# 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** |
| --- | --- |
| bool checkUserPermissionAccess()  {      string username;      string password;      cout << "Enter your username: \n";      cin >> username;      cout << "Enter your password: \n";      cin >> password;      if (password == "123")      {          return true;      }      return false;  } | User authentication does not validate the username  Password is not encrypted |
| cout << "Enter the number of the client that you wish to change\n";  int clientNumber;  int newChoice;  cin >> clientNumber;  cout << "Please enter the client's new service choice (1 = Brokerage, 2 = Retirement)\n";  cin >> newChoice; | No input validation for either clientNumber or newChoice |
| cout << "What would you like to do?\n";          cout << "DISPLAY the client list (enter 1)\n";          cout << "CHANGE a client's choice (enter 2)\n";          cout << "Exit the program.. (enter 3)\n";          cin >> choice;          cout << "You chose " << choice << endl; | No input validation |

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

Lack of username authentication means you only need to know the password to get into the program. The password not being encrypted means, like I have, the password can easily be extracted with reverse engineering.

The absence of input validation means in the second and third sections of code the program will enter an endless loop if the input data type is not an integer

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

For the username and password validation I used the hash function to get the hash values for the username “david” and the password “123” and set these as global constant variables for this sample code. Once the user enters their username and password it is hashed in the same manner and then compared to the hashes that are already stored in the program.

For input validation I created a function that validates that the characters in the input are all numbers and then verifies that the input is within the valid range for the current input segment. Then I used this function to get a number input when having the user select an option on the screen.