**Logo, company name

Description automatically generated**

COP 3003 Group Project Tetris Program

Khang Phan

COP 3078

Professor Greenwell

**Security Assessment Report**

Version N.1

May 1st, 2023

Table of Contents

[1. Summary 3](#_Toc127779445)

[1. Assessment Scope 3](#_Toc127779446)

[2. Summary of Findings 3](#_Toc127779447)

[3. Summary of Recommendations 4](#_Toc127779448)

[2. Goals, Findings, and Recommendations 4](#_Toc127779449)

[1. Assessment Goals 4](#_Toc127779450)

[2. Detailed Findings 5](#_Toc127779451)

[3. Recommendations 5](#_Toc127779452)

[3. Methodology for the Security Control Assessment 5](#_Toc127779453)

[4. Figures and Code 7](#_Toc127779454)

[4.1.1 Process flow of System (this one just describes the process for requesting) 7](#_Toc127779455)

[4.1.2 Other figure of code 7](#_Toc127779456)

[5. Works Cited 7](#_Toc127779457)

# Summary

Executive Summary Here: Describe the overall goal, method, and major findings/recommendations here. (it’s the TLDR)

## Assessment Scope

What tools, platforms, OSes, Browsers, and software (including your own) was tested or used in testing?

* The provided Tetris Program was written C++ using Visual Studio 2019, GCC, and CLion.
* The developers used debuggers like GDB and Valgrind to fix bugs and errors.
* Nessus was used to test software penetration.
* The code was written with Windows 11 and the lastest MAC OS.
* The code was tested on Windows, Mac, and a linux virtual machine.
* The code was tested on several versions of Windows.
* The team and I used Google Chrome as the default browser.
* Implemented the SFML library to manage graphics.
* Implemented the standard library.
* Use OpenSSL and GnuPG to ensure security.

## Summary of Findings

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

Figure 1. Findings by Risk Level

Explain above and link to full table of explanation of top risks like Figure 3.

* The tops risks that were mentioned in the table that have “High Priority” is that the program file is stored on inadequate hardware system that would potentially expose the program vulnerability.
* Testing was done minimally since the team has to submit it before a deadline, which results in minimal and basic testing methods.

Graphical user interface, table, calendar

Description automatically generated with medium confidence

Diagram

Description automatically generated

Figure 2. SWOT

Explain which issues were used from the above SWOT (which are addressed in this assessment).

* Major issues were coding practices like using global variables and including namespace std, Shapes spawning outside the border since smart pointers wasn’t implemented. Key injection is also one of the potential issues mentioned in the SWOT.

## Summary of Recommendations

* Several issues were fixed like creating a backup repository for the code, installing the latest version of SMFL, implement a system version control, remove namespace std, declare shape classes, implement smart pointer.
* Several issues that still need to be addressed are: two-factor authentication during access, implement encryption methods, user input error handle, missing local version control system.

# Goals, Findings, and Recommendations

## Assessment Goals

The purpose of this assessment was to do the following:

* Ensure that the system was in compliance with regulations you had to deal with or any other requirements (to include the assignments themselves).
* Determine if the application was secured.
* Practice secured coding practices.
* Implementing and introducing new security measures.
* To comply with different regulations and compliance.
* To save time and resources.
* Little to no errors left to analyze and fix.
* To identify vulnerabilities of the system and offers security solutions to fix those problems.
* Manageable “crisis recovery” when the system fails.

## Detailed Findings

Refer to Figure 1 of Section 4:

* Issue 6 involves implementing a third-party library (SFML). If it’s not up-to-date, it can create security risks within the code due to the old security patches.
* Issue 5 covers a back-up policy being implemented for the code. Data backup reduces the change of data theft and loss.
* Issue 7 mentions that the code is being store in an adequate hardware system to ensure the most secured and up-to-date security patches. Also prevents data loss and theft.
* Issue 14 covers safe coding practices to retain code tracibility and maintainability.
* Issue 6 covers the risk of not having a user access control system, which can give unwanted individual from full access. Which can lead to unwanted alterations and loss.
* Issue 12 covers encryptions for the Tetris gane code. The consequence can be similar to one of issue 6.
* Issue 14 covers the Security testing. It makes sure that your program is safe and secure when running in any environment.
* Issue 10 covers Internal Actors threat. A threat that was done from the inside to sabotage the project or system.

## Recommendations

Refer to the Figure 1 Table in section 4:

* The recommendations for the threat solutions are under the coluum named “Full Description of processes to address issue”, that correspond to valid risks mentioned above.

# 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 - 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 - 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**

- Unit testing was used to ensure every fragment of code is working properly individually and working together as a unit. A standard IDE like CLion was used for this process.

- Integration testing was used to ensure every fragment of code is working properly together as a coherent unit. A standard IDE like CLion was used for this process.

- Penetration Testing was used to identify further threats to identify the penetration vulnerabilities. I used Nessus for this process.

- Compatibility Testing was performed to make sure the system is able to run successfully under different conditions like different Operating systems, different versions of the same OS, different versions of SFML library, Clion version, and C++ versions to see if it would still run efficiently and securely.

**3.1.3 Tools**

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

* The provided Tetris Program was written C++ using Visual Studio 2019, GCC, and CLion.
* The developers used debuggers like GDB and Valgrind to fix bugs and errors.
* Nessus was used to test software penetration. To update and install Nessus through the Linux command line we type:

sudo apt-get update

sudo apt-get install nessus

* The code was written with Windows 11 and the lastest MAC OS.
* The code was tested on Windows, Mac, and a linux virtual machine.
* The code was tested on several versions of Windows.
* The team and I used Google Chrome as the default browser.
* Implemented the SFML library to manage graphics. To update and install OpenSSL through the Linux command line we type:

sudo apt-get update

sudo apt-get install libsfml-dev

* Implemented the standard STD library.
* Use OpenSSL and GnuPG to ensure security. To update and install OpenSSL through the Linux command line we type:

sudo apt-get update

sudo apt-get install openssl

* To update and install GnuPG through the Linux command line we type:

sudo apt-get update

sudo apt-get install gnupg

# Figures and Code

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

Diagram

Description automatically generated

Figure 3. Work flow of System

Describe the process flow here.

### Other figure of code

Lastest SFML must be installed.



Remove namespace std.

A screenshot of a computer

Description automatically generated with medium confidence

Make functions and called them in main instead of doing everything in main

A screenshot of a computer

Description automatically generated with medium confidence

Implement smart pointers.

A screenshot of a computer

Description automatically generated with medium confidence

Declare several shape classes.

Text

Description automatically generated

Link to the full code:

<https://github.com/phanandco/security-project-code/blob/main/main-tetris-code-cpp>

# Works Cited

Phanandco. "Security Project Code." GitHub, 2023, <https://github.com/phanandco/security-project-code>.

"Why is using namespace std considered bad practice?" Stack Overflow, 25 Aug. 2009, <https://stackoverflow.com/questions/1452721/why-is-using-namespace-std-considered-bad-practice.\>

Security Checklist SpreadSheet: <https://docs.google.com/spreadsheets/d/12dC0FqScO7Xtkuq0cuozdKKIPgVMsj-7/edit#gid=1050731267>

"What is a smart pointer and when should I use one?" Stack Overflow, Stack Exchange Inc., 3 Oct. 2008, <https://stackoverflow.com/questions/106508/what-is-a-smart-pointer-and-when-should-i-use-one>.