First4Aid

Application Demonstration

SSDCS_PCOM7E June 22 - Team Least Privilege

1900 BST, 25 August 2022

least privilege

Agenda

- Use Case Overview
- Key Application Features
- Application Security
- Testing
- Application Demonstration
- Lessons Learned

Use Case Overview

- Non-Governmental Organisations (NGOs) provide aid to those in need across the globe.
- Aid providers and requestors need a secure way to communicate to exchange information for assistance.
- Security is of paramount importance to protect at-risk populations from further harm.

Key Application Features

- Secure Repository encrypted at rest
- End-to-end application encryption
- GDPR compliant application security
- Hardened against OWASP Top 10 Vulnerabilities
- Distributed microservice architecture using Docker
- Concurrent threading for CRUD functions, alerts, and logging

Application Security

- Secure Repository
  - SQLAlchemy is the Object Relational Mapper (ORM) connecting Python objects to data storage
  - Data will be encrypted with BCrypt and PyCryptodome; encrypted upon form submit
  - Prototype uses SQLite for back-end; production version uses PostgreSQL
- Authentication
  - Username and password hashes will be stored using AES-256
  - Login information to be encrypted upon form submission
- Authorisation
  - Users authorized Create, Read, Update, Delete (CRUD) functions through principle of Least Privilege
  - Aid Requestors can only create, read, and update their own records; can only read received messages
  - Aid Providers can only create, read, and update their own records; can only read received messages
  - Aid Provider Admins can create, read, update, and delete all records and users; can authorise Providers access to cases
- Event Monitoring
  - End goal is for the application to log all logins, access attempts, and CRUD operations for all users

Function-Specific Security Concerns

Function	Concern	Remedy
Data protection: only requestor, assigned provider, and admins access a case	Unauthorised access to cases by registered users	Apply user-based and role-based access cases, validate login credentials for each route and template that reveals data (Zero Trust principle)
CRUD permissions for aid requests	Unauthorised CRUD operations	Minimise data exposure to users and roles: only show data necessary, restrict CRUD to necessary functions (Least Privilege principle)
Data input (primarily for aid requests, but applicable to all forms)	Cross-Site Scripting	Validate data, properly escape inputs and jinja tags, use Cross-Site Request Forgery token from WTForms library

Testing

- Automated testing (mock variables, objects, etc.)
  - Unit and integration testing with Pytest
  - Mock variables/objects
- Manual testing
  - Test plan based on scenarios/use cases
  - Cross-platform compatibility testing
  - Usability testing
- Automated testing
  - Selenium – source code analysis
  - Burp Suite – web application security, penetration testing
  - Nessus – web application security

Application Demonstration

- I am an aid provider, I need to set up First4Aid (Provider Admin role)
  - create Provider users
  - monitor system (admin logging)
  - CRUD operations (cases, messages)
- I am an aid requestor, I need to request aid (Aid Requestor role)
  - create an account
  - submit a request for aid
  - monitor case progress
  - demonstrate: only have access to their own requests and received messages
  - demonstrate: MFA implementation
- I am an aid provider (Aid Provider role)
  - receive an aid request
  - respond to an aid request
  - close out a case
  - demonstrate: aid providers only have access to assigned cases

Lessons Learned

- Instead of using both Flask and Django, it made more sense to select a single framework (Flask) due to its simplicity
- For a dedicated development effort, we would implement Agile methodology with daily scrums; however, using a Kanban board through Trello was more suited for part-time development and a geographically distributed workforce
- Database structure changed in development for easier implementation of role-based access.

References

- Docker (n.d.). *Enterprise Application Container Platform*. [online] Docker. Available at: https://www.docker.com/. [Accessed 22 Aug. 2022].
- GitHub (n.d.). *GitHub*. [online] GitHub. Available at: https://github.com/. [Accessed 22 Aug. 2022].
- JetBrains (n.d.). *PyCharm*. [online] JetBrains. Available at: https://www.jetbrains.com/pycharm/. [Accessed 22 Aug. 2022].
- Jupyter (n.d.). *Project Jupyter*. [online] Jupyter.org. Available at: https://jupyter.org/. [Accessed 22 Aug. 2022].
- OWASP (2021). *OWASP Top Ten*. [online] Owasp.org. Available at: https://owasp.org/www-project-top-ten/. [Accessed 22 Aug. 2022].