Security Assessment Report

Paulo Drefahl

Pokemon Blue Game Security Assessment

Version N.1

May 1, 2023

Table of Contents

1. Summary

This project focused on enhancing the security of a Pokémon game coded in C using the SFML library, through a series of tests and code reviews. The overall goal of the project was to ensure that the software is secure and to identify and mitigate any vulnerabilities.

To achieve this goal, various security concepts were applied, including vulnerability scanning, risk assessments, and the use of debug tools. The method involved analyzing and improving the code based on the security concepts learned in class (CEN 3078) and consulting past documentation for the game and the frameworks involved.

Throughout the project, major findings and recommendations were made based on the tests and code reviews conducted. These findings and recommendations were aimed at improving the security of the software and mitigating any vulnerabilities that were identified.

Through this document we would explain the steps and process to enhance the security of the Pokémon game, ensuring that it is more secure and less vulnerable to potential threats. The methods and recommendations made throughout the project can be applied to other software development projects, serving as a guide for future development teams to ensure the security of their software.

1. Assessment Scope

In this project, various tools and platforms were used for testing and code reviews. The primary IDE used for coding and debugging was CLion. Additionally, TryHackMe was used as a platform to test the security of the software through vulnerability scanning and risk assessments.

Various operating systems were also used during the testing phase, including Windows and macOS, to ensure that the software could function correctly on different platforms. Different browsers were also used to test the web-based components of the software to ensure cross-browser compatibility.

In addition to the aforementioned tools and platforms, several software libraries and frameworks were also utilized, including the SFML library for game development and various security-focused libraries for vulnerability scanning and risk assessments.

2. Summary of Findings

Of the findings discovered during our assessment, 1 were considered High risks, 1 Moderate risks, 3 Low, and 0 Informational risks. The SWOT used for planning the assessment are broken down as shown in Figure 2.

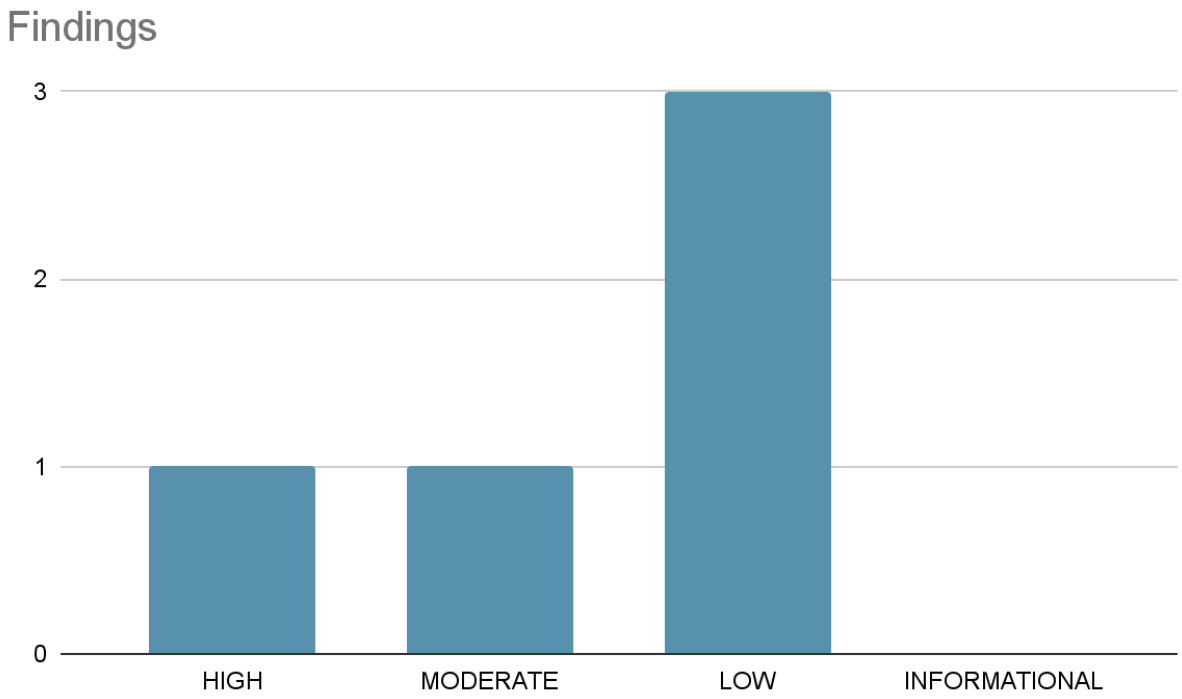


Figure 1. Findings by Risk Level

Security Vulnerability findings:

File path injection (HIGH): The source code, in its first versions, contained hardcoded file paths:

```
if (!font.loadFromFile( filename: "C:/Users/Paulo Drefahl/Desktop/hellosfml/PKMNRBYGSC.ttf")) {
    std::cout << "font error";
}
```

Hardcoded file paths can be vulnerable to file path attacks, in which an attacker can manipulate the file path to access sensitive files or execute arbitrary code. An attacker could create a file and place it in a directory that the application has access to, then use a file path injection attack to load that file instead of the intended file.

Buffer overflow (LOW): In the first versions of the software, we decided to use pointers to allocate dynamically certain features of the game and stats of the pokemons. However, in the newer versions, we are not using anymore, therefore, buffer overflow is not a security issue anymore.

Improper input validation (MODERATE): The source code did not validate user input, which could lead to unexpected behavior or vulnerabilities. A red team attacker could manipulate the mouse events to select a non-existent Pokemon or trigger unexpected behavior.

```
mouse.pos2X = event1.mouseButton.x;
mouse.pos2Y = event1.mouseButton.y;
```

Inadequate Backing Up of Files (LOW): One security vulnerability identified in the Pokemon project is that the computer that stores the code locally was not properly set up for backups within the cloud. This means that in the event of a hardware failure or other issue, the code could be lost permanently. It is important to have a reliable backup system in place to ensure that data is not lost, and this issue should be addressed as soon as possible to prevent potential data loss.

Improper Access Control in GitHub (LOW): In the Pokemon project, the access control for the GitHub repository was not configured correctly, resulting in two developers being unable to access it appropriately. Additionally, some confidential files other than the source code were visible to unauthorized individuals due to the incorrect access control settings.

Figure 2: SWOT

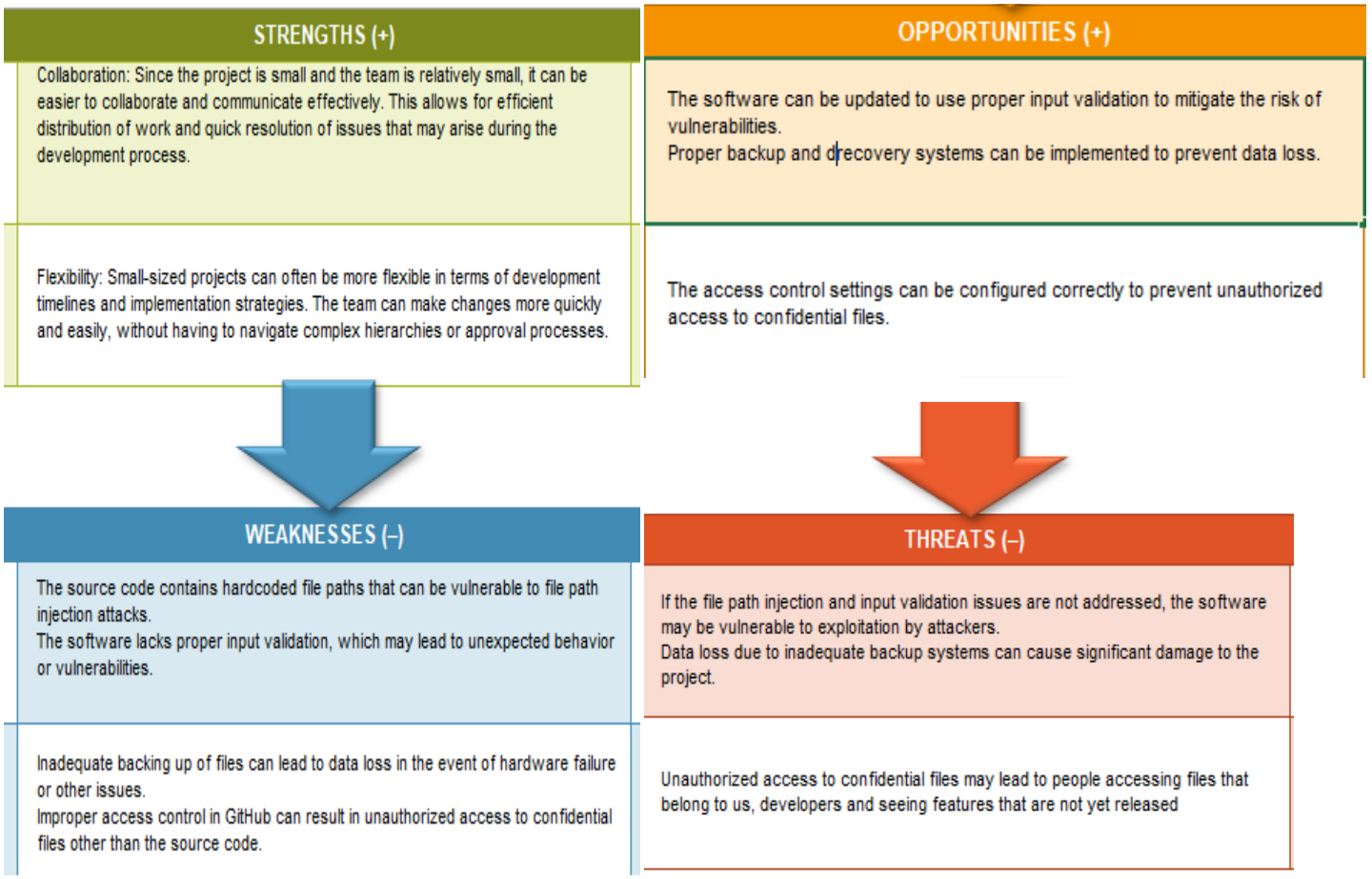


Figure 2. SWOT

All the issues mentioned in the assessment are pointed in this final version of SWOT:

File path injection (HIGH), Buffer overflow (LOW), Improper input validation (MODERATE), Inadequate Backing Up of Files (LOW), Improper Access Control in GitHub (LOW)

3. Summary of Recommendations

Changes have been made to address the security vulnerabilities identified in the project, such as removing hardcoded file paths, validating user input, and setting up proper backups. The access control issue on GitHub has also been addressed in the past security iterations. However, ongoing efforts are needed to maintain the security of the project, such as regular vulnerability scanning and addressing any new issues based on the comments by Prof Greenwell.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assignment was to get a project that we worked in the past and implement a series of new additions to make the program more secure against many types of threats.

- Examine if the system followed the appropriate norms and standards, including the assignments themselves.
- Make sure that the project was well stored in different places, so in case of losing or damaging the computer, we would be able to get the code again.
- Ensure that only allowed people have access to code and can edit.

2. Detailed Findings

The first step we took was to update the backup setting in all the machines there were being utilized. Then we included the files of the game in each Onedrive and we took one extra step by uploading the project to both GitHub accounts. Then, we integrated the IDE we were using (CLION) with GitHub to manage version control and pull requests.
We enabled the vulnerability report in the file, managed the access controls, and uploaded a security policy with our information at the end so that email can be sent to us in case of bugs were found. Then we had to make sure the code was able to validate input, ensure the security of file paths, and delete uninitialized variables. After that, we checked SFML for possible updates

and if it was viable to do so, there is not a more recent stable version, so we did not have to change anything but we are constantly checking for when it happens we fix everything that needs to be changed.

3. Recommendations

File path injection (HIGH): We avoid file path injection, using hardcoded file paths. Searching for references we could notice that to refer to files, you can instead use relative paths or environment variables. We also tried to verify and sanitize user input to prevent malicious input from being performed, modifying the code in this situation to remove any hardcoded file paths and replace them with more secure alternatives.

Buffer overflow (LOW): Although this vulnerability is no longer a security threat in recent versions of the program, we should still make code review to remove any instances of pointer use that may lead to buffer overflow vulnerabilities. This will help us in the prevention of any potential security concerns that we may face in the future.

Improper input validation (MODERATE): We included suitable input validation and checked to verify that user input is legitimate and does not constitute a security risk to mitigate improper input validation. To avoid SQL injection and other sorts of injection attacks, we also considered adding input filtering or parameterized queries.

Inadequate File Backup (LOW): To resolve this issue, we decided to establish a dependable backup strategy for the code and any other essential data. This was accomplished by utilizing cloud storage options or by establishing frequent backups to an external disk. We also used version control tools such as Git to manage and monitor changes to the code over time.

Improper GitHub Access Control (LOW): To remedy this issue, we verified the GitHub repository's access control settings and ensured that only authorized persons have access to sensitive files and data. To better safeguard the repository, we are using two-factor authentication or other security features. Furthermore, we reviewed any potentially exposed confidential files and took appropriate steps to mitigate any potential harm caused by the exposure.

3. Methodology for the Security Control Assessment

3.1.1 Risk Level Assessment

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 Table 1 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 • 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

By definition penetration testing is a process of testing a system's security by simulating attacks on it. In our project, we performed a superficial penetration testing to identify vulnerabilities in the system.

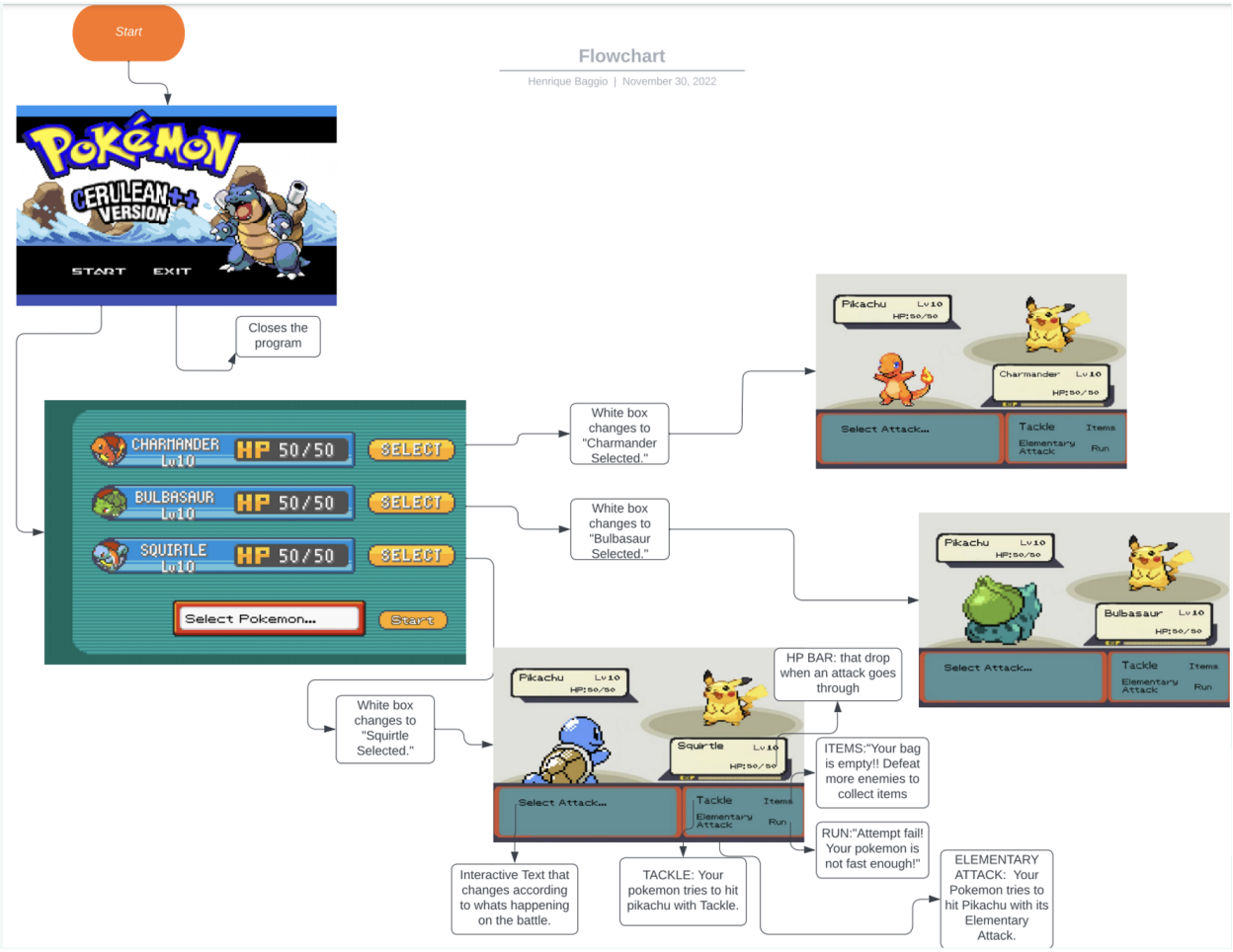
A lot of WhiteBox testing was used. The goal of white box testing is to validate thesoftware's functioning as well as its internal components and algorithms. We used a lot of White Box testing in our code mainly to test the adequate pixels in the images in order to make “buttons” and many other things.

3.1.3 Tools

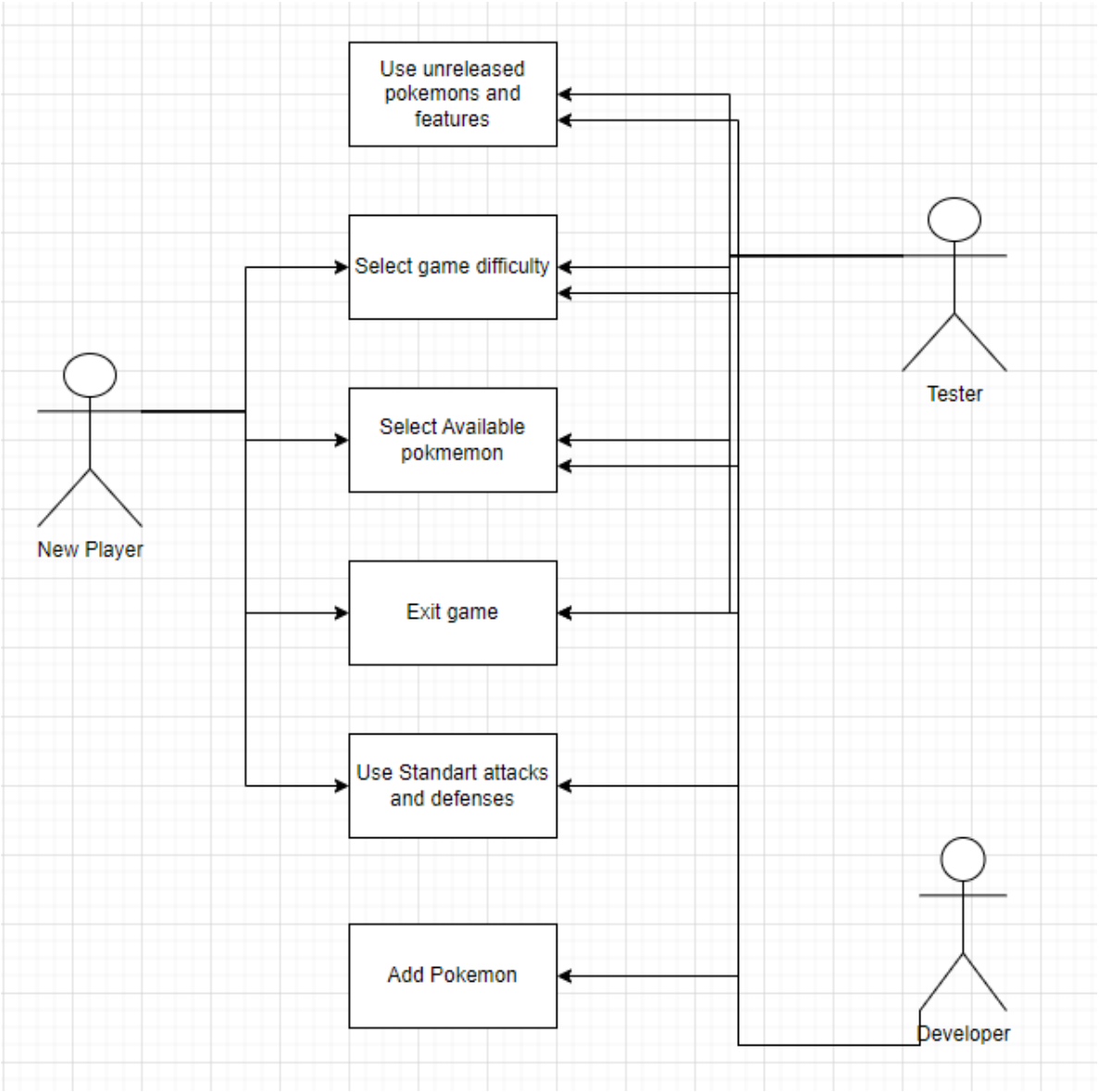
The tools used directly in your project was the Clion debugger however, indirectly we studied tools like OWASP ZAP and websites like try Hack me to gather enough knowledge to apply to our tests for this assessment.

4. Figures and Code

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



Description: This program opens with a home screen to the game (Cerulean++), when we press play we are prompted with 3 Pokemons to pick. The options are Charmander, Bulbasaur, and Squirtle. All of the Pokemons have the same hp and level, the only thing that will change is the design and if the elementary attack is effective against Pikachu or not, and if Pikachu is effective against your Pokemon. When you select your Pokemon you start the battle and you can choose between a tackle and an elementary attack. Every time you hit Pikachu he will hit back on you until one of the Pokemon's HP is 0. If that happens, the Pokemon that is still alive is the winner.



The new player actor can start the game and select a pokemon,. a difficulty, use the normal features and of course, exit the game. The tester is the one who can also use and test the unreleased features and the developer can basically use all the features and add new ones, such as pokemons, attacks moves, etc.

4.1.2 Other figure of code

```
if (!font.loadFromFile( filename: "C:/Users/Paulo Drefahl/Desktop/backup files/hellosfml/PKMNRBYGSC.ttf")) {
    std::cout << "font error";
}
```

```
pokemonSelectionImg.loadFromFile( filename: "C:/Users/Paulo Drefahl/Desktop/backup files/hellosfml/Poke_Menu_WIP.png");
mainMenuImg.loadFromFile( filename: "");
```

```
switch (pokemon)
{
    case 0:
        BattleBackground.loadFromFile( filename: "C:/Users/kskos/CLionProjects/PokemonBlueCpp/char_battle_1frame.png");
        break;
    case 1:
        BattleBackground.loadFromFile( filename: "C:/Users/kskos/CLionProjects/PokemonBlueCpp/bulba_battle_1frame.png");
        break;
    case 2:
        BattleBackground.loadFromFile( filename: "C:/Users/kskos/CLionProjects/PokemonBlueCpp/squir_battle_1frame.png");
        break;
}
```

The images above are from sections of code that could lead to vulnerabilities explained previously in this document.

```
395             }
396         }
397         renderBattleScreen.clear();
398         renderBattleScreen.draw( drawable: sprite3);
399         renderBattleScreen.draw( drawable: hp_player);
400         renderBattleScreen.draw( drawable: hp_player_const);
401         renderBattleScreen.draw( drawable: hp_pokemon);
402         renderBattleScreen.draw( drawable: hp_pokemon_const);
403         renderBattleScreen.draw( drawable: text2);
404         renderBattleScreen.display();
405
406     }
407 }
```

This image shows another problem that could lead to possible vulnerabilities and maintenance problems. The code is concentrated in only one file. Therefore, if any problem happens with a version of the file the whole application is affected. In addition. it is important to emphasize the organizational problems that it could cause as well.

5. Works Cited

TryHack Me: https://tryhackme.com/dashboard
NIST Cybersecurity Framework: *https://www.nist.gov/cyberframework*
OWASP (Open Web Application Security Project)*:* *https://owasp.org/*
SANS Institute: *https://www.sans.org/*