**Logo, company name

Description automatically generated**

**Security Assessment Pokemon Battling System Demo Report**

Version N.1

May 1, 2023

Table of Contents

[1. Summary 3](#_Toc133877573)

[1. Assessment Scope 3](#_Toc133877574)

[2. Summary of Findings 3](#_Toc133877575)

[3. Summary of Recommendations 3](#_Toc133877576)

[2. (Greenwell 2023)Goals, Findings, and Recommendations 3](#_Toc133877577)

[1. Assessment Goals 3](#_Toc133877578)

[2. Detailed Findings 3](#_Toc133877579)

[3. Recommendations 3](#_Toc133877580)

[3. Methodology for the Security Control Assessment 3](#_Toc133877581)

[3.1.2.1 Sandbox IDE Environments and Project files 3](#_Toc133877582)

[3.1.2.2 Whitebox testing & Grey Box testing 3](#_Toc133877583)

[3.1.3 Tools 3](#_Toc133877584)

[4. Figures and Code 3](#_Toc133877585)

[4.1.1 Risk Assessment & Checklist 3](#_Toc133877586)

[4.1.2 Process or Data flow of System (this one just describes the process for requesting), use-cases, security checklist, graphs, etc. 3](#_Toc133877587)

[4.1.3 Other figure of code 3](#_Toc133877588)

[5. Works Cited 3](#_Toc133877589)

# Summary

The goal of the assessment is to secure the program from vulnerabilities and in-game exploits in the source code. The methods used to fix the vulnerabilities was unit testing by throwing source code into a different environment and testing errors and outputs.

## Assessment Scope

The software was developed and compiled for the Clion C++ IDE, on computer running a Windows 11 Pro OS, the program and project is stored on a Github repository. The GUI library used is the SFML GUI Library.

## Summary of Findings

* Comments with improper syntax, grammar, explanation, completeness, etc. **Low risk:**
* Lack of comments. **Low risk**
* Code did not follow guidelines or IDE recommendations. **Moderate risk**
* File paths and other vulnerabilities were shown in the source code. **High risk**
* Processes/vulnerabilities shown in console. **High risk**
* Hard to Navigate folder. **Low risk**
* Lack of security plan or bug report system. **High risk:**

Encapsulation:

* SFML game stage windows were not separated into separate and proper functions and files. **Moderate risk**
* Folders and files were not organized. **Low risk**
* Functions were not organized. **Moderate risk**
* Code for each stage were not properly organized. **Moderate risk**
* Unused assets were not properly organized. **Low risk**

SFML Library:

* Lack of interrupts. **Moderate risk**
* The use of polling concepts needed for work arounds. **Moderate risk**
* Lack of/ needed work arounds for logging or accounting. **Moderate risk**
* Cannot use AIs. **Informational risk**

Pokemon Moves methods & the Battle Stage:

* Ptrs and conditions were improperly implemented with the pokemon objects and selection stage output. **High risk**
* Pokemon’s stats were not balance and caused in game exploitation of the movesets raising health. **Low risk**
* Oppenent AI had a bug were it will not attack when the trainer selects tackle. **Informational risk**
* The use of the random function to use for accuracy. **Informational risk**
* PTRs were improperly used in the moves methods. **High risk**
* Stat balancing with the moves methods. **Informational risk**

Of the findings discovered during our assessment, 5 were considered **High risks**, 7 **Moderate risks**, 6 **Low**, and 4 **Informational risks**. The SWOT used for planning the assessment are broken down as shown in Figure 2.

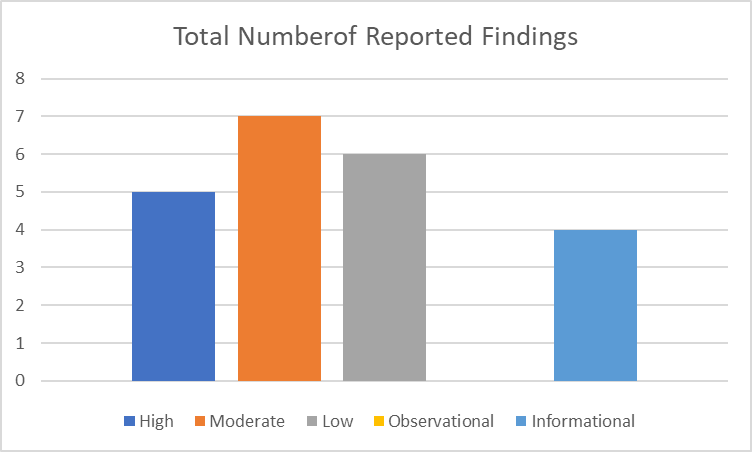


Figure 1. Findings by Risk Level

There are a total of 22 issues that were found and amended, there were less serious vulnerabilities because the product is a small program that doesn’t have any typed inputs, the reason why there were plenty of moderate and low vulnerabilities is that there were guidelines and coding fundamentals were not followed properly. See Figure 2.

Diagram

Description automatically generated

Figure 2. SWOT

All of the weaknesses were fixed or negated to a degree except for the SFML polling issue, that is still needs to patched. Since the vulnerabilities have been removed, the chances of threat actors and number of threats have been reduced.

The top threats are few because of the scale of the program, however these threats can manipulate data in the program in a large scale. Threat actors can exploit the game using the bugged AI and poorely balanced pokemon stats and moves. Threat actors can change the filepath easily to manipulate to go to any location. Ptrs in the moves methods and battle stage could of potentially caused a buffer overflow. The parts of the processes of each stage and some vulnerabilities were displayed in consol which can be used to manipulate other data. The security plan for the product did not exist meaning that there could be users who knew of the bugs but could not report it or notify the developers.

## Summary of Recommendations

* Organize all files, folders, and assets into respective folders.
* Correct the improper use of ptrs.
* Correct the syntax, grammar of comments.
* Additional comments explaining code.
* Debugging code to follow IDE and C++ guidelines.
* Hidden the filepaths with definitions.
* Rebalance the stats of the pokemon and their movesets.
* Separated function definitions into separate files by their stage of the program.
* Reorganize the functions and files by their stage of the program.
* Properly implemented the random function with a seed.
* Added Security plan.

# (Greenwell 2023)Goals, Findings, and Recommendations

## Assessment Goals

The purpose of this assessment was to do the following:

* To ensure that there were no in game exploitations that could be found in the source code.
* To have the program follow the guidelines and regulations of the IDE software and other C++ guidelines to increase overall readability of the source code.
* To organize and further encapsulate/develop functions, files, and folders.
* To add better documentation and understanding of the code.
* To determine if the program is securely maintained.
* To decrease vulnerabilities in the program.

## Detailed Findings

Readability & Comments:

* Comments with improper syntax, grammar, explanation, completeness, etc. **Low risk:** This issue does not pose a direct threat but adds confusion in understanding the source code which can lead to errors and add time to debug the product.
* Lack of comments. **Low risk:** This is issue had cause a lack of clarity and understanding that can lead to errors and further trouble debugging the product.
* Code did not follow guidelines or IDE recommendations. **Moderate risk:** This can result in errors, inefficiencies, and inconsistencies in the source code which can make the product suffer.
* File paths and other vulnerabilities were shown in the source code. **High risk:** This was sensitive data being exposed in the source code that could lead to the data or assets being manipulated, stolen, or leaked which could be detrimental to the product.
* Processes/vulnerabilities shown in console. **High risk:** This can lead to unauthorized access to sensitive information of data or a process which could lead to a data breach.
* Hard to Navigate folder. **Low risk:** This can make it difficult to find and modify source code and other assets which lead to longer time to debug or maintain the product.
* Lack of security plan or bug report system. **High risk:** The lack of a security plan meant that no user can report any bug or problem with the system, meaning that the bugs in the product will exist until the developer discovers it and fixes it. There are less ways of notifying the developer that there is a vulnerability in the program.

Encapsulation:

* SFML game stage windows were not separated into separate and proper functions and files. **Moderate risk:** This had lead to potential code duplication (it has), inefficiencies, issues in maintenance with the product.
* Folders and files were not organized. **Low risk:**  This makes the folder hard to navigate.
* Code for each stage were not properly organized. **Moderate risk:** This has affected the readability of the
* Unused assets were not properly organized. **Low risk:** This made it hard to navigate the folder of the program.

SFML Library:

* Lack of interrupts. **Moderate risk:** This had cause performance issues, debugging issues, and possible memory issues. It has caused crashes and memory overload.
* The use of polling concepts needed for work arounds. **Moderate risk:** This has affected the program by spaming the execution because the loop will execute too fast and will iterate more than expected. This can be manipulated to crash the program or manipulate the data.
* Lack of/ needed work arounds for logging or accounting. **Moderate risk:** This makes the logging or accounting for the program unreliable or unstable due to spaming of executions, this can affect the product in debug and tracking user activity, unauthorized access, and data modifications.
* Cannot use Ais functions due to the loop being unable to stop. **Informational risk:** This only limits functionality and encapsulation of the opponent AI function, but it can affect readability of the source code.

Pokemon Moves methods & the Battle Stage:

* Ptrs and conditions were improperly implemented with the pokemon objects and selection stage output. **High risk:** This had caused buffer overflows and data being sent to unintended memory addresses at one occasion from a condition not being thought of.
* Pokemon’s stats were not balance and caused in game exploitation of the movesets raising health. **Low risk:** Poor balancing with can caused the product to crash in the most unlikely of circumstances that if the hp stat reached beyond INT MAX, then it would crash the program. Other than that, it is affects the gameplay or memory because the game will never end or one pokemon will only get damaged which affected gameplay.
* Oppenent AI had a bug were it will not attack when the trainer selects tackle. **Informational risk:** This did was not a security risk but affected game play and could be exploited in a way the program did not intend.
* The use of the random function to use for accuracy. **Informational risk:** This affected the randomness of the chances and AI, it only affected the gameplay design of the product.
* PTRs were improperly used in the moves methods. **High risk:** This can lead to a buffer overflow and memory.
* Stat balancing with the moves methods. **Informational risk:** This is not a security threat but it affects the fairness and balance of the game.

## Recommendations

* Organize all files, folders, and assets into respective folders.
  + The current source code files, the unused assets, the current frames used by the program, and used assets were put into each respective folder.
* Correct the improper use of ptrs.
  + In the moves methods, currentHealth ptr was used to transfer the changed hp to the pokemon, but it was redundant. The fix was removing it and using just &defend.hpnow.
* Correct the syntax, grammar of comments.
  + Fixed all typos, gramical errors, and used PEP8 spacing and indenting which is slightly cleaner than C++ guide. The IDE doesn’t have a recommendation but it makes it easier to the reader’s eyes.
* Additional comments explaining code.
  + Some parts of the code had no explanation to its purpose, so comments were added, explaining its purpose. Some comments were highlighted to emphasize sections of the code such as the battle section of the user turn and cpu turn.
* Debugging code to follow IDE and C++ guidelines.
  + There were some improper indentions, redundant and poor syntax, and bad coding practices that were removed and replaced with code that follows the guideline.
* Hidden the filepaths with definitions.
  + The filepaths were made into definitions and the plain text was removed from the source code. The definitions took the place of the plain text.
* Rebalance the stats of the pokemon and their movesets.
  + Trial and error was used to fix their battles .
* Separated function definitions into separate files by their stage of the program.
  + This was done to better organize and debug the code, this wasn’t exactly nesseccarry to do, but it betters readability because the methods can be viewed/ tested separately.
* Properly implemented the random function with a seed.
  + Used a seeded value and a time variable to make a proper random number generator to improve the gameplay and a more secure and robust number generator.
* Added Security plan.
  + Added a document that tell users and developers how to report the bug, handle the existing vulnerabilities, and what versions are supported with the security patches.

# Methodology for the Security Control Assessment

**3.1.1 Risk Level Assessment**

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 * 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 separate projects in the same IDE software, using them as sandboxes to test parts of the pokemon move methods. I also used different systems which was a laptop on a Windows 10 Home OS, and a Mac computer on unknown iOS. to test the stability of the program and see if it performs the same way.

### 3.1.2.1 Sandbox IDE Environments and Project files

A different project in Clion was opened and was used as a sandbox for the functions to constantly test outputs and results, the process, and integration into other parts of the program. The first sandbox was in the first project iteration, where the functionality and integration was being tested. While the second sand box is from the second iteration and third iteration, were the process, output and results were observed and reported.

First sandbox: [sandbox\_battle.cpp](https://github.com/Keko787/Pokemon-C-battling-system-Demo/blob/main/PokemonBlueCpp/unused%20assets/sandbox_battle.cpp)

Second sandbox: [sandbox\_battle2.cpp](https://github.com/Keko787/Pokemon-C-battling-system-Demo/blob/main/PokemonBlueCpp/unused%20assets/sandbox_battle2.cpp)

[See Figure 6.](#_Figures_and_Code)

### 3.1.2.2 Whitebox testing & Grey Box testing

The whitebox testing was done with developers because of better insight and responses. Six “expert” test users interacted with the program for 5 minutes each. Each of them indentified the exploit with the opponent when selecting tackle. 3 out of the 6 identified how the exploit exists while the other 3 blamed the polling of the button press. This was to see if developers are capable of finding more ways to exploit the game or can accurately find the bug.

Another test done was using the 1.0 legacy code to test readability of the code. Six various expert users were gathered to look at the code unformally (in a classroom) to look over the code as a group (roasting the authors). The group observed the code , process, and result of the program for 10 minutes. None could find the issue, but they came very close to accidentally finding it. The result of this test was nothing more of a demonstration that the code needed to be reorganized.

The greybox testing was done with average users because of better insight and responses. A total of 6 “average” test users interacted with the program for 5 minutes each. 2 of them indentified the exploit with the opponent when selecting tackle. None of them identified how the exploit exists while each of them had various answers. However, one was close to the correct answer. The reasoning was that “If I click the other attack button [element] the pikachu [opponent] attacks, but if I hit tackle, it doesn’t. It must mean that something isn’t happening when Im selecting tackle.” This was to test to see if the average user notice the bug and can accurately identify and find the exploit.

### 3.1.3 Tools

This was completed using Clion C++ IDE compiler and notepad++. The Clion C++ IDE compiler was used to test process of certain methods and blocks of code. Outputs, breakpoints, call stacks, and memory operations was observed in both sandboxes testing security and stability for integration and bug fixing. The Notepad++ was used for the white box tests to be shown along side while proceeding through the program. It was also used to look over parts of comments and classes in the sandboxes.

**3.1.4 Risk Assessment**

[See in 4.1.3](#_Risk_Assessment_&)

# Figures and Code

Diagram

Description automatically generated

Figure 3. Class Diagram.

Diagram

Description automatically generated

Figure 4. File, Stages of Program, and Folders Organization and Interaction.



Figure 5. Table of Top Potential Threats that can take place in the Source Code.

Text

Description automatically generated

Figure 6. Output of from sandbox2 which is showing the moves process

### Risk Assessment & Checklist



Figure 7. Risk Assessment Check list with Access and Network Controls.

Table

Description automatically generated

Figure 8. Overal Risk Assessment Table.

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

A picture containing timeline

Description automatically generated

The user either selects “start” or “exit.” If the user clicks start, they directed to the selection menu where they go to select a pokemon. If the user clicks exit, the program ends. The user selects one of the three pokemon, if the user does not selects and go, it will default to Charmander. Each pokemon has a type that effects the move’s attack and its own elemental move type. The user can select tackle or elemental which does the tackle move and elemental move and the item and run buttons will tell the user they cannot run or use items in the battle. The end condition is either the opponent’s or the user’s pokemon’s hp reaches 0.

### Other figure of code

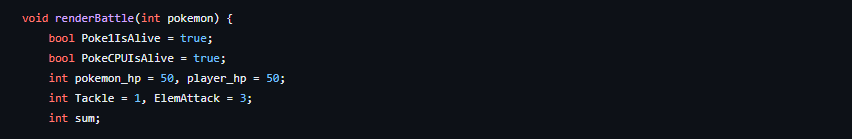
#### 0.9 Legacy Version

(it highlights how bad the lack of iterupts were before workarounds were made):

<https://github.com/Keko787/Pokemon-C-battling-system-Demo/blob/main/PokemonBlueCpp/unused%20assets/PokemonProject2.cpp>

Text

Description automatically generated



#### 1.0 Legacy Code:

<https://github.com/Keko787/Pokemon-C-battling-system-Demo/blob/main/PokemonBlueCpp/unused%20assets/pokemonProjFinal.cpp>

Text

Description automatically generated

* Missing opponent ai code

Text

Description automatically generated

* Bulk of code was in main() and unorganized

#### Latest Version:

<https://github.com/Keko787/Pokemon-C-battling-system-Demo/tree/main/PokemonBlueCpp/programs>

# Works Cited

Cplusplus.com2012*how to hide txt file through cpp(file is created by program)*https://cplusplus.com/forum/general/61322/

*Pointers*https://cplusplus.com/doc/tutorial/pointers/

2023*GitHub\_SecurityAndAccessControls*https://web.microsoftstream.com/video/f48ac5f7-a367-4881-a994-5188e4382cfe

2022*How to Create a Random Number Generator in C++*https://www.digitalocean.com/community/tutorials/random-number-generator-c-plus-plus

2023*Pokemon C Battling system Demo*Fort MyersGithub

*SFML*https://www.sfml-dev.org/tutorials/2.5/