Fearofcode's Internet Happenings

Doing a code on the computer, popping a wheelie at the mall and doing a flip

How to Scrub Sensitive Information From Django settings.py Files

Jan 15th, 2013

This post tries to describe how to take a private Django project so that you release it open source on Github without giving away your secret key, your database passwords, or other sensitive information. Different people do their Django configuration different ways for some reason, so if you want to do it differently, that's fine.

The situation I had was as follows: I made a new Django project and hosted it as a private repo on Github. I wanted to share the code, but my `settings.py` file had the secret key Django autogenerates when you make the project, along with other sensitive information: public and private keys issued by Twitter as the app used Twitter OAuth for authentication, and the production database password.

There are at least two issues here:

1. How you want to handle database passwords in your Django application when the repo will be public
2. How to remove Git commit history where sensitive information was committed to `settings.py`

Making a place for secrets to live in

The thing about Django's default settings file is that it mixes non-sensitive configuration information with sensitive passwords and keys. Why? Why? Whatever. The approach I took was the following:

- Segregate the sensitive information in `settings.py` into a new file, `settings_secret.py`.
- Import the secret information from `settings.py`:

```
1 from settings_secret import *
```

- Add `settings_secret.py` to `.gitignore`:

```
1 echo "<app_name>/settings_secret.py" >> .gitignore
2 git add .gitignore
3 git commit -m "Ignore secret settings"
```

- Copy `settings_secret.py` into a new file, `settings_secret.py.template` with values filled to dummy ones so that people who use your code can fill them in as needed.

Depending on your deployment setup, you can then take the next logical steps to fill in `settings_secret.py` on your production setup.

There are other ways of doing this. They are fine. If it works, fine. But at this point we have a `settings.py` free of secret information.

Now we need to clear our Git commit history of sensitive information.

Rewriting Git history

1. Make a backup copy of your newly scrubbed `settings.py` outside the repo. You might want to make a backup copy of the entire repo if you are bad at Git the way I am.
2. Follow Github's steps for removing sensitive data with your `settings.py` file as the file of interest. This will remove every mention of `settings.py` from your commit history.
3. Copy your clean `settings.py` file into your repo.

At this point you should have a repo that can be released as open source without revealing any sensitive information.

Again, you might have a better way of doing this, but this wasn't obvious to me, so I figured I'd jot down what I did as most answers on StackOverflow ignore the issue of how to deal with sensitive information in Git repo histories.

OK Bye

Patrick Thomson and Github helped me figure this out. Many thanks to both of them because holy cow is this stuff confusing.

Love,

Warren Henning (@fearofcode)

Posted by Warren Henning Jan 15th, 2013

Recent Posts

- How to scrub sensitive information from Django settings.py files