Enhancement One: Software Design and Engineering

Jeff Sowards

Southern New Hampshire University

Computer Science Capstone 20EW6

**Category One: Software Engineering/Design**

The artifact for category one will be the final submission Final Project: Security Brief for CS 410: Reverse Software engineering. With that, the plan being is to refine the comments and translations from binary to assembly, and then refinement of C code for an enhanced understanding. Thus, allowing for the skills and abilities within Software design and engineering to be captured, assessed, and dispensed adequately.

A picture containing text, map

Description automatically generated

The artifact for category one will be the final submission Final Project: Security Brief for CS 410: Reverse Software engineering. This shows the ability to convert binary to assemble, understand the process and create C code from the assembly language. By doing so, this allows for the code to be recreated and understand the utilization fundamentals of the code. This section of code was created June 22, 2020 for class CS-410.

This artifact is included in the ePortfolio because the conversion of a binary file into an human understandable language file that was of high interest. Also, showing the ability to have a deeper understanding of how programs interact with hardware was intriguing. Further, this showcases the ability to take machine language, convert it into a legible language that utilization hardware memory allocations, and then understand how the program worked to create original source code. This allows the developer to understand or discover vulnerabilities within the code and memory allocations. While evaluating the artifact, areas such all formatting was explored and syntax for the C code creation. As other areas of the artifact are generated through the use of GCC application, these were not of concern. During evaluation various potential flaws were discovered. Passing or possessing a password in the clear, and other small formatting issues. In additional, there is a lack of comments within the code that don’t allow for easy understanding of the code later on. These were added to ensure understanding and developed processing later on. Moreover, the code appeared to lack formatting overall within the artifact, but after injecting the code into a ATOM to evaluate the code, it was shown to have proper formatting for evaluation purposes.

When evaluating and checking the objectives originally outlined, it was shown that sections were not able to be enhanced as believed. However, other area like code refinements and comments were capable. The plan was to refine the comments and translations from binary to assembly, and then refinement of C code There will be no updates to outcome-coverage plan as most of the plan is capable of being conducted outside of not being able to refine the computer GCC process which is an automated computer processes.

When reflecting on the process of enhancing and/or modifying the artifact, area such as comment process and further code understanding were further instilled. The utilization of this code antitype is not usually used and when having to circle back to the archetype the understanding of C code is better understood. With that, when lacking proper comments, later understanding of the code is obscured and delay because holistic code evaluation is needed. This created challenges when vetting the artifact for changes or enhancements.