# CS 410 Project Two Security Report Template

## Instructions

Fill in the table in step one. In steps two and three, replace the bracketed text with your answer in your own words.

Identify where multiple security vulnerabilities are present within the blocks of C++ code. You may add columns and extend this table as you see fit.

| **Block of C++ Code** | **Identified Security Vulnerability** |
| --- | --- |
| vector<int> servChoice{ 1, 2, 1, 1, 2 };  int serv1 = servChoice[0], serv2 = servChoice[1], serv3 = servChoice[2], serv4 = servChoice[3], serv5 = servChoice[4];  string userName; | Global Variables |
| cout << "Enter the number of the client that you wish to change" << endl;    cin >> clientNum; *// Which client to change - 1, 2, 3, 4, or 5*    cout << "Please enter the client's new service choice (1 = Brokerage, 2 = Retirement)" << endl;    cin >> servChoice; *// 1 or 2* | Validation and verification |
| cout << "What would you like to do?" << endl;      cout << "DISPLAY the client list (enter 1)" << endl;      cout << "CHANGE a client's choice (enter 2)" << endl;      cout << "Exit the program.. (enter 3)" << endl;      cin >> userChoice; | Validation and verification |
| if(verified == 1){        break; | Lack of security |

Explain the *security vulnerabilities* that are found in the blocks of C++ code.

For the first security vulnerability that deals with global variable and those variables can be effortlessly accessed and manipulated by an attacker. Now, for the second vulnerability that deals with verifying, and it has to do with the inputs not being verified at all. Which means that the user can input anything, and this can cause the program to have problems, and this is where malicious characters can be entered. The third vulnerability also has to do with validation and verification. This has to do with the display menu where there is no input validation for the menu choices. This would allow any attackers to input malicious characters to manipulate the program. The last vulnerability has to do with the lack of security, where it is not a very secure way of verifying a user. So, the user input needs to be encrypted and then compared to the database.

Describe *recommendations* for how the security vulnerabilities can be fixed.

For this first vulnerability I am going to switch the variables to strings because this will give easier invalid input handling. The input validation for the second vulnerability can be changed to where the user can only input 1, 2, 3, 4, or 5 and this will let us avoid undefined behaviors, malicious characters, and any crashes. That is done for the other input validation vulnerabilities where you will only be able to input a number for whatever that choice correlates too.