EyeFlask Documentation

Release 0.1.3

Nathan Henrie

Sep 27, 2017

Contents

Contents:

CHAPTER 1

eyeflask package

Subpackages

eyeflask.server package

Submodules

eyeflask.server.crypto module

crypto.py Cryptographic functions for use in eyeflask.server

`eyeflask.server.crypto.`**`create_credential`**`(`*macaddress*, *nonce*, *upload_key*, *from_eyefi=False*`)`

Returns an EyeFi credential.

Generates the credential used by the EyeFi for authentication purposes. The credential is generated slightly differently based on if it is EyeFlask authenticating the EyeFi or the other way around.

`eyeflask.server.crypto.`**`gen_checksums`**`(`*file_handler*`)`

Generates the TCP checksums used to calculate *INTEGRITYDIGEST*.

`eyeflask.server.crypto.`**`make_digest`**`(`*upfile*, *upload_key*`)`

Returns the *INTEGRITYDIGEST*

INTEGRITYDIGEST is used to verify the integrity of the file transfer, calculated using the content of the compressed image and the upload_key

eyeflask.server.views module

“eyeflask.py https://code.google.com/archive/p/sceye-fi/wikis/UploadProtocol.wiki

`eyeflask.server.views.`**`allowed_file`**`(`*filename*`)`

Returns *True* if file extension is *.tar*

`eyeflask.server.views.`**`handle_SOAP`**`()`

`eyeflask.server.views.`**`make_path`**`(`*upload_dir*`)`

`eyeflask.server.views.`**`make_snonce`**`()`

Returns a unique 32 character string

`eyeflask.server.views.`**`page_not_found`**`(`*e*`)`

`eyeflask.server.views.`**`unauthorized`**`(`*e*`)`

`eyeflask.server.views.`**`upload_photo`**`()`

Module contents

Submodules

eyeflask.cli module

`eyeflask.cli.`**`run`**`()`

Run EyeFlask with the built-in Flask server.

Running with the Flask server will be problematic in high traffic circumstances; consider investigating gunicorn or a gunicorn / nginx combo if you encounter difficulties or are running outside of a simple private home network environment.

Module contents

`eyeflask.`**`create_app`**`(`*config=None*`)`

CHAPTER 2

Credits

Development Lead

- Nathan Henrie nate@n8henrie.com

Contributors

- None yet. Why not be the first?

CHAPTER 3

Changelog

0.1.3 :: 20170310

- Add strftime based subfolders to config (see issue #2)
- Encourage use of virtualenv in the README
- Test docs creation in Travis
- Fix some flake8 compliants

0.1.2 :: 20170303

- Python 3.6 compatibility
- Stop using `src/` subdirectory
- Try to fix some Pandoc exceptions

0.1.1 :: 20160318

- Use `array.array` instead of `struct.iter_unpack` for modest speed boost
- Rename `start_session` to `handle_SOAP` – because that's what it does
- Extract the image from the tarfile data prior to writing to disk (eliminating the need to delete the tarfile afterwards)

0.1.0 :: 20160227

- Initial release to GitHub, PyPI

CHAPTER 4

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/n8henrie/eyeflask/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/n8henrie/eyeflask/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 :)

Get Started!

Ready to contribute? Here's how to set up EyeFlask for local development.

1. Fork the `eyeflask` repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/eyeflask.git
   ```

3. Install your local copy into a virtualenv. Assuming you have python >= 3.4 installed, this is how you set up your fork for local development:

   ```
   $ cd eyeflask
   $ python3 -m venv venv
   $ source venv/bin/activate
   $ pip install -e .
   ```

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 eyeflask tests
   $ python3 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.

Pull Request Guidelines

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

1. The pull request should include tests if I am using tests in the repo.
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.md
3. The pull request should work for Python 3.4 and 3.5. If I have included a `.travis.yml` file in the repo, check https://travis-ci.org/n8henrie/fauxmo/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests: `py.test tests/test_your_test.py`

CHAPTER 5

EyeFlask

Simple Flask-based Python3 EyeFi server

- Documentation: eyeflask.readthedocs.org

Introduction

I use an Eye-Fi SD card in my portable scanner. Unfortunately, it used to upload directly to Evernote, but no longer supports uploading directly to any service that suits my needs. Additionally, they don't provide a Linux version of their server software.

EyeFlask is a simple Flask-based Eye-Fi server written for Python >= 3.4. The Eye-Fi card can connect to it and will upload images to the folder specified in the config. EyeFlask attempts to verify the file integrity using the same security protocols used by Eye-Fi Server.

Dependencies

- Python >= 3.4
- See `requirements.txt`

Quickstart

Using a virtual env is highly recommended. Since EyeFlask is Python 3.4+ only, there's no excuse not to use one!

1. `python3 -m venv venv` on some systems may need to install venv for Python3.4 (e.g. `sudo apt-get install python3.4-venv`)
2. `venv/bin/pip3 install eyeflask`

3. Copy `eyeflask/extras/eyeflask-sample.cfg` to `eyeflask.cfg`, modify with your values, and put it in your instance folder
4. Run: `venv/bin/eyeflask`
5. Scan some stuff, see if it ends up in your uploads folder

Development Setup

1. Clone the repo: `git clone https://github.com/n8henrie/eyeflask && cd eyeflask`
2. Make a virtualenv: `python3 -m venv venv`
3. Make an instance folder: `mkdir -p instance`
4. Copy the config sample: `cp eyeflask/extras/eyeflask-sample.cfg instance/eyeflask.cfg`
5. Edit the config to include your upload directory and upload_key (see below): `vim instance/eyeflask.cfg`
6. Install with dev dependencies: `venv/bin/pip install .[dev]`
7. Run: `eyeflask` (or `venv/bin/python -m eyeflask.cli`)
8. Scan some stuff, see if it ends up in your uploads folder

Configuration

- `UPLOAD_KEY`: See FAQ, below
- `UPLOAD_FOLDER`: Destination folder for uploads. May optionally be a strftime string to specify a subfolder format (e.g. '/path/to/eyeflask-uploads/%Y/%m-%d/')

Extras

EyeFlask will help get the images uploaded and extracted to your server (e.g. a Raspberry Pi in my case), but what do to from there? If you're running Raspbian Jessie (and using systemd), I've included in the `extras` folder a few files that may be of interested.

- `upload_scans.service` will run a given script when called (e.g. `sudo systemctl start upload_scans.service`)
- `upload_scans.timer` is an example systemd timer unit that will call `upload_scans.service` every 10 minutes
- `upload_scans.path` is an example systemd path unit that will call `upload_scans.service` every time a file changes in a watched directory

Put together, this makes it *really* easy to put together a script to upload new scans to a Dropbox folder whenever a new one is added, or whatever command you'd like to run on all your scans.

I've also included `eyeflask/extras/eyeflask.service`, which is a sample systemd service file to run EyeFlask at startup and restart it on errors.

Acknowledgements

Much of the code for EyeFlask came from or was inspired by the following projects / links. Many thanks to the authors for their work! If I've forgotten anyone, let me know.

- https://github.com/tachang/EyeFiServer
- https://github.com/dgrant/eyefiserver2
- https://code.google.com/archive/p/sceye-fi/wikis/UploadProtocol.wiki
- https://launchpad.net/eyefi
- https://code.google.com/archive/p/eyefiserver/
- https://github.com/BrentSouza/WP7EyeFiConnector

Troubleshooting / FAQ

Where do I find my upload key?

You'll need a supported platform (OS X or Windows) with `Eye-Fi Center.app` installed, and need to have uploaded photos to that computer at least once. This ensures everything is working, and generates the `Settings.xml` file, from which you need to copy the upload key into `eyeflask.cfg`.

- OS X: `~/Library/Eye-Fi/Settings.xml`
- Windows 7: `C:\Users\[user]\AppData\Roaming\Eye-Fi\Settings.xml` (source)
- Windows XP: `C:\Documents and Settings\[user]\Application Data\Eye-Fi\Settings.xml` (source)

Is it okay to be running this with the built-in Flask server?

It's not perfect, but it seems to work okay for me and my single Eye-Fi card setup. You'd probably be better off running it behind gunicorn or a gunicorn / nginx setup, but I'm running it behind Flask alone for simplicity and because I haven't had any issues so far.

If you want to give it a go with gunicorn / nginx, I've included an *extremely* simplified nginx configuration file: `eyeflask/extras/nginx.conf`. After installing `gunicorn` into your virtualenv, hopefully you'll be able to get it running behind nginx without much trouble with something like:

```
venv/bin/pip install gunicorn
venv/bin/gunicorn 'eyeflask:create_app("instance/eyeflask.cfg")'
```

For debugging you can also use the flags `--log-file=- --log-level=debug`.

NB: I do **not** plan on providing support for nginx / gunicorn setups, as I don't know enough about it to be particularly helpful. I just verified that it seemed to work. (Just FYI, Gunicorn *without* nginx did **not** seem to work unless I used one of the async workers, kept getting timeouts.)

Why am I getting repeat or unreliable file uploads on my Raspberry Pi?

I'm not sure. I was getting *excellent* reliability when running EyeFlask on my Macbook Air and *very* poor reliability on my Raspberry Pi B+ with EyeFlask 0.1.0. It seemed like the file would be uploaded exctracted without issue, but the EyeFi card kept sending the same file over and over, leading me to believe that the confirmation response wasn't getting received every time. I thought it might have something to do with slow response times, so I did a little code optimization with 0.1.1 which seems to have helped. I also gave up and put EyeFlask behind a gunicorn / nginx setup, and between the two of these changes I have much better upload reliability.

CHAPTER 6

Indices and tables

- genindex
- modindex

Python Module Index

Index