

“ Da Vinci Code World ”

A Information Encryption Blog System based on Python

Eve Thullen, Claremont Graduate University & Harvey Mudd College

1. Project Background:

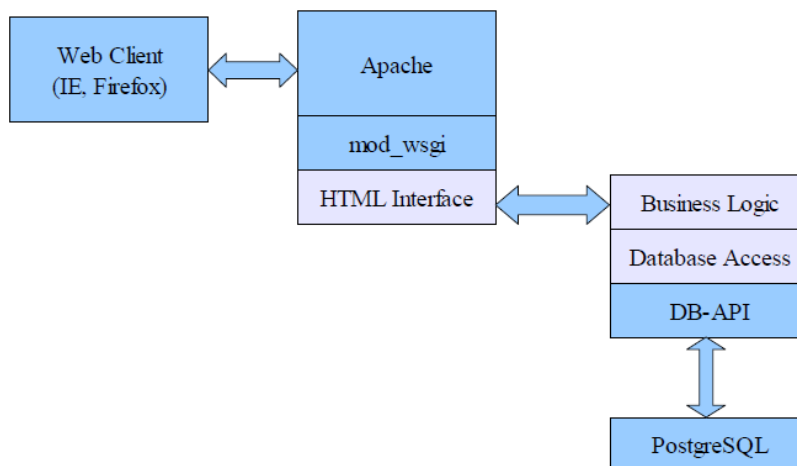
Let's call this project Da Vinci Code World. It's a website that the user could post text messages, that the user wants to hide real messages behind those text for different reasons, such as avoiding web censorship and monitoring systems, or the user wants to deliver an important message to specific people. In a world, it's a website for specific people to share secrets.

2. Project Description:

1. A website written by Python and connected to a database, that the user can register an account and login.
2. User can post text or articles. Those text are encrypted, and at the same time an encryption key is created for each post. The algorithm is written by Python.
3. Readers can add comments to the article.
4. Readers can require a key to decrypt the message.

3. Project Platform and Structure:

1. Python code to connect Apache and database PostgreSQL together
2. Python and Pesto to generate HTML to display to the user



- Project Language:

Python
HTML
SQL

- Platform and Frame:

Database: PostgreSQL 9.6.5
Server: Apache 2
Python: 2.7

- Python Library to Use:

mod_wsgi (Python based Web Application Run on Apache)
Pesto (HTML interface)
pgdb (database interface)

4. Project Installation:

- Platform and Frame Installation:

1. **PostgreSQL 9.6.5** installation: <https://www.postgresql.org/download/macosx/>

Set up pg_config path: /users/yingfenhuang/PostgreSQL/pg96/bin

```
$ export PG_HOME=/users/yingfenhuang/PostgreSQL/pg96
```

```
$ export PATH=$PATH:$PG_HOME/bin
```

,

2. **Apache 2**: MacOS built in Apache 2

3. **Python 2.7.14**, <https://www.python.org/downloads/>

- Python Library Installation:

1. **mod_wsgi 4.5.20**

The mod_wsgi package provides an Apache module that implements a WSGI compliant interface for hosting Python based web applications on top of the Apache web server.

- 1) Install mod_wsgi to python: [pip install mod_wsgi](#)
or download and install to Python: https://pypi.python.org/pypi/mod_wsgi
[python setup.py install](#)

2) Run the test.py: rom terminal cd into the test.py folder

```
mod_wsgi-express start-server
```

or

```
mod_wsgi-express start-server test.py
```

or

```
mod_wsgi-express start-server test.py --port 8080
```

3) Open the web:

<http://localhost:8000/>

2. pesto/ Python 2.7

1) Install pesto: compile with Python 2.7

```
$ pip install pesto
```

or download and install to Python:

```
python setup.py install
```

<https://pypi.python.org/pypi/pesto/25>

2) Run pesto

```
$ python test_pesto.py
```

3) Open the web:

<http://localhost:8000/>

* test_pesto.py

```
from pesto import to_wsgi, Response
```

```
from wsgiref import simple_server
```

```
def handler(request):
```

```
    return Response([
```

```
        "<html>",
```

```
        "<body><h1>Hello World!</h1></body>",
```

```
        "</html>",
```

```
    ])
```

```
if __name__ == "__main__":
```

```
    httpd = simple_server.make_server("", 8000, to_wsgi(handler))
```

3. Festo

1) Install fresco

```
$ pip install fresco gunicorn
```

2) Run fresco

```
$ gunicorn Test_fresco:app
```

3) Open the web:

<http://localhost:8000/>

```
* test_fresco.py
```

```
from fresco import FrescoApp, GET, Response
```

```
def helloworld():
```

```
    return Response(["<h1>Hello World!</h1>"])
```

```
app = FrescoApp()
```

```
app.route('/', GET, helloworld)
```

4. pgdb

1) Install PyGreSQL, need install PostgreSQL first, then

set up pg_config path: /users/yingfenhuang/PostgreSQL/pg96/bin

```
$ export PG_HOME=/users/yingfenhuang/PostgreSQL/pg96
```

```
$ export PATH=$PATH:$PG_HOME/bin
```

Install PyGreSQL

```
$ pip install PyGreSQL
```

```
* test_pgdb.py
```

```
import os
```

```
os.environ['PATH'] += ";/users/yingfenhuang/PostgreSQL/pg96/bin"
```

```
import pgdb
```

```
#Connect to database
```

```
try:
```

```
    connection = pgdb.connect(user="postgres", password="testpgdb",  
database="postgres")
```

```
    cursor = connection.cursor()
```

```
except:
```

```
print "\n failed to connect to Database"  
print "\n Exception:", sys.exc_value
```

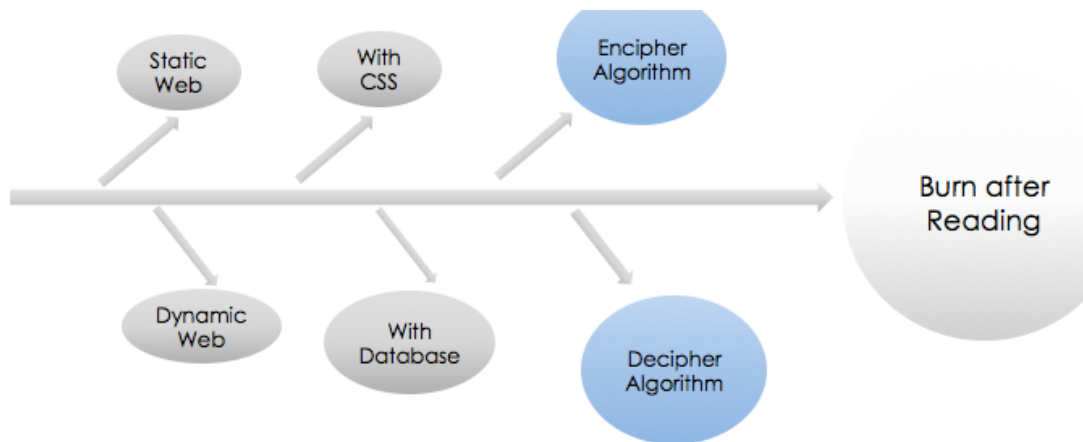
Eve Thullen

IST 431 Python Programming

Dec 16, 2017

5. Project Process

- Initial vision and its evolution



- The progress and the final system's capabilities

Version 0: Built in Static Web Page with html in local host

```
import test  
  
if __name__ == "__main__":  
    from wsgiref import simple_server  
    httpd = simple_server.make_server('', 8000, test.application)  
    httpd.serve_forever()
```

Version 1: Built in Dynamic Web Page with html

Blog Entries

New blog post

[Cancel](#)

You are logged in as test1

Time 2017-12-02 11:27:19.830524

IP address 127.0.0.1

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5)

Browser AppleWebKit/603.3.8 (KHTML, like Gecko) Version/10.1.2 Safari/603.3.8

Version 2: Built in Dynamic Web Page with Database

The screenshot shows a web browser window with the URL localhost. The page has a purple header with the text "Welcome To MyBlog". On the left, there is a sidebar with a "HOME" link and a list of links: "this is test4", "this is test3", and "test1". The main content area is titled "Blog Entries" and contains two entries. The first entry is "this is test4" by Author: test4, dated 2017-11-11, with a "View Details" button. The second entry is "this is test3" by Author: test3, dated 2017-11-11, also with a "View Details" button. On the right side, there is a login form with fields for "Username:" and "Password:", and buttons for "Login" and "New User".

Version 3: Built in Dynamic Web Page with CSS

The screenshot shows a web browser window with the URL localhost. The page has a dark, space-themed background with a rocket and stars. The header features the text "DaVinci's Code World" in a white cursive font, followed by "Burn after Reading" in a white sans-serif font. Below the header, there is a quote: "YOU SHALL KNOW THE TRUTH, THE TRUTH WILL SET YOU FREE!!" and "Your Adventure Starts Here...". The main content area contains a blog entry titled "this is test4" by Author: test4, dated 2017-11-11. On the right side, there is a login form with fields for "Username:" and "Password:", and buttons for "Login" and "New User". On the left, there is a sidebar with a "HOME" link and a list of links: "this is test4", "this is test3", and "test1".

Version 4: Built in Cryptographic Dynamic Web Page

The screenshot shows a web page with a dark background. On the left, there is a navigation menu with links: HOME, My Page, Brutal Kangaroo, Vault 7 main publication., Macron Campaign, Emails, Macron Campaign Emails Encrypted, this is test4, and this is test3. The main content area features a post titled 'Brutal Kangaroo' by 'test1' from '2017-12-09'. The post text is encrypted and includes a 'Decipher' button. Below the post is a text input field with the placeholder 'put your code here' and another 'Decipher' button. A paragraph of text follows, mentioning WikiLeaks and the Brutal Kangaroo project. At the bottom of the post area is an 'Add Comments' link. On the right side, there is a sidebar with the text 'You are logged in as test1' and a list of user details: Time (2017-12-16 00:35:25.709), IP address (127.0.0.1), Browser (Mozilla/5.0 (Macintosh; Intel OS X 10_10_5 AppleWebKit/603.3.8)), and a 'Logout' button. There is also an 'add new blog' button.

System capabilities:

- User Register and Error Validation
- Front Page Posts and Page navigation
- Add New and Delete Post
- Add Comments to the Post
- Encipher Post and Decipher Post

Tackle the project differently:

One of the most important part of this the project is about encipher and decipher part. I probably use python Cryptographic library to encrypt the text. However, I designed my own encipher algorithm to encrypt the text for this project, just for security reason.

Future Version:

Version 5: Burn after Reading

- The user could set up the time for the post, that their post will disappear itself at specific time.
- Other language version, such as Chinese Version