onetimepass Documentation

Release 0.2.0

Tomasz Jaskowski

June 21, 2013

CONTENTS

CHAPTER ONE

VERSIONS

Current stable release: onetimepass-v0.2.0.tar.gz

Current development release: onetimepass-v0.2.1.tar.gz

1.1 Changelog

Version	**Date**	**Changes**
0.2.1	TBD	• support for unicode secrets, • preliminary support for Travis CI,
0.2.0	2013-04-11	• added compatibility with Python 3.x, • removed compatibility with Python 2.5 and earlier,
0.1.2	2013-01-23	• added automated case fold to secret,
0.1.1	2013-12-20	• internal code improvements, • documentation,
0.1.0	2011-12-19	(initial public release)

CHAPTER

TWO

WHAT IS ONETIMEPASS

OneTimePass (actually `onetimepass`) is a module for generating one-time passwords, namely HOTPs (HMAC-based one-time passowords) and TOTPs (time-based one-time passwords). They are used eg. within Google Authenticator application for Android or iPhone.

CHAPTER

THREE

HOW TO INSTALL

To install the library, you can either use `pip`, or just download it separately. Installing in `pip` is the simplest. Assuming you are installing it system-wide:

```
$ sudo pip install onetimepass
```

(if you are installing it in virtualenv, you do not need “`sudo`” part).

Alternatively, you can follow the download link above and unpack in some directory on your `sys.path`, or clone it as Git submodule to your own directory.

CHAPTER

FOUR

HOW TO USE ONETIMEPASS

You can use this module in the following way:

1. Install module (download it into your application's directory or into modules directory)
2. To get time-based token you invoke it like that:

   ```
   import onetimepass as otp
   my_secret = 'MFRGGZDFMZTWQ2LK'
   my_token = otp.get_totp(my_secret)
   ```

3. To get HMAC-based token you invoke it like that:

   ```
   import onetimepass as otp
   my_secret = 'MFRGGZDFMZTWQ2LK'
   my_token = otp.get_hotp(my_secret, intervals_no=3)
   ```

   where `intervals_no` is the number of the current trial (if checking on the server, you have to check several values, higher than the last successful one, determined for previous successful authentications).

4. To check time-based token you invoke it like that:

   ```
   import onetimepass as otp
   my_secret = 'MFRGGZDFMZTWQ2LK'
   my_token = 123456 # should be probably from some user's input
   is_valid = otp.valid_totp(token=my_token, secret=my_secret)
   ```

5. To check HMAC-based token you invoke it like that:

   ```
   import onetimepass as otp
   my_secret = 'MFRGGZDFMZTWQ2LK'
   my_token = 123456 # should be probably from some user's input
   last_used = 5 # store last valid interval somewhere else
   is_valid = otp.valid_hotp(token=my_token, secret=my_secret, last=last_used)
   ```

   where:

   - `last` argument (in this case being assigned `last_used`) is the number of the last successfully checked interval number (as `valid_totp()` will skip it and start checking from the next interval number)
   - `is_valid` is being assigned value of `False` if `my_token` has not been identified as valid OTP for given secret (`my_secret`) and checked interval range. If it has been successful, `is_valid` is assigned a number of the working interval number (it should be saved into the database and supplied to the function as `last` argument next time the password is being checked, so you cannot use the same token again).

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*