**CS-499 Computer Science Capstone**

**(Narrative)**

**SNHU, Justin D. Perez  
Professor Mike Alesso  
November 23, 2024**

**Enhancement Two: Reverse Engineering Artifact for Algorithms and Data Structures**

The artifact selected for this milestone is a reverse-engineered program reconstructed from disassembled assembly code. It was developed during my work in reverse engineering practices as part of my CS 410: Reverse Software Engineering coursework. This artifact was initially created to analyze low-level machine code, infer its high-level logic, and rebuild a functioning C++ program.

The original artifact was a Client Management Application with functionality for managing customer choices and processing sensitive data. This functionality was crucial, providing a real-world scenario for applying reverse engineering and secure coding practices. The assembly code, derived from an ELF executable file, provided a foundation for reconstructing logic and designing algorithms to handle the program’s functionality. The project emphasized identifying vulnerabilities in data handling, implementing input validation, and ensuring secure algorithms for processing user interactions.

This artifact fits the Algorithms and Data Structures category because it optimizes logic and secures data structures in a reconstructed program. It demonstrates my ability to analyze low-level code, rebuild it using algorithmic principles, and enhance its efficiency and security. The enhancements further showcase my skills in refining algorithms, validating inputs, and ensuring secure data handling through modular design and encryption mechanisms.

The artifact was improved through the following enhancements:

1. **Strengthened Input Validation and Data Sanitization:** Input validation mechanisms were implemented to prevent buffer overflows and injection attacks, ensuring secure and reliable data processing.
2. **Optimized Memory Management:** Areas in the code where memory leaks could occur were identified and updated with smart pointers to manage resources effectively and improve efficiency.
3. **Secure Sensitive Data Handling:** An encryption mechanism was added to handle sensitive data securely, aligning with best practices for protecting client information.
4. **Improved Code Documentation:** Detailed comments were added to document vulnerabilities identified in the original artifact, explain the fixes implemented, and outline secure coding practices.

The planned enhancements align with several course outcomes:

* **Outcome 3:** The reconstruction and enhancement of the artifact demonstrate my ability to design and evaluate computing solutions using algorithmic principles. For example, adding input validation **algorithms mitigates security risks and ensures robust functionality.**
* **Outcome 4:** Using smart pointers and encryption mechanisms showcases innovative techniques and tools for computing practices to implement solutions that align with industry-specific goals. The enhancements I planned in Module One were successfully implemented, meeting the outlined objectives. These updates ensure the artifact reflects my skill level and adheres to the best software development and security practices, marking a significant accomplishment in my journey.

The enhancement process followed a structured plan developed during the initial review of the artifact. Planned enhancements focused on securing input handling, improving memory management, and ensuring sensitive data was encrypted. The pseudocode for validation and memory management informed the implementation and was a roadmap for the enhancements.

I gained a deeper understanding of integrating secure coding practices into reconstructed artifacts during enhancement. Implementing input validation and encryption mechanisms reinforced the importance of proactive vulnerability mitigation in software development. The introduction of intelligent pointers highlighted the value of modern memory management techniques in preventing resource leaks and improving program efficiency.

One challenge was integrating modern practices such as encryption and intelligent pointers into a reconstructed artifact while preserving its original functionality. Understanding the artifact’s architecture and dependencies was critical to avoiding regressions. Designing robust input validation mechanisms for various edge cases also required iterative testing and debugging. These challenges strengthened my problem-solving skills and attention to detail.

The enhancements made to this artifact demonstrate my ability to reverse-engineer low-level code into a functional, secure, and maintainable high-level program. These improvements showcase my skills in algorithms and data structures, vulnerability assessment, and secure software design. Including this artifact in my ePortfolio highlights my capabilities in solving complex problems using algorithmic principles, aligning with the Computer Science program’s overarching outcomes. Refining this artifact based on feedback will be a solid testament to my growth and expertise in the field.