# 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** |
| --- | --- |
| cout<<"Enter the number of the client that you wish to change"<<endl;  cin>>newChoice;  cout<<"Please Enter The Clients new Service(1 = Brokerage, 2 = Retirement)"<<endl; | Doesn’t check input for cin >> |
| cout<<"Exit (enter 3)"<<endl;  cin>>choice; | Doesn’t check input for cin >> |
| *const* char \*clints[5] = { "Bobby boy", "Sara davy", "Amy lastname","Johnny Smithy", "Carol Speary" }; | Hard coded variables |
| if(password == "123"){  *return* 0;  } | Hardcoded password…. |
|  |  |
|  |  |
|  |  |
|  |  |

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

Whenever you have input from a user you should always assume it will be dangerous and trying to break the software. This cin >> doesn’t check the input at all. Meaning the user could inject some bad code in some cases even wipe a database. Also hardcoded variables is a vulnerability as we learned in this class can be picked from reverse engineering. Although these variables aren’t necessarily important the problem is it could be. For example the \*clients variables isn’t very important it’s just some string(although in the real world a client list is important) but the password is literally hardcoded into program and can be found very easily with some assembly knowledge. Crazy to think this was a real thing up until a few years ago.

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

For all of the input validation you could add some sort of checking method to ensure that the string added by the user is not lethal or dangerous. Then for the variables you could encrypt with an AES encryption that way even in the assembly they shouldn’t be able to see the variables. For the password being hardcoded I would remove that method completely and try from a different angle of approach.