Joshua Battaglia

Professor Goggin

CS-499: Computer Science Capstone

19OCT2024

**Software Design and Engineering Narrative**

The artifact I chose to enhance is a reverse-engineered investment application. The original artifact was created on 23 August 2024 and was meant to demonstrate my ability to reverse engineer a legacy application and convert it into a C++ application. The C++ application demonstrated a login-type functionality where the user has to authenticate themselves before having some (simulated) data displayed to them.

            Showcasing this artifact demonstrates my ability to reverse engineer a binary executable and convert it into a more modern programming language. I decided to further enhance the functionality of the C++ application by porting the application into Python to take advantage of the robust library of existing modules and create a UI design for this login screen. Additionally, I chose to create the application modularly so that further enhancements can be made efficiently. The original application was a single C++ file with a couple of security vulnerabilities due to the nature of the C++ programming language.

I met the course outcomes I planned to meet with this enhancement. My goal is to simplify the application’s process and user experience. Turning a console application (only executed within a terminal) into a login screen limits the possibility of adversarial exploits, creates a better experience for the user, and looks more professional. With this first enhancement, I demonstrated an ability to use well-founded and innovative techniques, skills, and tools in computing practices to implement computer solutions that deliver value and accomplish industry-specific goals by implementing a modular software architecture, introducing custom Python packages, utilizing database technology, and taking advantage of Python’s built-in libraries and input handling capabilities.

As I was creating the Python packages for each functionality present within the application such as authentication, and database operations I realized how complex this process can become if you don’t pay attention. Python’s simple syntax is deceptively confusing when you begin to oversimplify naming conventions. One challenge I faced was that I wasn’t sure how to navigate the different custom packages within the main() function. Adding too much into the main function could quickly devolve the simplicity I was aiming for.

**C++ application pseudocode:**

PRINT “welcome” message

WHILE running:

PRINT menu

WHILE choice is not valid:

REQUEST valid menu option

END WHILE

EXECUTE user choice

IF (authenticated == TRUE):

DISPLAY information

ELSE:

REQUEST user to authenticate

END IF

IF (choice == authenticate)

SIMULATE user authentication

END IF

IF (choice == exit):

BREAK loop

END IF

PRINT goodbye message

END WHILE

**Enhanced Python application pseudocode:**

CREATE database

CREATE login instance

DISPLAY login screen

WHILE logging in:

IF user select “Login”:

CHECK database for username

IF username in database:

IF (provided\_password == stored\_password):

LOGIN successful

ELSE:

LOGIN failed

END IF

ELSE:

LOGIN failed

END IF

ELSE IF user select “Create”:

CHECK database for username

IF username in database:

LOGIN failed

ELSE:

ENCRYPT password

STORE username AND password in database

END IF

END IF

END WHILE