Jeremiah Norcross  
Southern New Hampshire University

CS 499  
**Milestone Two Narrative**

The artifact I chose to enhance is an encryption program from a Secure Coding course I took, and it was created in April of 2021. It is written in C++. The original program takes an input text file (with some pre-generated text), loads it into a string, and then performs a XOR encryption on it. The program then saves the string into a file called encrypteddatafile.txt. It then does the same thing to decrypt the text, saving the decrypted text into decrypteddatafile.txt.

I chose this artifact because it demonstrates several basic C++ competencies, such as reading from a file, using loops, use of ints and strings, saving data to a file, printing to the terminal, etc. It also demonstrates interest in and understanding of encryption and security, which is important for anyone who wants to get into cybersecurity in the future. I improved this artifact by implementing a menu that allows the user to choose whether to use the already-implemented XOR cipher algorithm, or the newly implemented Caesar Cipher algorithm. This expands the complexity of the artifact in an interesting way, and there is room to add even more encryption algorithms as the code was written in a modular fashion. With the added encryption algorithm and interactivity from the user, I have met the course objectives I planned to meet. The enhanced artifact uses algorithmic principles (in the encryption algorithms) to solve a given problem (namely encrypting strings that would otherwise be plaintext and easier to attack). The code also demonstrates the ability to use well-founded and innovating techniques to implementing computer solutions. The use of encryption algorithms demonstrates a security mindset that anticipates adversarial exploits in software architecture. Due to this, the enhanced artifact meets objectives CS-499-03, 04, and 05.

When enhancing the artifact, one challenge for me was getting the Caesar Cipher to work. The famous Caesar Cipher is something that is easy to conceptualize and do on paper, but is much harder to do with C++. For example, using a key of four is easy to do in your head because we all know the alphabet by heart and can shift the letters easily. However, the algorithm to do that needs to be explicitly stated in the code, and there is no easy shortcut to perform that kind of shift due to the way letters are represented in the code. The solution I ended up using is a rather ugly (yet effective) series of if and else if statements in a for loop that iterates through the string and transforms the characters based on whether they are upper or lower case. Coming up with this solution required me to look on Github and other similar solutions, as well as looking at C++ documentation and learning how characters are represented in the code. I had to learn what it meant to “add” a number to a character to transform it into a different one. Overall, improving the artifact gave me a deeper understanding of the inner workings of the language.