# 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** |
| --- | --- |
| void CheckUserPermissionAccess()  {  string username;  string pass;  bool answer = false;  do {  cout << "Enter your username: " << endl;  cin >> username;  cout << "Enter password for " << username << ":" << endl;  cin >> pass;  if (pass == 123) {  answer = true;  } else  {    cout << "Invalid Password. Please try again..\n" << endl;  }  } while (answer == false);  return;  } | The vulnerability here is that only password is compared when attempting to login, you can enter any username you want and it will allow it as long as the password is correct which is a security risk |
| void DisplayInfo()  {  cout << "\tClient's Name" << "\tService Selected (1 = Brokerage, 2 = Retirement)" << endl;  for (int i = 0; i <= 4; i++)  {  cout << client[i].id << ". ";  cout << client[i].name << " selected service option: ";  cout << client[i].service << endl;  }  cout << "\n" << endl;  } | The vulnerability with this code is that it shows all client information without having to know anything. This in amongst itself is a security risk because then potential hackers would know exactly what they were working with. |
| if (pass == 123) {  answer = true;  } | The vulnerability with this is that the password is 123 which is statically set in the code which is another huge security risk. |
| void ChangeCustomerChoice()  {  int change\_choice = 0;  int new\_service = 0;  cout << "\nEnter the number of the client that you wish to change" << endl;  for (int i = 0; i <= 4; i++)  {  if (i == 0) {  cout << "(";  }  if (i == 4) { cout << ")" << endl;  } else {  cout << i + 1 << " = " << client[i].name << ", ";  }  }  cin >> change\_choice;  cout << "\nPlease enter the client's new service choice (1 = Brokerage, 2 = Retirement)" << endl;  cin >> new\_service;  client[change\_choice - 1].service = new\_service;  } | The vulnerability in this code is that it accepts any value for a customer and is not limited to the number of clients, the same can be said for the actual service. User entry isn’t checked and limited to 1 or 2. |

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

When the CheckUserPermissionAccess function we can see that the password is the only field that is compared when the user enters their username and password. This lowers the security level of the application and creates a huge vulnerability in the code which would allow hackers to easily gain access to the code. Another vulnerability that is in the same block of code is the hard coding of the password, which is 123, only allowing for a password of 123 is a vulnerability and having it hard coded in code is another security risk.

In the DisplayInfo function we show users all client information, if a hacker were to gain access to this information it would be considered a data breach. So essentially by displaying all client information we technically are creating a data breach.

In the ChangeCustomerChoice function we see that it will accept any value entered for the client and/or the service. This is a security risk because it allows for string injection or string manipulation and could possibly present a real big problem for a developer.

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

I would start with the CheckUserPermissionAccess function, I would make sure the software checked for a username and then compare that username against a password that is set for that username. I would then designate individual passwords for each user that is different and much more difficult to guess than 123. The use on this program the password must be hard coded which is a risk but if I were developing production software the password would be located on an external server that would require authentication to access.

For the DisplayInfo and ChangeCustomerChoice functions, I would start by removing the displaying of customer information and would make the user specifically enter in the client’s name which must match before showing client information. The entry would be checked against the list of clients so the user could not just enter anything which would reduce the likelihood of string manipulation or injection. I would also put in a check to make sure the user was logged in before displaying client information. That was a user could not gain unauthorized access to the system and then get client information.