Enhancement 2 Narrative

The artifact that was chosen for Algorithms and Data structure is the buffer overflow example from the secure coding course. This application is an application where a user can compare an input to a given user account number. This was created with c++. Buffer overflow in this example happens if the user input is greater than 20 characters. When buffer overflow occurs in this example the account number is overwritten. Before enhancement a char that was limited to 20 characters was used for the user input that was compared to the account number. While this prevented buffer overflow it did so by cutting off the user input. With a loop the input is only accepted if it valid othersise the loop continues and prompts the user for input again.

I chose this artifact for the eportfolio because it shows an application that was made with c++. This program also shows a mitigation of buffer overflow to ensure data is not overwritten and corrupted. This is important to ensure the code is more secure. The enhancement made to this code improved it by preventing buffer overflow in a more effective way. A user is prompted to provide input until valid input is provided. This was done with a do while loop. A do while loop was chosen because the loop will iterate at least one whether the condition is met or not. The condition in this case is the user\_input\_length is greater than 20. A user will be prompted for input until correct input is provided and the user will be prompted for input at least once ensuring the code works with valid input only and allows the user multiple opportunities to provide valid input. This is an effective way to create a loop that is guaranteed to iterate at least once. The code is simple and organized. It is a short and effective piece of code.

This narrative meets the course outcome “Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts”. This is because this narrative is designed to allow employers to understand the enhancements made to the project. This narrative is technically sound and written in an informal and professional manner. The enhancements I made to buffer overflow meets the course outcome “Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources”. These enhancements made to buffer overflow example meets this outcome by mitigating the buffer overflow to prevent a potential exploit that an adversary can use which in this case is buffer overflow. Buffer overflow can be used in this program to change the account number which may allow someone to gain access to the account after the number is changed. Security is enhanced by preventing buffer overflow in a more effective way using a loop. This allows a user multiple opportunities to provide valid input that does not cause buffer overflow. This possible exploit being prevented also shows a security mindset by showing that a possible exploit is identified and mitigated resulting in more secure code.

The enhancement process was an interesting learning experience. I learned that sometimes a better solution is found while coding than the original plan. My original plan was to use a while loop to check the user input to ensure that valid input was provided in order to prevent buffer overflow. The issue I ran into with the original plan was getting user input and checking it to ensure that buffer overflow could not happen while keeping the code simple. If I went with the original plan, I would have to get user input before the loop as well as in the loop. This is because the loop checks the input to see if it is valid and if it is not it will need to get user input again. I realized that a do while loop would be more effective because a do while loop has the loop iterate at least once and will continue as long as the condition is valid. This means in the loop the user provides input and it is checked, and it repeats until the valid input is provided. The valid input is a string that is less than 21 characters.