

COMP.SEC.300 Project work

Introduction

This work was done for [Secure Programming](#) course at Tampere University. The result of the project was a password manager storing encrypted passwords into a mongo database.

The work is not based on any previous works.

Tech

The project was implemented with Python and a web framework called Flask. Mongo database is used to store user information.

Structure

- Environment files, documentation, and actual running file main.py reside in the root folder.
- Blueprints folder has Flask blueprints.
- Each blueprint handles routes and functionalities relating to the filename.
- Logs folder stores logs.
- Static folder is a default Flask folder. This folder stores style.css stylesheet.
- Template folder is also a default Flask folder. This folder stores HTML files.
- Utils folder stores different python files providing utilities for the application.

```
root
|   - .env
|   - .gitignore
|   - docker-compose.yml
|   - environment.yml
|   - main.py
|   - README.md
|   - requirements.txt
|---blueprints
|   - login.py
|   - signin.py
|   - vaults.py
|---logs
|   - * Generated log files *
|---static
|   - style.css
|---templates
|   - create_vault.html
|   - index.html
|   - login.html
|   - show_vault.html
```

```

|   - signin.html
|   - vaults.html
|___utils
|   - crypto.py
|   - mongo.py
|   - psw_validation.py

```

Secure programming solutions

OWASP Top 10 list provided a checklist for programming solutions. Here are points that were taken into consideration:

1. Security Logging and Monitoring Failures

- System logging is implemented. Every POST method functionality is thoroughly logged.
 - All database queries are also logged.
- Voluntary encryption is implemented for the log files.
- Code:
 - Logger is initialized in main.py. If an encryption key is provided in the .env file the log files are encrypted.
 - Log filename is **logfile_dd_mm_yyyy.log** or **logfile_dd_mm_yyyy_encrypted.log**
 - Logger initialization:

```

65     # Check if the environment variable exists
66     if log_encryption_key is None:
67         logger.warning("LOG_ENCRYPTION_KEY is not set in the .env file. Logs will not be encrypted!")
68
69     # Logger
70     date_time = datetime.strftime(datetime.now(), "%d_%m_%Y")
71     if log_encryption_key is not None:
72         try:
73             crypto.set_logging_cipher(log_encryption_key)
74             logger.add(f"logs/logfile_{date_time}_encrypted.log",
75                       format=crypto.encrypted_formatter)
76         except ValueError as e:
77             logger.error(e)
78             logger.info("Existing!")
79             sys.exit()
80     else:
81         logger.add(f"logs/logfile_{date_time}.log")

```

- Logger formatter for encryption:

```

86     def encrypted_formatter(self, record):
87         encrypted = self.logging_cipher.encrypt(record["message"].encode("utf8"))
88         record["extra"]["encrypted"] = base64.b64encode(encrypted).decode()
89         return "{extra[encrypted]}\n{exception}"
90
91     def set_logging_cipher(self, key: str):
92         key_bytes = base64.b64encode(key.encode())
93         self.logging_cipher = Fernet(key_bytes)

```

- Logger is used with for example **logger.info("This text is logged!")**

2. Cryptographic failures

- HTTPS is used with self-signed certificates.
 - Self-signed certificate and HTTPS connection is initialized in main.py.
 - `ssl_context="adhoc"` creates certificate and enables HTTPS connection

```
104     app.run(host='localhost', port=5001, debug=True, ssl_context='adhoc')
```

- No deprecated cryptographic algorithms.
 - Used algorithms:
 - PBKDF2

```
42  def derive_key(self, psw: str, salt: bytes):
43      """
44      Use PBKDF2 to produce a key using the given password and salt
45      """
46      kdf = PBKDF2HMAC(
47          algorithm=hashes.SHA256(),
48          length=32,
49          salt=salt,
50          iterations=480000,
51      )
52      return kdf.derive(psw.encode())
53
54  def verify(self, psw: str, key: bytes, salt: bytes):
55      kdf = PBKDF2HMAC(
56          algorithm=hashes.SHA256(),
57          length=32,
58          salt=salt,
59          iterations=480000,
60      )
61
62      try:
63          kdf.verify(psw.encode(), key)
64          return True
65      except Exception:
66          return False
```

- AES256 in CBC mode

```

13  def __init__(self) -> None:
14      self.logging_cipher : Fernet = None
15
16  def encrypt(self, key: bytes, data: str) -> bytes:
17      iv = os.urandom(16)
18      # Add padding
19      padder = padding.PKCS7(algorithms.AES.block_size).padder()
20      padded_data = padder.update(data.encode()) + padder.finalize()
21      # Encrypt
22      cipher = Cipher(algorithms.AES(key), modes.CBC(iv))
23      encryptor = cipher.encryptor()
24      ct = encryptor.update(padded_data) + encryptor.finalize()
25      return base64.b64encode(iv + ct)
26
27  def decrypt(self, key: bytes, ct: bytes) -> bytes:
28      enc = base64.b64decode(ct)
29      # Separate IV and data
30      iv = enc[:16]
31      encdata = enc[16:]
32      # Decrypt
33      cipher = Cipher(algorithms.AES(key), modes.CBC(iv))
34      decryptor = cipher.decryptor()
35      pt = decryptor.update(encdata) + decryptor.finalize()
36      # Unpad
37      unpadder = padding.PKCS7(algorithms.AES.block_size).unpadder()
38      output = unpadder.update(pt) + unpadder.finalize()
39
40      return output

```

- SHA256 is used for hashing.

```

68  def hash_str(self, to_hash: str):
69      """
70      Hash string with SHA256
71      """
72      digest = hashes.Hash(hashes.SHA256())
73      digest.update(to_hash.encode())
74      digested = digest.finalize()
75      return digested
76
77  def hash_bytes(self, to_hash: bytes):
78      """
79      Hash bytes with SHA256
80      """
81      digest = hashes.Hash(hashes.SHA256())
82      digest.update(to_hash)
83      digested = digest.finalize()
84      return digested

```

- Randomness is cryptographically safe.
 - `os.urandom()` function is used to create randomness.
- No keys in the source code.
- All data is stored encrypted and keys are stored as byte arrays.

```
_id: ObjectId('645f74c265315bd490ebc952')
username: BinData(0, 'fyaJpUrVBoAmvvziPdefWg0510J1YTaRpsmKhS4DwdA=')
psw: BinData(0, 'fyaJpUrVBoAmvvziPdefWg0510J1YTaRpsmKhS4DwdA=')
```

```
_id: ObjectId('645f987f72890cfd89df80d6')
username: BinData(0, 'UxVu054fqgDCqor6clpwY2nDfpPj2UqoABmtScuXqAY=')
psw: BinData(0, 'UxVu054fqgDCqor6clpwY2nDfpPj2UqoABmtScuXqAY=')
```

```
_id: ObjectId('645f98ccbce56c6d84873910')
username: BinData(0, 'j8NSTCTsAzILPGmkU7+RPmiCfW1SqVyId2eqiLWy9qE=')
psw: BinData(0, 'j8NSTCTsAzILPGmkU7+RPmiCfW1SqVyId2eqiLWy9qE=')
```

```
_id: ObjectId('645f98ea0c36d6e769f585d0')
username: BinData(0, 'JBHgS0sdkF3hFGkzdB3jtriEhSHG2Euf/CmNEHrytLE=')
psw: BinData(0, 'JBHgS0sdkF3hFGkzdB3jtriEhSHG2Euf/CmNEHrytLE=')
```

3. Vulnerable and Outdated Components

- System was developed with Python 3.11.
- All used cryptography libraries are considered safe libraries.
 - Used cryptography library uses OpenSSL for cryptographic operations.
- Tried to find security concerns regarding used components and didn't find any.

4. Injections

- All data used in mongo queries is hashes as byte arrays thus making this vulnerability obsolete.

Other security related things

1. CSRF prevention

- This is implemented using Flask CSRF library. All POST methods require a valid CSRF token.
 - CSRF protection is initialized at **csrf = CSRFProtect(app)** in the picture below.
 - CSRF token is generated in HTML forms with **{{ csrf_token() }}**.
 - CSRF library automatically handles CSRF tokens.

2. Sessions

- Stored and encrypted server-side. Client only has session key.
 - Key for encryption sessions data is generated in the picture below at:
 - **app.config["SECRET_KEY"] = os.urandom(16)**
- Max length is set to 30 mins.
 - **app.config["PERMANENT_SESSIONS_LIFETIME"]** in the picture below.
- Session is deleted when browser is closed.
 - **app.config["SESSIONS_PERMANENT"]** in the picture below.

```

15 def create_app() -> Flask:
16     app = Flask(__name__)
17     # Used for encrypting sessions
18     app.config["TESTING"] = False
19     app.config["SECRET_KEY"] = os.urandom(16)
20     app.config["SESSION_TYPE"] = "filesystem"
21     # Logs out after 10 minutes
22     app.config["PERMANENT_SESSION_LIFETIME"] = 600
23     # The session will be deleted when the user closes the browser.
24     app.config["SESSION_PERMANENT"] = False
25     return app
26
27 # Create app and register blueprints
28 sess = Session()
29 app = create_app()
30 app.register_blueprint(login)
31 app.register_blueprint(signin)
32 app.register_blueprint(vaults)
33 csrf = CSRFProtect(app)

```

3. Brute-force attack prevention

- ReCaptcha v3.0 is used to prevent brute-force attacks and bots.
 - ReCaptcha is enabled in the HTML files using buttons shown below.

```

22 <input class="btn green g-recaptcha"
23     data-sitekey="{ site_key }"
24     data-callback="onSubmit"
25     data-action="submit"
26     value="Log In"
27     type="submit">

```

- ReCaptcha is validated in login and sign in form with the code below.

```

34 secret_response = request.form["g-recaptcha-response"]
35 verify_response = requests.post(
36     url=f"{RECAPTCHA_VERIFY_URL}?secret={RECAPTCHA_SECRET_KEY}&response={secret_response}").json()
37 # Check success and score
38 # 0.5 threshold is default recommended in reCaptcha docs
39 if verify_response["success"] == False:
40     logger.error("Unable to verify recaptcha!")
41     abort(401)
42 elif verify_response["score"] < 0.5:
43     logger.error("ReCaptcha score is under 0.5!")
44     abort(401)

```

4. User is able to define mongo credentials

- This depend on the user but it is possible to set mongo credentials.
- Setting mongo creadentials is explained below at how to setup enviroment section.

Testing

Manual testing

During the development extensive manual testing was done. Manually tested things were:

1. CSRF library
2. User sign in, login (in and out).
 - Invalid passwords and usernames were tested multiple times.
3. Vault creation and deletion.
 - Invalid vault names (duplicate, empty or just space) tested.
4. Item creation inside vaults.
 - Invalid names (duplicate, empty or just space) and usernames were tested.

Automated testing

Automated test have been implemented for following functionalities:

- User log in and sing in
- CRUD for vaults and passwords inside vaults
- Password validation
- Cryptographic functionalities

These tests cover everything the application does. Test files are name accordingly.

NOTE: CSRF tokens are disabled in testing mode

How to setup environment

Python environment

The are a couple of ways to setup python environment:

1. Use environment.yml for creating a conda environment.
 - Commands
 - **conda env create --file=environment.yml**
 - **conda activate sec_prog2**
2. Use requirements.txt to install python dependencies.
 - **NOTE:** Python 3.11 was used in developement. Please check that given requirements are compatible with your python version.
 - Commands:
 - **python -m pip install -r requirements**

Mongo

The are a couple of ways to setup mongo.

1. Use docker-compose.yml file to create a docker container.
 - .env file's mongo related variables are compatible with this .yaml file's mongo settings.
 - command:
 - **docker-compose up --build**
2. Use an installed mongo on your computer.
 - **NOTE:** .env file stores mongo username, password, address and port.

.env file

NOTE: If you are using docker-compose and environment.yml you don't have to use this The .env file contains different parameters for running the application:

- LOG_ENCRYPTION_KEY (32 byte key) : Voluntary
 - This variable contains an encryption key for the logger. If this is used the logger encrypts the output.
 - By removing this you can remove logger encryption.
- RECAPTCHA_SECRET_KEY, RECAPTCHA_SITE_KEY, RECAPTCHA_VERIFY_URL : **MANDATORY**
 - Keys and URL for reCaptcha.
- MONGO_URL and MONGO_PORT : **MANDATORY**
- MONGO_USERNAME and MONGO_PSW : Voluntary

Using and testing the application

After setting up the environment run **python main.py** and the application is now running at <https://localhost:5001>.

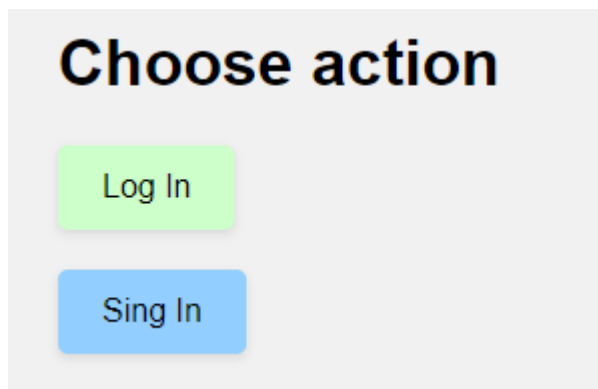
Run tests

Test can be run by running **coverage run -m pytest -s**

UIs

The UI is very straight forward.

- On the index page of localhost:5001 the user can select wether to log in or sign in.



- On the sing in page the user can input username and password. If the sing in is successful the user is redirected to index page and otherwise they are notified with a red box.

Sing In

Username :

Password :

Sing In

Cancel

Password criteria:

- Minimum length of 8 characters.
- Must include both uppercase and lowercase letters.
- Must include at least one digit.
- Must include at least one special character.

No uppercase or lowercase characters!

- On the log in page the user can login. The user is redirected to vaults page if the login is successful. Otherwise they are notified with a red box.

Log In

Username :

Password :

Log In

Cancel

Password is incorrect!

- On the vaults page the user can see their existing vault and create new ones.

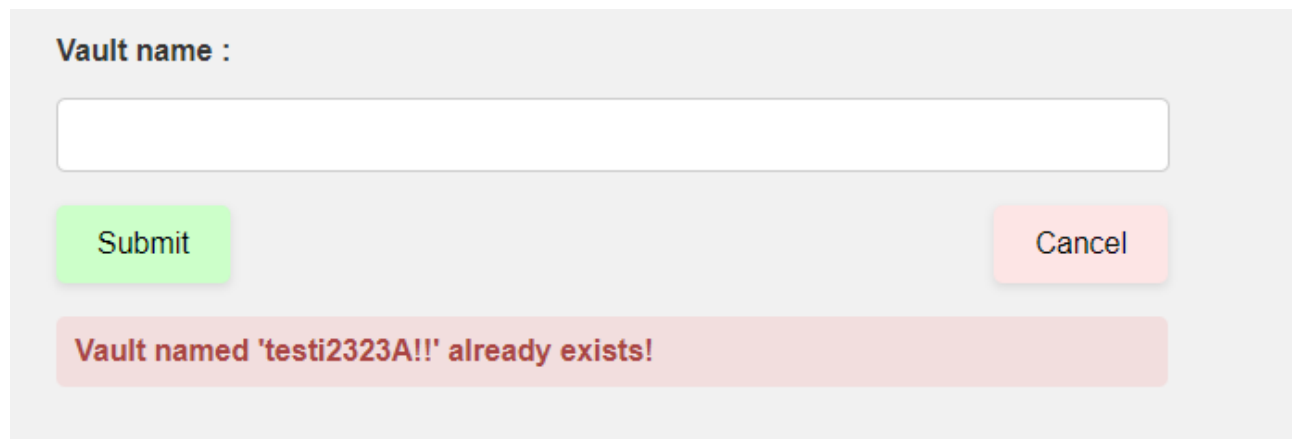


Vaults:

- TestVault1
- TestVault2
- testi2323A!!

Create Vault Log Out

- On the vault creation page the user can create new vaults. If the vault is created successfully the user is redirected to vaults page. Otherwise they are notified with a red box.







Vault name :

Submit Cancel

Vault named 'testi2323A!!' already exists!

- By clicking a vault the user can create new items inside a vault as well as modify and delete them. The user can also delete the whole vault from this page. When performing actions the user is notified of the action's result accordingly.

Current vault: TestVault

Item name :	Item name :
<input type="text" value="TestItem"/>	<input type="password" value="TestItem!1"/>  
Item name :	Item name :
<input type="text" value="TestItem2"/>	<input type="password" value="....."/>  

Save changes

Add Item

Remove vault

Cancel

Password criteria:

- Minimum length of 8 characters.
- Must include both uppercase and lowercase letters.
- Must include at least one digit.
- Must include at least one special character.