# CS 410 Binary to C++ With Security Vulnerabilities Activity Guidelines and Rubric

## Overview

In previous activities, you converted files using languages including binary, assembly, and C++. You have successfully completed a full conversion of legacy code, from binary to C++. In this activity, you will convert legacy binary code to C++, examine the code for security vulnerabilities, fix the identified security vulnerabilities, and report on those vulnerabilities. This is a key step toward the successful completion of Project Two, which is centered on fixing security vulnerabilities. The coding for this assignment will be performed in Codio. You will submit the completed Binary to C++ With Security Vulnerabilities Activity Template Word Document and a new binary file and CPP file.

## Prompt

Specifically, you must address the following rubric criteria:

1. Convert the binary file to assembly code.
    - These files can be found in the Software Reverse Engineering Playground in the Module Six file folder in Codio.
2. Explain the functionality of the blocks of assembly code.
    - Use the Binary to C++ With Security Vulnerabilities Activity Template to complete this step.
3. Convert the assembly code to binary.
    - Create a new binary file for submission.
4. Convert the assembly code to C++ code.
    - Use the Binary to C++ With Security Vulnerabilities Activity Template to complete this step.
    - Compile the C++ code in the Eclipse integrated development environment.
5. Identify the security vulnerabilities within the C++ code.
    - Comment within the C++ code to indicate where the security vulnerabilities are identified.
6. Fix the identified security vulnerabilities within the C++ code.
    - Correct the C++ code to fix the security vulnerabilities.
7. Explain how the updated C++ code fixes the identified vulnerabilities.
    - Comment within the C++ code to indicate how the security vulnerabilities were fixed.

## What to Submit

**Binary to C++ With Security Vulnerabilities Activity Template**

This should be a Word document. Use the Binary to C++ With Security Vulnerabilities Activity Template to convert the legacy binary file into assembly, explain the functionality of the assembly code, and convert the assembly code to C++ code.

**C++ File**

This file includes your comments on the identified security vulnerabilities, the fixes, and the comments about how the security vulnerabilities were fixed. This file is needed to ensure that the code, identified vulnerabilities, fixes, and comments are correct.

**Binary File**

This file is needed to run the application.

### Binary to C++ With Security Vulnerabilities Activity Rubric

| Criteria | Exemplary | Proficient | Needs Improvement | Not Evident | Value |
|---|---|---|---|---|---|
| **Binary to Assembly Conversion** | N/A | Converts binary file into assembly code (100%) | Shows progress toward proficiency, but with errors or omissions; areas for improvement may include converting the binary file into | Does not attempt criterion (0%) | 9 |

| | | | | | |
|---|---|---|---|---|---|
| | | | assembly code (85%) | | |
| **Assembly Functionality** | Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner (100%) | Explains the functionality of the blocks of assembly code with minimal errors and adequate detail (85%) | Shows progress toward proficiency, but with errors or omissions; areas for improvement may include explaining the functionality of the blocks of assembly code with minimal errors and richer detail (55%) | Does not attempt criterion (0%) | 13 |
| **Assembly to Binary Conversion** | N/A | Converts the assembly code to binary (100%) | Shows progress toward proficiency, but with errors or omissions; areas for improvement may include converting the assembly code to binary (85%) | Does not attempt criterion (0%) | 9 |
| **Assembly to C++ Conversion** | Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner (100%) | Converts each block of assembly code to C++ code with minimal errors (85%) | Shows progress toward proficiency, but with errors or omissions; areas for improvement may include converting each block of assembly code to C++ code with fewer errors (55%) | Does not attempt criterion (0%) | 10 |
| **Identify Security Vulnerabilities** | Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner (100%) | Identifies all of the security vulnerabilities within the C++ code with minimal errors (85%) | Shows progress toward proficiency, but with errors or omissions; areas for improvement may include identifying all of the security vulnerabilities within the C++ code with fewer errors (55%) | Does not attempt criterion (0%) | 18 |
| **Fix Security Vulnerabilities** | Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner (100%) | Fixes the identified security vulnerabilities within the C++ code with minimal errors (85%) | Shows progress toward proficiency, but with errors or omissions; areas for improvement may include fixing the identified security vulnerabilities within the C++ code with fewer errors (55%) | Does not attempt criterion (0%) | 18 |
| **Security Vulnerabilities Explanation** | Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner (100%) | Explains how the updated C++ code fixes the identified security vulnerabilities with minimal errors (85%) | Shows progress toward proficiency, but with errors or omissions; areas for improvement may include explaining how the updated C++ code fixes the identified security vulnerabilities with fewer errors (55%) | Does not attempt criterion (0%) | 18 |
| **Articulation of Response** | Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner (100%) | Clearly conveys meaning with correct grammar, sentence structure, and spelling, demonstrating an understanding of audience and purpose (85%) | Shows progress toward proficiency, but with errors in grammar, sentence structure, and spelling, negatively impacting readability (55%) | Submission has critical errors in grammar, sentence structure, and spelling, preventing understanding of ideas (0%) | 5 |
| | | | | **Total:** | 100% |