Escalona, J.M.

Group 10

**Case Study 1 - Authentication**

|  |  |  |  |
| --- | --- | --- | --- |
| **Security Issue** | **Java File** | **Vulnerabilities** | **Solution** |
| No input validation on Login input fields | Login.java SQLite.java | Logging in does not verify login input fields and will let the user proceed to the main screen of the application regardless of the content.  Unvalidated inputs also make the application susceptible to SQL injections. | Login.java: (**IMPLEMENTED**)  Verify that both the username AND password fields are not empty and comply with character validity will be based on the rules on minimum credential requirements upon registering which are:   1. Lowercase letters: a-z 2. Numbers: 0-9 3. Symbols (Username): \_-. 4. Symbols (Password): ~`!@#$%^&\*()\_-+={[}]|\:;"'<,>.?/ |
| No user verification | Login.java  SQLite.java | Users are not necessarily verified to exist on the database before proceeding to checking authenticity.  \*This is not strictly for security purposes but rather includes performance ones. | Login.java: (**IMPLEMENTED**)  Add user verification which will dictate whether the user authenticity will proceed or not. Assumes login input verification returns true.  SQLite.java: (**IMPLEMENTED**)  Implement function that would only return a Boolean value to indicate whether user exists or not instead of the already implemented SQLite.getUsers() which returns all users including plaintext passwords of users. |
| Password visible | Login.java  Register.java | Password visible on screen. | Login.java & Register.java:  Change password fields and confirm password fields from JTextField to JPasswordField. |
| Passwords not hashed | SQLite.java User.java | Passwords are stored and accessed in the DB in plaintext. | SQLite.java: (**IMPLEMENTED**) Implement hashing at a controller level for both password verification (read) and password saving (write).  User.java: (**IMPLEMENTED**)  Change values of ‘password’ to hash equivalent in database.  \*DB was rewritten such that all passwords in plaintext are hashed. |
| No user authenticity verification (i.e., username & password match) | Login.java | User can enter application’s main screen without proper credentials.  Login logging is also recommended to be implemented should login attempts be successful or not. | Login.java: (**IMPLEMENTED**) Implement controls that would verify the username and password before proceeding to the main screen should it be successful. Logging will also be added for audit trail purposes. |
| No invalid login pop-ups/notifiers | Login.java | Should an login-related error occur, the user should be notified so in a manner that is ambiguous or generic. | Login.java: (**IMPLEMENTED**) Implement necessary login error pop-ups that is generic should an invalid log-in condition occur. |
| No login timeout mechanism | Login.java | Repetitive login attempts (whether successful or not) is allowed and does not stop user from logging in after certain retries at a certain time. | Login.java:  Implement login timeout measure should successive missed logins are attempted. |
| No input validation on Register input fields | Register.java  Frame.java | Inputs for user account credentials do not require minimum credential requirements such as minimum length (especially for passwords) and character composition.  Action of register was also found to be lost and accessible (public) on another class file Frame.java | Register.java:   1. Remove Frame.registerAction() and transfer to Register.java 2. Add input validations for each user field such that it meets the following:    1. Lowercase letters: a-z    2. Numbers: 0-9    3. Symbols (Username): \_-.    4. Symbols (Password): ~`!@#$%^&\*()\_-+={[}]|\:;"'<,>.?/ 3. Add input validations for username that meets no uppercase characters |
|  |  |  |  |