group2

Dutch Police Internet Forensics Web App System with DJANGO Framework

1) To run the application:

```
#navigate to correct directory
    cd Group2
#create a virtual environment
    pip install pipenv
#install django
    pipenv install django
#start the server
    python manage.py runserver
#click on the URL, example below
    Starting server at http://127.0.0.1:8000/
```

2) Application Description

The web application was created using the Django web framework:

2a) Login Page
This requires a valid user id and valid password

2b) Dashboard Page
The dashboard displays statistics - cases(total, pending, reviewed, completed) and criminal activities
There is a menu to the left of the dashboard and menu items are restricted by the user's role and permissions
Only the administrator has access to the administration function

2c) Administration Page
Provided with Django, the UI has been extended to include a user profile section

Enables roles and permissions to be created and assigned to users, to restrict access to functions/CRUD actions

2d) Password Change Page

Enables the user to change their password (greater than 7 characters, not commonly used, has to contain a digit, letter, lowercase character, uppercase character, special character)

3) Differences between the Design and the Application

Below are the 5 threat categories, and security threat mitigations which were proposed as part of the first sprint, all of these were included in the implementation

3a) Authorization

User permissions and roles
Menu functions and CRUD operations restricted by user permissions
The administrator can de-activate a user and add a new user
Logout function
The user has to have the correct permissions to access the administrator page
The user is prevented from navigating back to the login page once the user has logged in

3b) Authentication

Strict password standards

3c) Input Validation

The Login UI has input validation
The administrator function consists of drop down menu's for fields where relevant

3d) Encryption

The password is encrypted using a PBKDF2 algorithm and a SHA256 hash

3e)Security Misconfiguration

Django uses MVT (model-view-template) which is the same principal as MVC

The following functionality was included, which was not part of the original sprint one scope in the design:

- A user dashboard to test roles and permissions
- CSRF token
- Logout function
- Controls in terms of navigation, so a user cannot navigate to the admin page via the URL, or navigate back to the login page

Two items were taken out of scope for sprint one and will be included in sprint two:

- OTP
- Three password attempts to login/account locked

4) Was the appropriate design approach used?

The application coding did follow the UML use case and class diagram design

Django as a framework was proposed, this was the right approach because:

4.1) Django supports O-O:

- MVT supports O-O design by encapsulating the logic in the UI (template), the controller (view) and the database (model)
- Django uses ORM, which provides abstraction between the database and the model
- Django supports class based views and so code is re-useable, classes and functions were used in the application
- An example of inheritance in the application code is where the navbar is inherited by other templates

4.2) Django has built in security features:

- Django's ORM uses parametized statements, reducing the risk of SQL injection
- Django uses templates to prevent Cross Site Scripting attacks (XSS)
- Django uses a CSRF token to reduce the risk of cross site forgery
- Django stores session information in the database and not in cookies
- Django provides an administrator function

5) Libraries Used

Some of the Django libraries used include:

1. Django-extensions
2. Django-oscar: for building ecommerce sites eg django.contrib.admin, django.contrib.auth.decorators login_required (for authorization)
3. django-push-notifications: For storing and interacting with push notifications eg django.db.models Model (for database), django.db.models BooleanField, django.db.models CharField, django.db.models DateTimeField (for date/time field)
4. Django REST Framework
5. Django-allauth: for authentication
6. AuditLog: For logging changes to python objects

(Makai, N.D.)

References

Makai, M., N.D.. Django Extensions, Plug-ins and Related Libraries. [Online]
Available at: https://www.fullstackpython.com/django-extensions-plug-ins-related-libraries.html
[Accessed 19 December 2022].