Project Implementation Report

Name: Alexander McKenzie

Student ID: S1507940

Server Side Implementation and Security Controls

Python was used to implement server side code utilising the flask micro-framework. Flask was used for handling of certain security features, examples include:

- Session and token management
  - Tokens held within sessions are encrypted using the SHA384 encryption algorithm , and flask sessions employ the use of symmetric cryptography, which defends against the decryption of intercepted session data.
- Werkzeug library access
  - Flask is built upon Werkzeug. Werkzeug manages web-app security details such as page redirections, sending and receiving of cookies and data hashing functions.
  - Werkzeugs encryption and data hash checking functions were leveraged to securely store and validate sensitive data where possible (e.g. usernames and passwords).
- HTTP authentication
  - Flask was used to perform HTTP authentication.
  - Within the application, *login_authenticate* was called whenever a user attempted to access a page that required login to access.
  - *login_authenticate* redirected users to the login page if they attempted to access login-required pages while not logged in.
- Support for parametrised SQL queries
  - Utilised to defend against SQL injection attempts.

Client Side Implementation and Security Controls

HTML, CSS and the Jinja templating language were used for client side UI.

Jinja was chosen due to the client-side security features it implements, examples include:

- Autoescaping input text
  - When used with Flask, Jinja automatically auto-escapes all input data.
  - This helps prevent XSS attacks, as dangerous characters used in HTML and javascript scripts (such as < and >) are rendered in a format not interpreted by browsers, preventing malicious scripts from running on the client or server.
- Sandboxed execution environment
  - Jinja uses a sandbox environment for the execution of templating code. This is useful in defending against malicious Jinja code execution, as the environment is isolated from the host devices resources and network.
  - Jinja also analyses untrusted sandboxed code for any unsafe operations. Jinja will throw a security error for operations it deems unsafe. Potential unsafe operations include the access of object fields and methods.

Additional Security Controls

- The mac address of the users device is captured and stored during the registration process. The mac address is used to identify the device used during login. Users are denied access when the device's mac address doesn't match the mac address captured during registration.
- Users are directly tied, through user IDs, to the data they have generated, allowing an insight into actions taken by users. A timestamp of when data was created is also logged.
- 'True random' IDs were generated for SQL table data entries. Used to prevent attackers being able to guess the order of database entries, helping prevent SQL injection that utilises entry IDs.
- Enforced a minimum password length of 8
- Enforced special character usage
- Checked password used against a 10000 entry list of the most commonly used passwords. Denied any passwords that were too common.