# 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.

1. **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** |
| --- | --- |
| int CheckUserPermissionAccess(){  int x;  string password = "123";  string input;  cout<<"Enter your username:"<<endl;  cin>>input;  cout<<"Enter your password:"<<endl;  cin>>input;  if(password.compare(input) == 0){  x = 1;  }else{  x = 2;  }  return x;  } | CWE-259, use of Hard-coded password.  CWE-307, Improper Restriction of Excessive Authentication Attempts,  No failed attempts time-out.  CWE-284, Improper Access Control  Username not checked.  CWE-20 Improper Input Validation. |
| do{  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>>x;  cout<<"You chose "<<x<<endl;  switch(x){    case 1:  DisplayInfo();  break;    case 2:  ChangeCustomerChoice();  break;    case 3:  break;    }  }while(x != 3); | CWE-20 Improper Input Validation. |
| int num1 = 1, num2 = 2, num3 = 1, num4 = 1, num5 = 2;  string Bob = "Bob Jones", Sarah = "Sarah Davis", Amy = "Amy Friendly", Johnny = "Johnny Smith", Carol = "Carol Spears"; | Hard-coded account information |
| do{  x = CheckUserPermissionAccess();  if(x != 1){  cout<<"Invalid Password. Please try again"<<endl;  }  }while(x != 1); | CWE-200, Exposure of Sensitive Information to an Unauthorized actor.  Ie: the unauthorized user is informed the password is incorrect. |

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

CWE-259 deals specifically with passwords that are hardcoded. According to the CWE Team(n.d), “A hard-coded password typically leads to a significant authentication failure that can be difficult for the system administrator to detect”. This vulnerability has a high likelihood of exploitation and given that I was able to figure out the password from reversing, I agree. Hard-coded passwords are usually introduced in the Architecture and Design phase as a necessity, but if not removed before release, pose a serious threat.

CWE-307 deals with how many times a user can attempt to log in before some preventative action is taken. This makes the software vulnerable to brute-force attacks (CWE Team n.d). A user can perform as many authentications attempts as possible and eventually guess a weak password.

CWE-284 deals with Authentication, Authorization, and Accounting. This program fails to properly implement any of these security concepts. I had a hard time finding a specific CWE that deals with not checking the username, and only checking the password. I chose this as a catch-all. If the username is not checked, then authentication has not happened. Additionally, this program does not log authentication attempts, or who changed user profiles, which also falls under CWE-284. Without these functions, changes cannot be tracked, and exploits will go unnoticed.

CWE-20 is improper user input validation. The user is prompted to enter a number to indicate a response, but the software does not ensure the user inputs values or data that the program is expecting. Improper input validation may result in “altered control flow, arbitrary control of a resource, or arbitrary code execution”. In this case, unexpected input results in altered control flow.

CWE-200 is the exposure of sensitive information to an unauthorized actor. This is a complicated way of telling the user that they had the password wrong specifically. Usually, during log-in, the system tells you as little information as possible to make it harder for people to simply guess passwords. An improper log-on should say vague phrases like “invalid credentials” or “username or password are incorrect”. Our program specifically tells the user the password is incorrect.

Hard-coded user information. I could not find a specific CWE for this particular vulnerability, mainly because I could not determine how sensitive this information would be considered.

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

Most of the vulnerabilities occur because this program does not have a back-end system to deal with the data it manipulates. CWE-259 could be solved by storing salted passwords in a database and creating an interface in the program to retrieve and compare the passwords. This also solves part of the CWE-284, because in order to retrieve stored passwords you’d need to create a relationship between the password and the username or another identifying feature. Implementing a back-end can also alleviate CWE-307, as a lock can be placed on the stored account with a database flag, rather than based on the instance in the running program. What I mean by that is that simply creating a logic loop that counts log-in attempts inside the program will not properly solve CWE-307. The user could just exit and restart the program to reset their attempts to zero. Using a properly configured database would be able to manage sessions and enforce log-in fail attempts correctly. The last issue that is solved by using a properly configured database is the hard-coded user information. The user information can be stored on the database and simply having the client program would not provide the attacker with potentially sensitive information. Instead, they’d also need access to the database, or at very least, the file structure storing the information. The database could even be encrypted while the data is at rest, further protecting critical data.

Changes that can be made inside the program include addressing CWE-20 and CWE-200. In project six I created a simple input validation function that prevented the program from behaving erratically when given unexpected input. That function can be added anywhere “cin” occurs. To correct CWE-200, the phrase “Invalid Password. Please try again”, should be changed to “Invalid login credentials”.

References:

CWE Team. (n.d.). *Common weakness enumeration*. CWE. Retrieved October 12, 2021, from https://cwe.mitre.org/.