OATH Toolkit Python Bindings Documentation

Release 2.0.dev1

Mark Lee

Sep 27, 2017

Contents

This package is a set of Python bindings for the OATH Toolkit library. Please note that it is *OATH* (open authentication, e.g., one-time passwords) and not *OAuth* (an open standard for authorization).

CHAPTER 1

Features

- Runs on a variety of Python versions/implementations
- QR code generator, compatible with apps like Google Authenticator
- Integration with WTForms
- Integration with Django via `django-otp`

CHAPTER 2

Usage

To generate a time-based one-time password (TOTP):

```
from oath_toolkit import TOTP
from time import time

digits = 6
time_step = 30
oath = TOTP(b'secret key', digits, time_step)
one_time_password = oath.generate(time())
```

To validate a HMAC-based one-time password (HOTP):

```
from oath_toolkit import HOTP
from oath_toolkit.exc import OATHError

def verify(otp, counter):
    digits = 6
    oath = HOTP(b'secret key', digits)
    try:
        return oath.verify(otp, counter)
    except OATHError:
        return False
```

For an explanation of terms like `time_step` and `counter`, refer to the *API documentation*.

More complex examples can be found in the `examples/` directory, which includes a port of the command-line app `oathtool`, a sample Django project, and a simple Flask app which shows how WTForms integration works.

CHAPTER 3

Table of Contents

Requirements

The package requires the following:

- `liboath` from OATH Toolkit. If you can't find it with your distribution's package manager, please consult the OATH Toolkit download page. This has been tested with 1.12.6 and 2.0.2.
- It is recommended that you use this package on a **64-bit architecture**.
- Python 2.6, 2.7, 3.3, 3.4, PyPy 2.0, or PyPy3 2.3.1.
- One of the following:
  - For CPython, a Cython/C extension is available. In order to compile this, the development/header files for `liboath` and a C compiler are required. If installing from Git, Cython 0.18 or higher is also required.
  - The CFFI package (this is included with PyPy/PyPy3, otherwise requires `libffi` development/header files).
- For optional `django-otp` integration, the django-otp library is required. Additionally, the OTP models use a field that only exists in Django 1.6 and above.
- For optional QR code support, the Pillow and qrcode libraries are required. This feature does not work with PyPy3 2.4.0, as `qrcode` requires at least one Python 3.3 feature.
- For optional WTForms integration, the WTForms library is required.
- If you would like to build the documentation, install Sphinx and run `python setup.py build_sphinx`.

Installation

Basic installation from Git:

```
pip install git+https://github.com/malept/pyoath-toolkit.git#egg=pyoath-toolkit
```

Installation from Git with the `qrcode` feature:

```
pip install git+https://github.com/malept/pyoath-toolkit.git#egg=pyoath-
↪toolkit[qrcode]
```

API Documentation

`oath_toolkit`: Core API

Base API for handling one-time passwords.

There are two API types: simple and advanced. The simple API (*HOTP* and *TOTP*) is based on the two-factor authentication API in the Cryptography library. The advanced API (*OATH*) is based on the functional API in OATH Toolkit's liboath.

When compared to the *HOTP*/*TOTP* classes:

- *OATH* has a more customizable set of parameters.
- *OATH* is more likely to add parameters to its method as OATH Toolkit gains APIs.

class oath_toolkit.**HOTP**(*key*, *length*, *algorithm=None*)
HMAC-based one-time password (HOTP) convenience implementation.

API based on `cryptography.hazmat.primitives.twofactor.hotp.HOTP`.

Parameters

- **key** (*bytes*) – The secret key.
- **length** (*int*) – The length of generated one-time passwords.
- **algorithm** – The hash algorithm used during OTP generation. Not currently implemented (requires liboath >= 2.6.0). Defaults to HMAC-SHA1.

generate(*counter*)
Generate an OTP at the specified offset in the OTP stream.

Parameters **counter** (`int` or `long`) – The start counter in the OTP stream.

Return type `bytes`

verify(*hotp*, *counter*, *window=0*)
Verify that the given one-time password is within the range of generated OTPs, given `counter` and `window`.

Parameters

- **hotp** (*bytes*) – The OTP to verify.
- **counter** (`int` or `long`) – The start counter in the OTP stream.
- **window** (*int*) – The number of OTPs after the start counter to test.

Returns The position in the OTP window, where `0` is the first position.

Return type `int`

Raise `OATHError` if invalid

class oath_toolkit.**TOTP**(*key*, *length*, *time_step*, *algorithm=None*)
Time-based one-time password (TOTP) convenience implementation.

API based on `cryptography.hazmat.primitives.twofactor.totp.TOTP`.

Parameters

- **key** (*bytes*) – The secret key.
- **length** (*int*) – The length of generated one-time passwords.
- **time_step** (*int*) – The time step size, which is essentially the lifetime of a given OTP, in seconds. To be clear, this does not mean that the start of the lifetime is the `time` value given to a method of this object. It is recommended to set this value to `30`.
- **algorithm** – The hash algorithm used during OTP generation. Not currently implemented (requires liboath >= 2.6.0). Defaults to HMAC-SHA1.

generate(*time*)

Generate an OTP for the given time value.

Parameters **time** (`int` or `long`) – The UNIX timestamp-encoded time value.

Return type `bytes`

verify(*totp*, *time*, *window=0*)

Verify that the given one-time password is within the range of generated OTPs, given `counter` and `window`.

Parameters

- **totp** (*bytes*) – The OTP to verify.
- **time** (`int` or `long`) – The UNIX timestamp-encoded time value.
- **window** (*int*) – The number of OTPs before and after the start OTP to test.

Returns The position in the OTP window, where `0` is the first position.

Return type `int`

Raise `OATHError` if invalid

class `oath_toolkit.`**OATH**

Bases: `object`

A convenience class that is a direct port of the OATH Toolkit API.

base32_decode(*data*)

Decode Base32 data. Unlike `base64.b32decode`, it handles human-readable Base32 strings.

Parameters **data** (*bytes*) – The data to be decoded.

Return type bytes

base32_encode(*data*, *human_readable=False*)

Base32-encode data.

Parameters

- **data** – The data to be encoded. Must be castable into a `bytes` object.
- **human_readable** (*bool*) – If `True`, transforms the Base32 string into space-separated chunks of 4 characters, removing trailing =.

Return type bytes

check_library_version(*version*)

Determine whether the library version is greater than or equal to the specified version.

Parameters **version** (*bytes*) – The dotted version number to check

Return type bool

hotp_generate(*secret*, *moving_factor*, *digits*, *add_checksum=False*, *truncation_offset=-1*)
Generate a one-time password using the HOTP algorithm (**RFC 4226**).

Parameters

- **secret** (*bytes*) – The secret string used to generate the one-time password.
- **moving_factor** (*int*) – unsigned, can be long, in theory. A counter indicating where in OTP stream to generate an OTP.
- **digits** (*int*) – unsigned, the number of digits of the one-time password.
- **add_checksum** (*bool*) – Whether to add a checksum digit (depending on the version of liboath used, this may be ignored).
- **truncation_offset** (*int*) – A truncation offset to use, if not set to a negative value (which means 2^32 - 1).

Returns one-time password

Return type bytes

hotp_validate(*secret*, *start_moving_factor*, *window*, *otp*)
Validate a one-time password generated using the HOTP algorithm (**RFC 4226**).

Parameters

- **secret** (*bytes*) – The secret used to generate the one-time password.
- **start_moving_factor** (*int*) – Unsigned, can be long, in theory. The start counter in the OTP stream.
- **window** (*int*) – The number of OTPs after the start offset OTP to test.
- **otp** (*bytes*) – The one-time password to validate.

Returns The position in the OTP window, where 0 is the first position.

Return type int

Raise OATHError if invalid

library_version
The version of liboath being used.

Return type bytes

totp_generate(*secret*, *now*, *time_step_size*, *time_offset*, *digits*)
Generate a one-time password using the TOTP algorithm (**RFC 6238**).

Parameters

- **secret** (*bytes*) – The secret string used to generate the one-time password.
- **now** (*int*) – The UNIX timestamp (usually the current one)
- **time_step_size** (int or None) – Unsigned, the time step system parameter. If set to None, defaults to 30.
- **time_offset** (*int*) – The UNIX timestamp of when to start counting time steps (usually should be 0).
- **digits** (*int*) – The number of digits of the one-time password.

Returns one-time password

Return type bytes

totp_validate(*secret*, *now*, *time_step_size*, *start_offset*, *window*, *otp*)
Validate a one-time password generated using the TOTP algorithm (**RFC 6238**).

Parameters

- **secret** (*bytes*) – The secret used to generate the one-time password.
- **now** (*int*) – The UNIX timestamp (usually the current one)
- **time_step_size** (int or None) – Unsigned, the time step system parameter. If set to None, defaults to 30.
- **start_offset** (*int*) – The UNIX timestamp of when to start counting time steps (usually should be 0).
- **window** (*int*) – The number of OTPs before and after the start OTP to test.
- **otp** (*bytes*) – The one-time password to validate.

Returns The absolute and relative positions in the OTP window, where 0 is the first position.

Return type *oath_toolkit.types.OTPPosition*

Raise OATHError if invalid

oath_toolkit.types: Specialized Types

class oath_toolkit.types.**OTPPosition**(*absolute*, *relative*)
Bases: tuple

absolute
Alias for field number 0

relative
Alias for field number 1

oath_toolkit.uri: URI Generator

oath_toolkit.uri.**generate**(*key_type*, *key*, *user*, *issuer*, *counter=None*)
Generate a URI suitable for Google Authenticator. See: https://code.google.com/p/google-authenticator/wiki/KeyUriFormat

Parameters

- **key_type** (*str*) – the auth type, either totp or hotp
- **key** (*str*) – the secret key
- **user** (*str*) – the username
- **issuer** (*str*) – issuer name
- **counter** (int or None) – initial counter value (HOTP only)

Returns a URI

Return type str

oath_toolkit.qrcode: QRCode Generator

oath_toolkit.qrcode.**generate**(*key_type*, *key*, *user*, *issuer*, *counter=None*, ***kwargs*)
Generate a QR code suitable for Google Authenticator.

See: https://code.google.com/p/google-authenticator/wiki/KeyUriFormat

Parameters

- **key_type** (*str*) – the auth type, either totp or hotp
- **key** (*str*) – the secret key
- **user** (*str*) – the username
- **issuer** (*str*) – issuer name
- **counter** (int or None) – initial counter value (HOTP only)
- ****kwargs** – Arguments passed to the qrcode.QRCode constructor

Returns an image object

Return type qrcode.image.base.BaseImage

oath_toolkit.wtforms: WTForms Integration

WTForms-related code for one-time password fields.

class oath_toolkit.wtforms.**HOTPValidator**(*digits*, *window*, *start_moving_factor*, *verbose_errors=False*, *get_secret=None*)
Bases: *oath_toolkit.wtforms.OTPValidator*

Validator for HOTP-based passwords.

Parameters

- **digits** (*int*) – The expected number of digits in the OTP.
- **window** (*int*) – The number of OTPs after the start offset OTP to test.
- **start_moving_factor** (*int*) – Unsigned, can be long, in theory. The start counter in the OTP stream.
- **verbose_errors** (*bool*) – Whether to raise verbose validation errors.
- **get_secret** (*callable*) – If specified, a callable which returns the OATH secret used to validate the OTP.

class oath_toolkit.wtforms.**OTPValidator**(*digits*, *window*, *verbose_errors=False*, *get_secret=None*)
Bases: object

WTForms abstract base field validator for a OTP field.

Parameters

- **digits** (*int*) – The expected number of digits in the OTP.
- **window** (*int*) – The number of OTPs before and after the start OTP to test.
- **verbose_errors** (*bool*) – Whether to raise verbose validation errors.
- **get_secret** (*callable*) – If specified, a callable which returns the OATH secret used to validate the OTP.

get_oath_secret (*form*, *field*)
Retrieve the OATH secret from a given form/field.

Either uses the callback passed in when creating the validator, or the `oath_secret` attribute of the `user` attribute of the `form`.

Return type bytes

otp_validate (*form*, *field*)
This should call the appropriate OTP validation method.

Returns `True` on success

Raises `OATHError` on failure

class `oath_toolkit.wtforms.`**TOTPValidator** (*digits*, *window*, *verbose_errors=False*, *get_secret=None*, *start_time=0*, *time_step_size=30*)
Bases: *oath_toolkit.wtforms.OTPValidator*

Validator for TOTP-based passwords.

Parameters

- **digits** (*int*) – The expected number of digits in the OTP.
- **window** (*int*) – The number of OTPs before and after the start OTP to test.
- **verbose_errors** (*bool*) – Whether to raise verbose validation errors.
- **get_secret** (*callable*) – If specified, a callable which returns the OATH secret used to validate the OTP.
- **start_time** (*int*) – The UNIX timestamp of when to start counting time steps (usually should be `0`).
- **time_step_size** (*int*) – Unsigned, the time step system parameter. If set to a negative value, defaults to `30`.

oath_toolkit.django_otp: django-otp Integration

Contributing

This project is hosted in two places: GitHub and Gitorious. I gladly accept pull/merge requests from both services. Gitorious does not seem to provide an issue tracker, so the GitHub issue tracker is the only one to use at the moment.

Filing Issues

Issues include bugs, feedback, and feature requests. Before you file a new issue, please make sure that your issue has not already been filed by someone else.

When filing a bug, please include the following information:

- Operating system name and version. If on Linux, please also include the distribution name.
- System architecture. For example, `x86`, `x86-64`, `ARM7`.
- The version of OATH Toolkit installed, and the method that it was installed (for example, from source or via package manager).
- Python version, by running `python -V`.

- Installed Python packages, by running `pip freeze`.
- A detailed list of steps to reproduce the bug.
- If the bug is a Python exception, the traceback will be very helpful.

Pull Requests

If you contribute code, please also create tests for your modifications, otherwise your request will not be accepted (I will most likely ask you to add tests). Please make sure your pull requests pass the continuous integration suite, by running `tox` before creating your submission. (Run `pip install tox` if it's not already installed.) The CI suite is automatically run for every pull request on GitHub, but at this time it's faster to run it locally. It would probably also be in your best interests to add yourself to the `AUTHORS.rst` file if you have not done so already.

As of version 1.0.0, this project uses the Semantic Versioning scheme. If your pull request makes incompatible API changes or adds new functionality in a backwards-compatible manner, please emphasize that in your pull request message.

Development Environment

A Vagrant environment is available for developing `pyoath-toolkit`. Run the following command in the top-level source directory (once Vagrant is installed):

```
user@host:pyoath-toolkit$ vagrant up
```

...and it will install all of the Python dependencies in a virtualenv. You can then log into the virtual machine and install the package in develop mode:

```
user@host:pyoath-toolkit$ vagrant ssh
# ...
vagrant@vagrant:~$ source .virtualenv/bin/activate
(.virtualenv)vagrant@vagrant:~$ pip install -e /vagrant
```

Contributors

- Mark Lee

CHAPTER 4

License

Unless otherwise noted in the respective files, the code is licensed under the Apache License 2.0. The otherwise-licensed files have the requisite separate license details. Specifically:

- `oath_toolkit/django_otp/hotp/tests.py` and `oath_toolkit/django_otp/totp/tests.py` are originally licensed under the two-clause BSD license.
- `examples/django/example/forms.py` is originally licensed under the MIT license.

The documentation is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

Python Module Index

o

Index