# CS 410 Project Two Security Report Template

# Paloma Rodriguez CS410 2/16/24

## 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** |
| --- | --- |
| #include <iostream> |  |
| void DisplayInfo()  std::cout << "DisplayInfo()" << std::endl;  } |  |
| int CheckUserPermissionAccess() {  int result = 0;  // Function calls  DisplayInfo();  DisplayInfo();  DisplayInfo();  if (result == 1)  DisplayInfo();  else if (result == 2)  DisplayInfo();  else if (result == 3)  DisplayInfo();  else  result = 0;  return result;  } | Result is assigned a value of 0 by default and code doesnt validate it against any values before using it to make decisions.  DisplayInfo() is called multiple times w/o any validation which could lead to misuse of this function.  Function could potentially allow unauthorized access. |
| int main() {  // Add an output statement to the beginning of the C++ code  std::cout << "Created by Paloma Rodriguez" << std::endl;  int result = CheckUserPermissionAccess();  if (result == 1)  DisplayInfo();  else if (result == 2)  DisplayInfo();  else if (result == 3)  DisplayInfo(); | Code doesnt validate the input value from CheckUserPermissionAccess() function before using it to make decisions.  DisplayInfo() is called based on the value of result obtained from CheckUserPermissionAccess(), which could lead to abuse use of this function.  Theres no logic to handle errors or exceptions that might occur. |
| return 0;  } |  |

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

**[In a paragraph or two for each security vulnerability, explain in detail how and why these are security vulnerabilities.]**

Input validation is crucial for making sure that the data used by a program is within our expected scope. The variable `result` I assigned a value without any validation. This means that an attacker could possibly manipulate the value of `result` to bypass access control mechanisms or trigger unexpected behavior in the program. Without proper validation, the program may accept invalid inputs, leading to vulnerabilities such as buffer overflows, integer overflows, or injection attacks. I also invoked the `DisplayInfo()` function multiple times without sufficient validation or restriction. Bad actors could exploit this by injecting their own functions or commands, leading to unauthorized access, data leakage, or attacks like command injection or SQL injection. The `CheckUserPermissionAccess()` function does not have permission control logic, without proper permission control mechanisms attackers may exploit weaknesses in access control to escalate privileges, gain unauthorized access to sensitive resources, or perform unauthorized actions within the system. This is essential for enforcing the principle of least privilege ensuring that users only have access to the resources and functionality necessary for their own tasks.

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

**[In a paragraph or two for each recommendation, describe how you would fix these vulnerabilities.]**

Some recommendations to address the security vulnerabilities, it would be important for me to implement detailed input validation, function call restriction, and have detailed permission control mechanisms. First, validate user inputs such as the `result` variable in `CheckUserPermissionAccess()` to ensure it falls within expected scope and mitigating the risk of unauthorized inputs and injection attacks. Second, restrict and validate functions like `DisplayInfo()` based on user permissions, utilizing access control mechanisms, and including input validation checks within functions to prevent misuse. Lastly, enhance permission control logic within `CheckUserPermissionAccess()` while also including error handling to manage unauthorized access attempts.