Template based on the Centers for Medicare & Medicaid Services, Information Security & Privacy Management's Assessment

Security Assessment Report

Erick Rodriguez

Version 1

May 1, 2023

Table of Contents

1. Summary 3
    1. Assessment Scope 3
    2. Summary of Findings 3
    3. Summary of Recommendations 5
2. Goals, Findings, and Recommendations 5
    1. Assessment Goals 5
    2. Detailed Findings 5
    3. Recommendations 6
3. Methodology for the Security Control Assessment 7
4. Figures and Code 9
    4.1.1 Process flow of System (this one just describes the process for requesting)........9
    4.1.2 Other figure of code 9
5. Works Cited 9

1. Summary

The overall goal of this project was to identify and possibly fix potential security weaknesses in a C++ recreation of Pokemon Blue's battle gameplay. Through the use of Github, resources from the course Canvas page, and solutions from strackoverflow.com I was able begin working on identifying and fixing hazards in the program.

1. Assessment Scope

- **Tools(Websites)**
    - GitHub: This was used to host and make changes to my program online thorugh a secure outlet.
    - Canvas: Resources such as a "SWOT Analysis" template was used to help me orientate myself when searching for security flaws in my code.
    - ChatGPT: This was used sparingly and was mainly for minor coding hickups I came across.

- **Browsers**
    - Safari: This was used whenever I worked on this project on-the-go and was mainly used to look up any solution to potential issues I came across.
    - Opera: This was the browser I used the most for research and using resources provided through Canvas.

- **Software**
    - CLion: It was the one and only IDE which I used to write and execute the program.

- **Tested Software**
    - Pokemon C++ Project: This was the software that was being tested, which was created on the Fall 2022 semester as a group project for the Porgramming 2 course. The program is a C++ interpretation of a simplified battle scenario for Pokemon Blue.

- **Limitiations**
    - Due to my novice experience with SFML I had a rather difficult time sgetting to work on MacOs, therefore, no real testing on any OS besides Windows 10 was able to be performed.
    - Time was another limitation as I had a lot of personal issues come up during the semester and was not able to implement as many fixes as I would've liked.

2. Summary of Findings

Of the findings discovered during our assessment, 0 were considered High risks, 2 Moderate risks, 0 Low, and 0 Informational risks. The SWOT used for planning the assessment are broken down as shown in Figure 1.

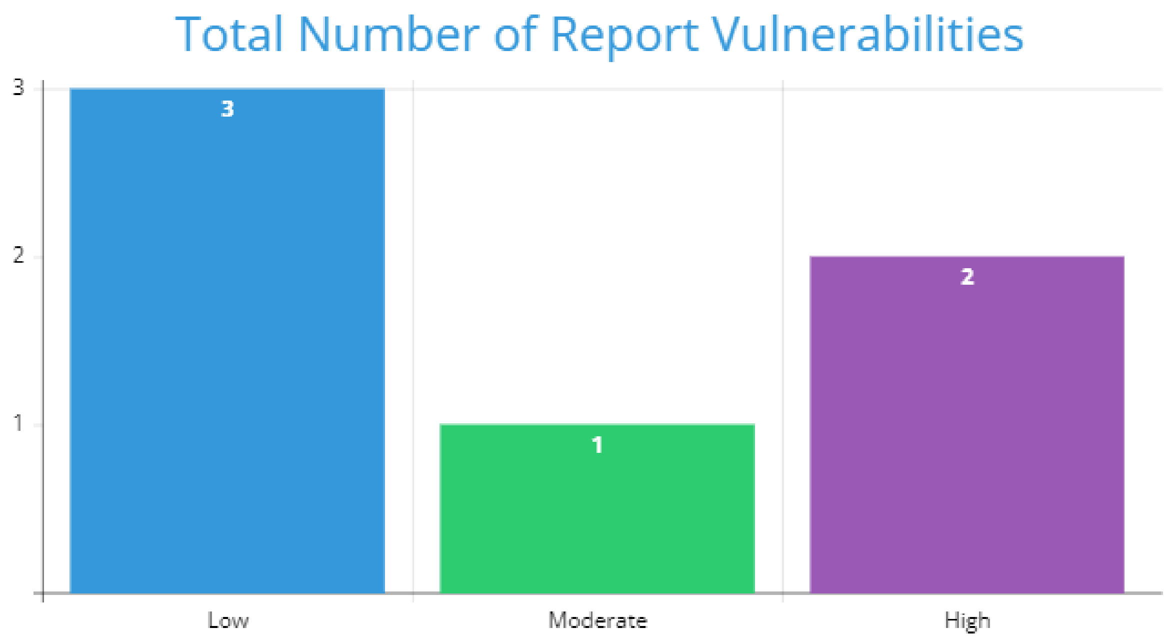

Figure 1. Findings by Risk Level

Explain above and link to full table of explanation of top risks like Figure 2.

SWOT ANALYSIS

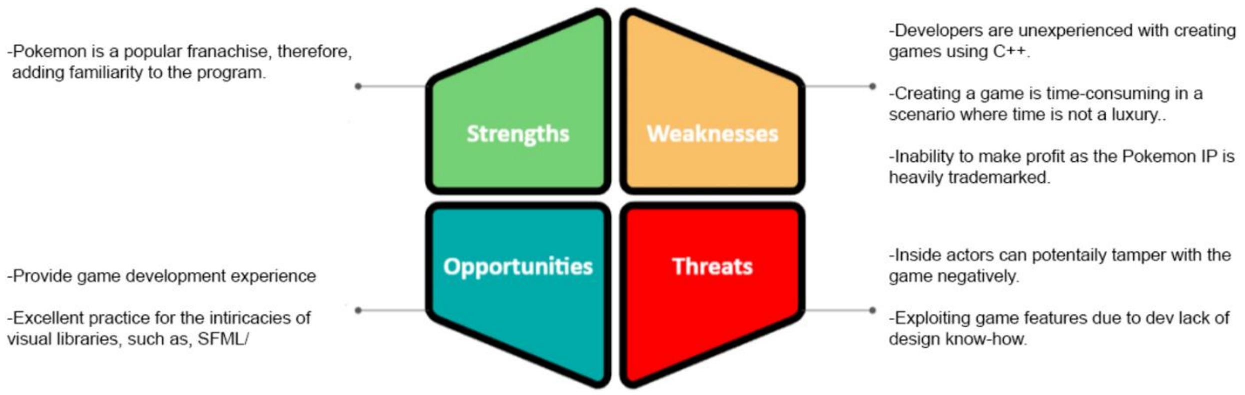
-Pokemon is a popular franachise, therefore, adding familiarity to the program.
Strengths
Weaknesses
-Developers are unexperienced with creating games using C++.
-Creating a game is time-consuming in a scenario where time is not a luxury..
-Inability to make profit as the Pokemon IP is heavily trademarked.
-Provide game development experience
-Excellent practice for the intiricacies of visual libraries, such as, SFML/
Opportunities
Threats
-Inside actors can potentaily tamper with the game negatively.
-Exploiting game features due to dev lack of design know-how.

Figure 2. SWOT

SWOT explanation…

3. Summary of Recommendations

A variety of changes were made to the original program such as, including a partial fix to the first instance of a hardcoded filepath by having the path be relative hopefully preventing them from being exploited. Howwever, I have yet to implement a full fix for a potential buffer overflow located in the “renderBattle” function cause by pointers, the temporary change was by using vectors. Another change that needs to be made is having a proper method to validate a user’s mouse inputs.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to do the following:

- Determine that application was securely maintained by way of GitHub.
- Verify that third-party libraries, SFML, are secure and up to date.
- Scanning for any vulnerabilities and coming up with a fix.
- Enforce secure coding practices.

2. Detailed Findings

Vulnerabilities	**Risk Value**	**Explanation**
Hardcoded Filepaths	High	This program relies upon quite a bit on files for its visuals, therefore there are a lot of instacnes of hardcoded filepaths through the entire program. Due to the nature of their pathing it would not be difficult for someone to locate them and exploit them. This is deemed a high risk as there due to the systems reliance on these files and if they were to be tampered with it may be catastrophic.
Buffer Overflow	Moderate	There are multiplace cases of pointer being used in this program to change data. However, these pointers seem to cuse a buffer overlow that may lead the program to be prone to crashes. This was found to be an issue after working on the “Bad Code” assignment for the Computer Security course. This is classified as a moderate risk as this is a game and crashes would be detremental to the player’s enjoyment but it is not too prevalent of an issues throughout the system.
Lack of Input Validation	High	The entirety of the user’s interactions with the system is through mouse clicks. However, thereis no way to validate if the user is clicking in a athorized place. Due to the prevalence of this feature, not having a

		way to moderate it can lead to unathorized actions deeming it a high risk.
SFML Not Up to Date	Low	The program uses the third-party library SFML for its visuals. Upon my research it appears that SFML is not up to date and can lead to visual issues and lower performance which may hinder the user's experience. However, as SFML is a reliable library having an outdated version won't be catastrophic but should not be ignored, hence the low risk.
Lack of Physical Security	Low	The physical aspect of the program is located within the bounds of my bedroom, meaning it does not hae the traditional security that code stored in a professional environment would have. This is a low risk as no one is planning a spec ops mission to steal my heardrives yet.
Internal Actor Threat	Low	As it is just me working on the program there is not a major risk of another internal actor tampering, thus it being classified as low. However, precautions should still be made.

3. Recommendations

Vulnerabilities	Difficulty	Fix Description
Hardcoded Filepaths	Moderate	A fix for this issue would be to have all the file paths be relative, thus making their location harder to pinpoint. This is of moderate difficulty solely based on how often file paths are used in the program, taking some time.
Buffer Overflow	Moderate	As pointers are the cuase for the buffer overflows a fix would be to use vecotrs instead. This is of moderate difficulty due to me not being all to comfortable dealing with vectors in this situation and lead to complications.
Lack of Input Validation	Very	A fix for this issue would be to add a method to properly validate user inputs nad ensure they are withing proper bounds. However, due to my lack of experience with having mouse clicks as inputs and how big of a part it plays in the program, it will take quite a bit of time and effort to fully implement a fix for this.
SFML Not Up to Date	Easy	This issue is fixed by sismoky downloading and installing the latest version of SFML. Then update the library files in the project and any code that uses out

		of date methods. This is a fix that might take some time but is relatively simple.
Lack of Physical Security	Easy	This fix entails ordering a consumer level security system to ensure no unathorized entitied interfer with the physical state of the program. Since installing the security system is a separate process, the actual program is unaffected and can run normaly.
Internal Actor Threat	Low	Due to the low risk of a bad internal actor a safeguard would be the use of GitHub's logging for any changes. Dimsihing the chances of a bad inernal actor by means of an easy fix.

3. Methodology for the Security Control Assessment

3.1.1 Risk Level Assessment (delete this text: you don't have to change 3.1.1)

Each Business Risk has been assigned a Risk Level value of High, Moderate, or Low. The rating is, in actuality, an assessment of the priority with which each Business Risk will be viewed. The definitions in **Error! Reference source not found.** apply to risk level assessment values (based on probability and severity of risk). While Table 2 describes the estimation values used for a risk's "ease-of-fix".

Table 1 - Risk Values

Rating	Definition of Risk Rating
High Risk	Exploitation of the technical or procedural vulnerability will cause substantial harm to the business processes. Significant political, financial, and legal damage is likely to result
Moderate Risk	Exploitation of the technical or procedural vulnerability will significantly impact the confidentiality, integrity and/or availability of the system, or data. Exploitation of the vulnerability may cause moderate financial loss or public embarrassment to organization.
Low Risk	Exploitation of the technical or procedural vulnerability will cause minimal impact to operations. The confidentiality, integrity and availability of sensitive information are not at risk of compromise. Exploitation of the vulnerability may cause slight financial loss or public embarrassment
Informational	An "Informational" finding, is a risk that has been identified during this assessment which is reassigned to another Major Application (MA) or General Support System (GSS). As these already exist or are handled by a different department, the informational finding will simply be noted as it is not the responsibility of this group to create a Corrective Action Plan.
Observations	An observation risk will need to be "watched" as it may arise as a result of various changes raising it to a higher risk category. However, until and unless the change happens it remains a low risk.

Table 2 - Ease of Fix Definitions

Rating	Definition of Risk Rating
Easy	The corrective action(s) can be completed quickly with minimal resources, and without causing disruption to the system or data
Moderately Difficult	Remediation efforts will likely cause a noticeable service disruption • A vendor patch or major configuration change may be required to close the vulnerability

Rating	Definition of Risk Rating
	• An upgrade to a different version of the software may be required to address the impact severity • The system may require a reconfiguration to mitigate the threat exposure • Corrective action may require construction or significant alterations to the manner in which business is undertaken
Very Difficult	The high risk of substantial service disruption makes it impractical to complete the corrective action for mission critical systems without careful scheduling • An obscure, hard-to-find vendor patch may be required to close the vulnerability • Significant, time-consuming configuration changes may be required to address the threat exposure or impact severity • Corrective action requires major construction or redesign of an entire business process
No Known Fix	No known solution to the problem currently exists. The Risk may require the Business Owner to: • Discontinue use of the software or protocol • Isolate the information system within the enterprise, thereby eliminating reliance on the system In some cases, the vulnerability is due to a design-level flaw that cannot be resolved through the application of vendor patches or the reconfiguration of the system. If the system is critical and must be used to support on-going business functions, no less than quarterly monitoring shall be conducted by the Business Owner, and reviewed by IS Management, to validate that security incidents have not occurred

3.1.2 Tests and Analyses

This was completed using <list and describe any type of testing you performed here>.

- **Tests**
  -
- **Analyses**
  - **Vulnerabilty Management:** Through out the various iterations of this security assessment I have had to identify threats and prioritize them based on their risk, as seen in the two tables in section 2. This helped keep track and put these issues into perspective.

3.1.3 Tools

This was completed using <list and describe any tools used for testing (include Linux Command Line commands>.

- **Tools**
  - **ChatGPT:** I used ChatGPT, sparingly as to no grow dependent on it, to locate any potential issues that I may have overlooked. As well as test out if any of my fixes were optimal by asking it if there were better methods.

4. Figures and Code

Insert any pictures here (including of major code issues or code that was used as a tool – can just screenshot and add link to github). This section must include at least 4 figures or code portions:

4.1.1 Process or Data flow of System (this one just describes the process for requesting), use-cases, security checklist, graphs, etc.

Describe the process flow here.

4.1.2 Other figure of code

HERE

5. Works Cited

-Fixing Bad Code Assignment.