seconds. The validator will not wait longer than this for a response from the API.	`2`
`PWNED_VALIDATOR_FAIL_SAFE`	If the API fails to get a valid response, should we fail safe and allow the password through?	`True`
`PWNED_VALIDATOR_URL`	The URL for the API in a string format.	`https://haveibeenpwned.com/api/v2/pwnedpassword/{short_hash}`
`PWNED_VALIDATOR_ERROR`	The error message for an invalid password.	`"Your password was determined to have been involved in a major security breach."`
`PWNED_VALIDATOR_FAIL_ERROR`	The error message when the API fails. Note: this will only display if `PWNED_VALIDATOR_FAIL_SAFE` is *False*.	`"We could not validate the safety of this password. This does not mean the password is invalid. Please try again later."`
`PWNED_VALIDATOR_HELP_TEXT`	The help text for this password validator.	`"Your password must not have been detected in a major security breach."`
`PWNED_VALIDATOR_MINIMUM_BREACHES`	The minimum number of breaches needed to raise an error	`1`

1.6 Rate Limiting

Historically, requests to the API were rate limited. However, with the new k-anonymity model-based API, there are no such rate limits.

1.7 Running Tests

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install tox
(myenv) $ tox
```

1.8 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

Install via pip:

```
$ pip install django-pwned-passwords
```

Add it to your `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    ...
    'django_pwned_passwords',
    ...
)
```

Add django-pwned-passwords's `PWNEDPasswordValidator` to `AUTH_PASSWORD_VALIDATORS`:

```
AUTH_PASSWORD_VALIDATORS = [
    ...
    {
        'NAME': 'django_pwned_passwords.password_validation.PWNEDPasswordValidator'
    }
]
```

CHAPTER 3

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:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/jamiecounsell/django-pwned-passwords/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.

3.1.2 Fix Bugs

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

3.1.3 Implement Features

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

3.1.4 Write Documentation

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

3.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/jamiecounsell/django-pwned-passwords/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 :)

3.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/django-pwned-passwords.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-pwned-passwords
$ cd django-pwned-passwords/
$ python setup.py develop
```

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 flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 django_pwned_passwords tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. 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
```

7. Submit a pull request through the GitHub website.

3.3 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 2.7, 3.5, and 3.6, and for PyPy. Check https://travis-ci.org/jamiecounsell/django-pwned-passwords/pull_requests and make sure that the tests pass for all supported Python versions.

3.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_django_pwned_passwords
```

CHAPTER 4

Credits

4.1 Development Lead

- Jamie Counsell <jamiecounsell@me.com>

4.2 Contributors

None yet. Why not be the first?

CHAPTER 5

History

See Github Releases