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

Security Assessment Report

Version N.0

January 1, 1900

Table of Contents

1. Summary

The overall goal of this project was to determine the security strengths and weaknesses of ShapesProject.cpp. Testing was performed to determine what forms of input could be accepted by the program and what weaknesses it contained that could pose a threat to the security of its developers or the health of any device running it.

1. Assessment Scope

All testing on the code found at https://github.com/WMDragstrem/CEN3078-Computer-Security-Project was performed on a machine with Windows 10. Testing and debugging was performed in the Jetbrains CLion IDE. Test scope was limited due to lack of access to alternate OSes for testing (such as Apple).

2. Summary of Findings

Of the findings discovered during our assessment, 3 were considered High risks, 4 Moderate risks, 3 Low, and 0 Informational risks. The SWOT used for planning the assessment are broken down as shown in Figure 2.

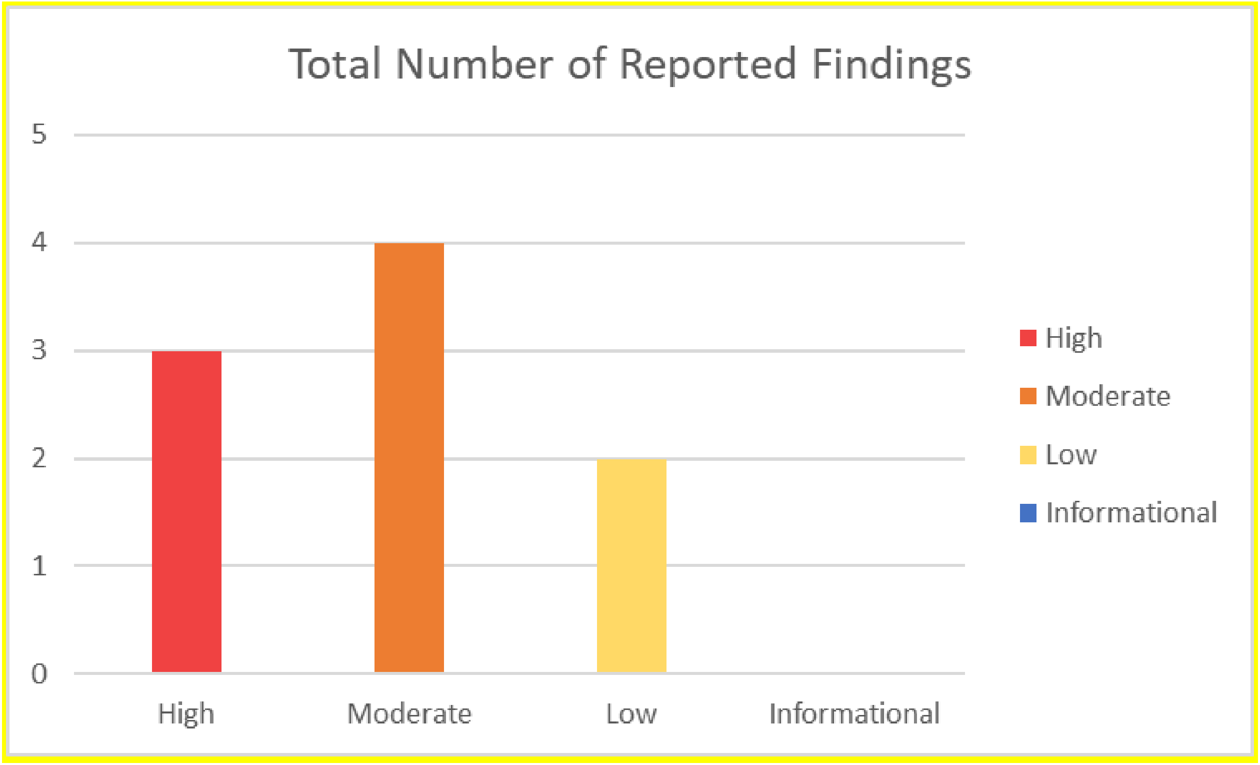
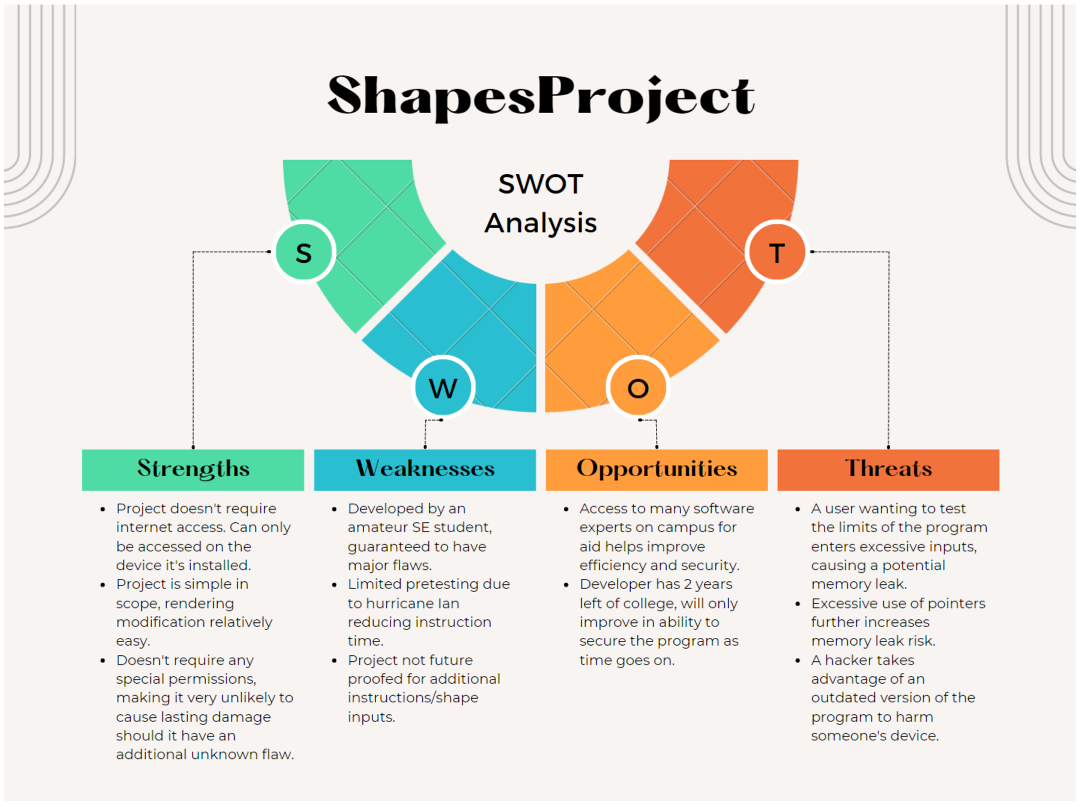


Figure 1. Findings by Risk Level

Security Assessment – ShapesProject.cpp

The 3 major risks were risk of memory leak, lack of a project backup, and the lack of an online terms of use. Memory leak is a severe issue that can cause damage and completely stop a system from functioning should it occur, the risk is due to improper usage of pointers and no limits placed on the total number of iterations. No project backup or prior version documentation can severely impact code maintenance and makes it completely impossible to roll back the code should a mistake be made in a future update. And a terms of use is immediately necessary for any publicly released code to prevent liability from improper use of what is currently flawed code.



Strengths

- Project doesn't require internet access. Can only be accessed on the device it's installed.
- Project is simple in scope, rendering modification relatively easy.
- Doesn't require any special permissions, making it very unlikely to cause lasting damage should it have an additional unknown flaw.

Weaknesses

- Developed by an amateur SE student, guaranteed to have major flaws.
- Limited pretesting due to hurricane Ian reducing instruction time.
- Project not future proofed for additional instructions/shape inputs.

Opportunities

- Access to many software experts on campus for aid helps improve efficiency and security.
- Developer has 2 years left of college, will only improve in ability to secure the program as time goes on.

Threats

- A user wanting to test the limits of the program enters excessive inputs, causing a potential memory leak.
- Excessive use of pointers further increases memory leak risk.
- A hacker takes advantage of an outdated version of the program to harm someone's device.

Figure 2. SWOT

3. Summary of Recommendations

Currently 2 of the 3 major threats listed have been accounted for. A terms of use has been written and the code now utilizes enums and only accepts integers as input, preventing the risk of dependency injection. Memory leak risk has been reduced by limiting the total number of iterations through which the code is allowed to run, but rewriting the current utilization of pointers is still a work-in-progress, luckily the scope is not so grand that an entire rewrite is required. Testing on different OSes is still needed, and further penetration testing is required to ensure that dependency injection isn't possible. Many existing documentation issues were solved through utilization of Github as an online repository.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to do the following:

- Ensure that the program follows proper SDL methodology.
- Determine if the application was securely maintained.
- Determine if the application and its updates were properly documented.
- Eliminate possible threats caused by flaws within the program that would expose security vulnerabilities or damage the hardware running it.

2. Detailed Findings

Finding	Threat Priority	Solved?
1) Memory leak caused by returning pointers	High	N
2) Dependency injection possible	Moderate	N
3) Memory leak by continuous program looping	Moderate	Y
4) Program crashes if it receives an incorrect input type.	Moderate	Y
5) Program untested in non-windows OS	Low	N
6) No terms of use	High	Y
7) No form of access-control for the project	Low	Y
8) No project backup and documentation	High	Y
9) Program not future proofed for shapes/functions that start with the same letter.	Moderate	Y

3. Recommendations

Currently, the unsolved threats should be solved as soon as possible in order of highest to lowest threat priority.

1) Pointers should be either completely eliminated, or replaced with smart pointers so that they don't build up and clog the memory from being returned in the programs functions.
2) Further penetration testing is required but utilization of prepared statements is a necessary and immediate step in resolving this issue.
3) Program has had a hard cap placed on the amount of shapes allowed to be stored within memory (currently 256). Once the limit is reached it outputs an error whenever you request to input an additional shape. Utilize valgrind for additional testing.
4) Input has been restricted to integers only. Use cin.clear() and cin.ignore() to test and prevent any crashes for non-integer input.
5) Either acquire an apple device or an apple OS equipped virtual machine with which to test the program.
6) Write the terms of use ASAP to prevent any form of liability from flawed code and any damage it could potentially cause.
7) Utilize Github as an online repository to save prior versions of code and permit access control.
8) Make use of Github's information hiding provisions to conceal sensitive information and code.
9) Utilize enums and accept integers as input to select which shape / function the user wishes to perform.

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 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
Informationa l	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.

Rating	Definition of Risk Rating
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

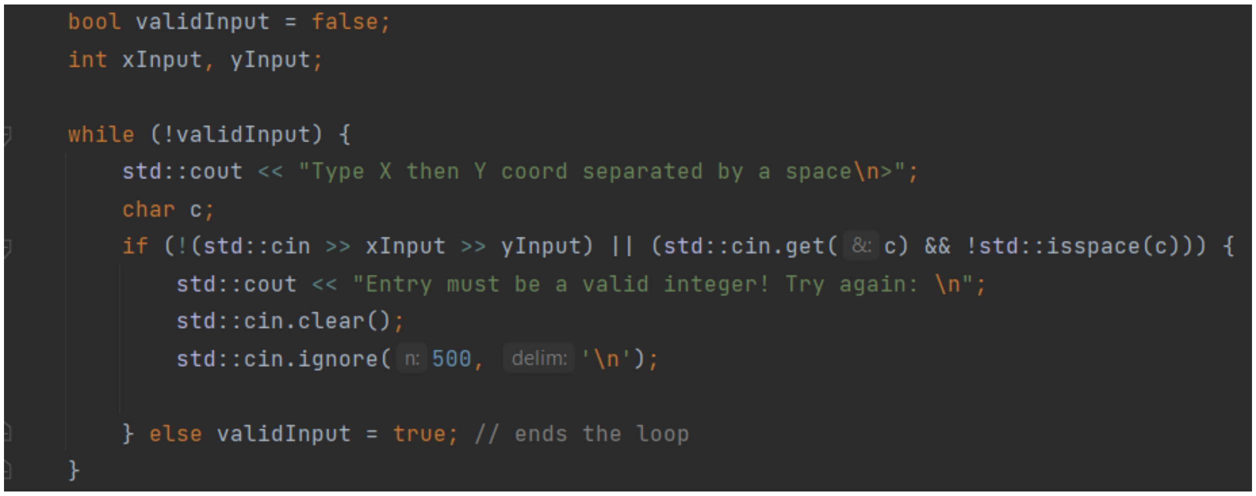
3.1.2 Tests and Analyses

Testing was completed using both Black Box and White box testing techniques. For tblack box, volunteers were asked to utilize the code as they normally would and see if there was any confusion as to the accepted inputs, then they were requested to attempt to "break" the program through purposely incorrect inputs. White box testing was performed by the project's sole-programmer in which he would deliberately input data that was not formatted as intended for the program and its functions.

3.1.3 Tools

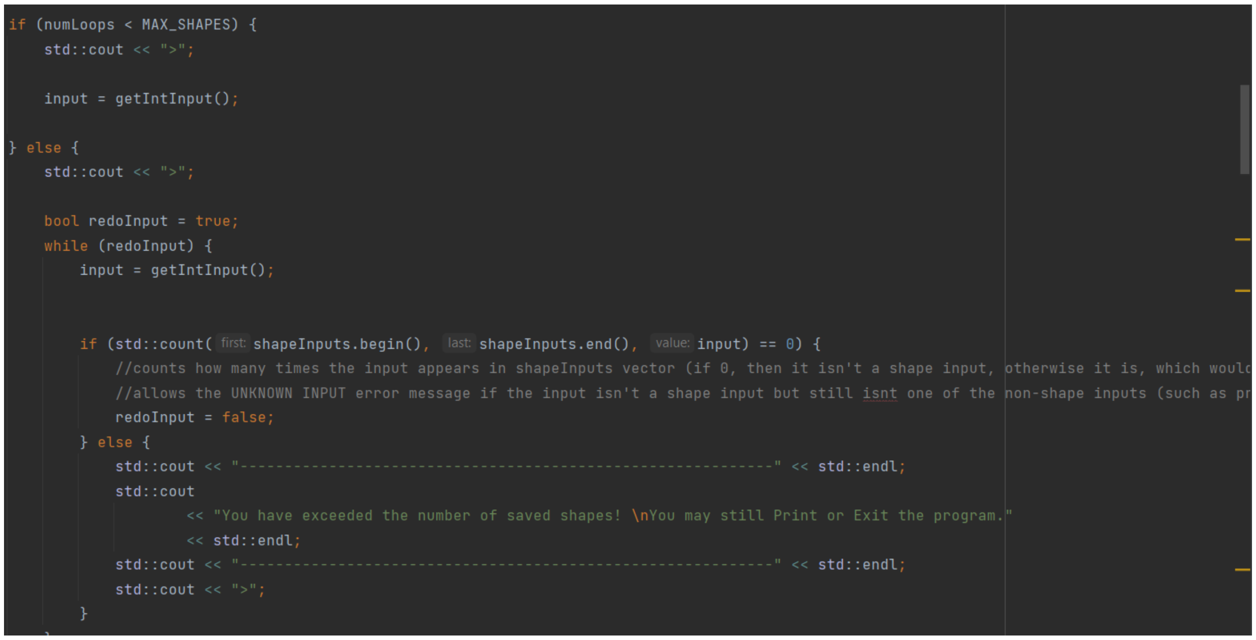
This was completed using Jetbrains CLion IDE and its built-in debugging utility.

4. Figures and Code



```cpp
bool validInput = false;
int xInput, yInput;

while (!validInput) {
    std::cout << "Type X then Y coord separated by a space\n>";
    char c;
    if (!(std::cin >> xInput >> yInput) || (std::cin.get(c) && !std::isspace(c))) {
        std::cout << "Entry must be a valid integer! Try again: \n";
        std::cin.clear();
        std::cin.ignore(500, '\n');

    } else validInput = true; // ends the loop
}
```

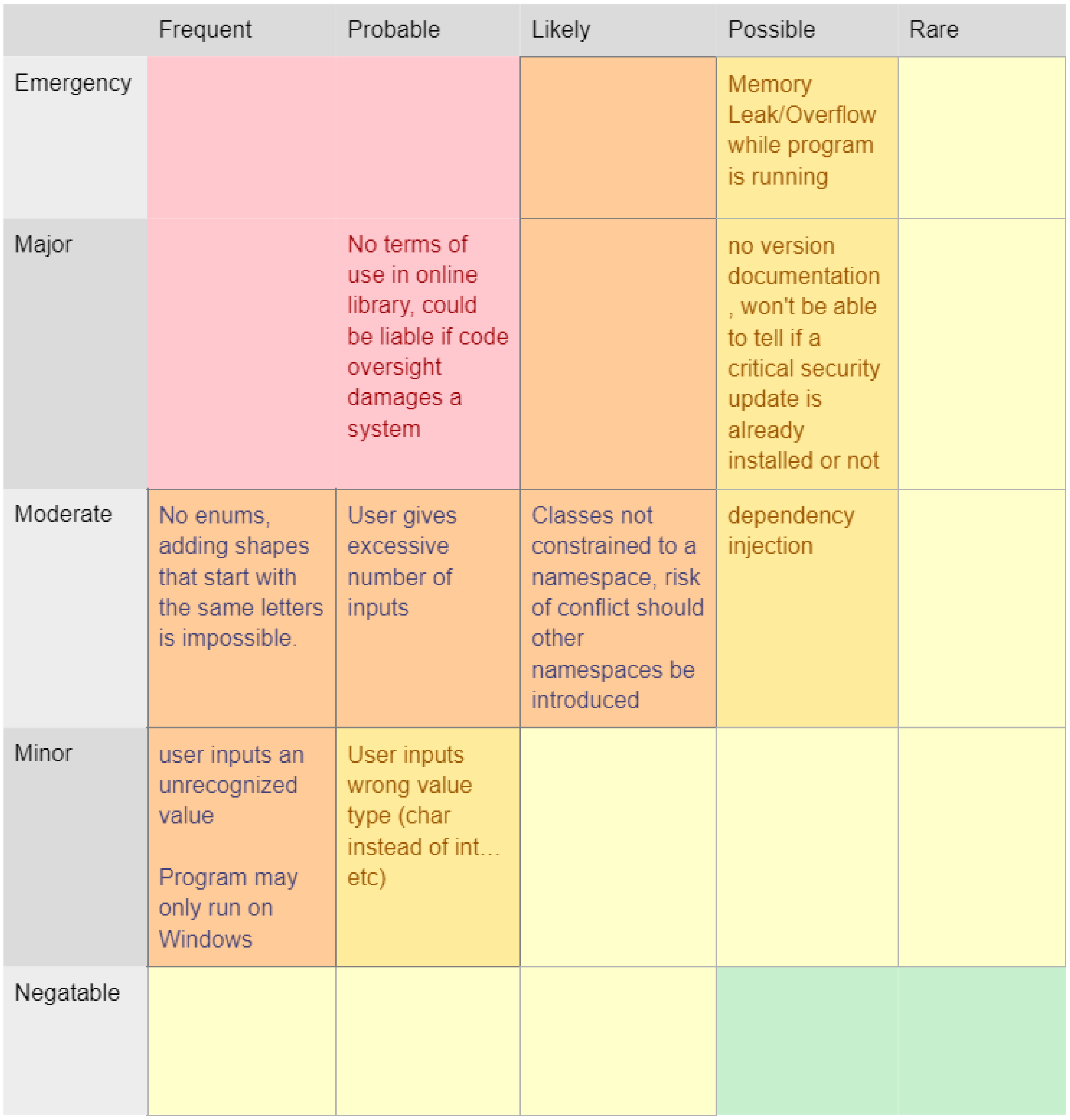
An example of the current code used to limit input to integers only (based on https://cplusplus.com/forum/beginner/123439/) This code will be updated based on the information found at https://www.codespeedy.com/taking-only-integer-input-in-cpp/



```cpp
if (numLoops < MAX_SHAPES) {
    std::cout << ">";

    input = getIntInput();

} else {
    std::cout << ">";

    bool redoInput = true;
    while (redoInput) {
        input = getIntInput();


        if (std::count(shapeInputs.begin(), shapeInputs.end(), input) == 0) {
            //counts how many times the input appears in shapeInputs vector (if 0, then it isn't a shape input, otherwise it is, which woul
            //allows the UNKNOWN INPUT error message if the input isn't a shape input but still isnt one of the non-shape inputs (such as pr
            redoInput = false;
        } else {
            std::cout << "------------------------------------------------------------" << std::endl;
            std::cout
                    << "You have exceeded the number of saved shapes! \nYou may still Print or Exit the program."
                    << std::endl;
            std::cout << "------------------------------------------------------------" << std::endl;
            std::cout << ">";
        }
```

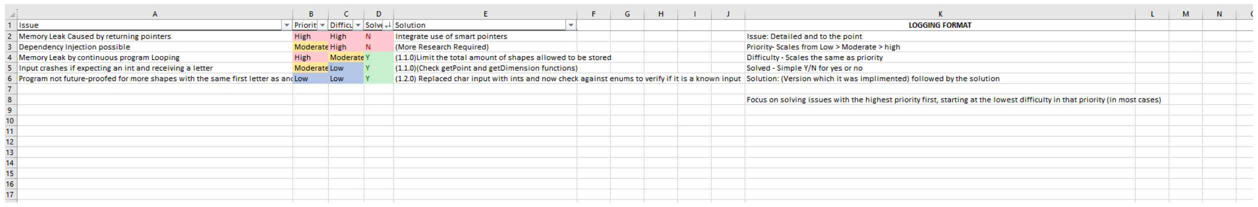
Current code used to limit the total number of saved shapes to prevent memory leaks. Once the number exceeds the const value for MAX_SHAPES, an error message is output that asks for a different input instead.

Security Assessment – ShapesProject.cpp



	Frequent	Probable	Likely	Possible	Rare
Emergency				Memory Leak/Overflow while program is running	
Major		No terms of use in online library, could be liable if code oversight damages a system		no version documentation , won't be able to tell if a critical security update is already installed or not	
Moderate	No enums, adding shapes that start with the same letters is impossible.	User gives excessive number of inputs	Classes not constrained to a namespace, risk of conflict should other namespaces be introduced	dependency injection	
Minor	user inputs an unrecognized value Program may only run on Windows	User inputs wrong value type (char instead of int… etc)			
Negatable					

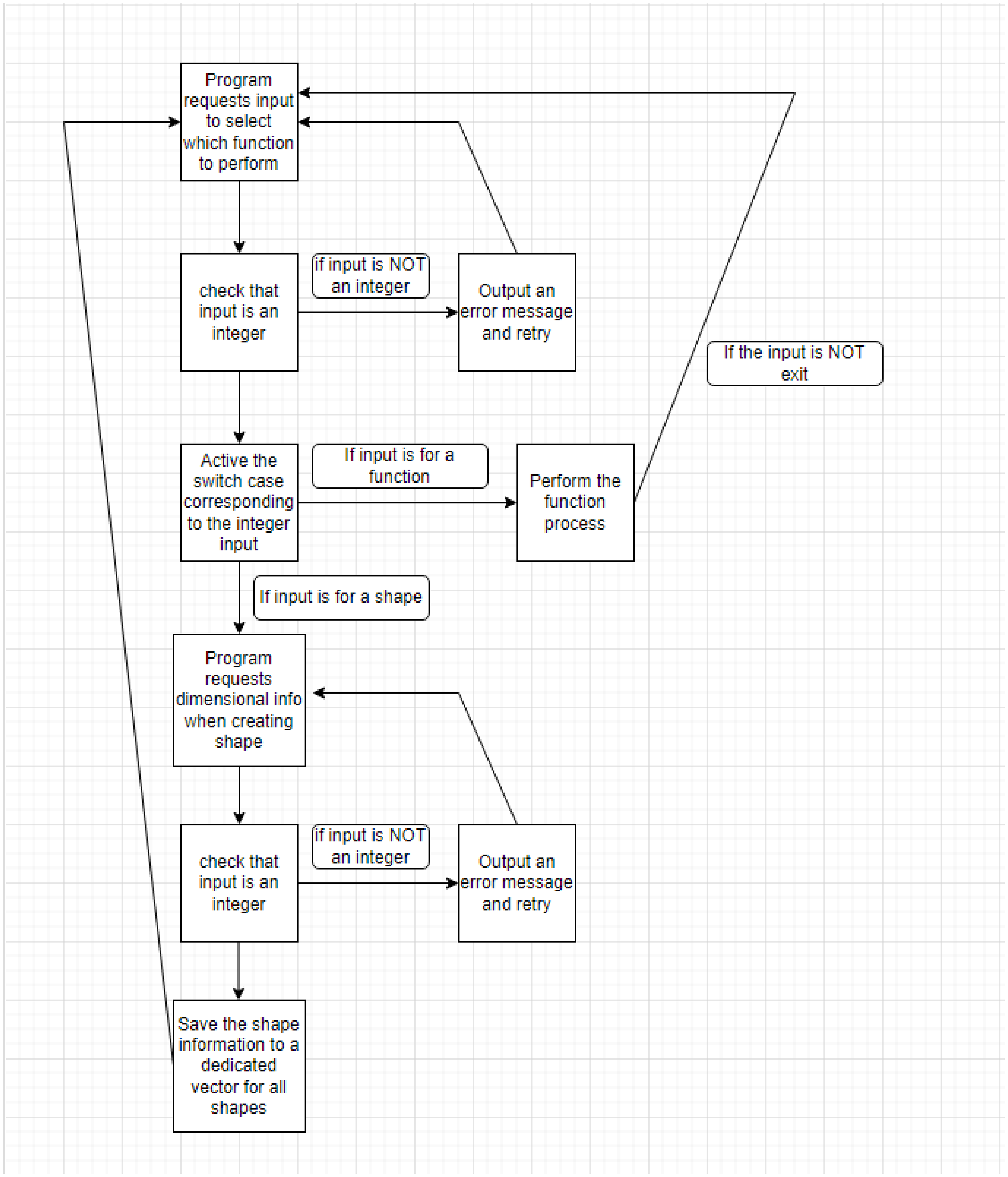
Risk Assessment Matrix for the program



	Issue	Priorit	Difficu	Solve	Solution	LOGGING FORMAT
2	Memory Leak Caused by returning pointers	High	High	N	Integrate use of smart pointers	Issue: Detailed and to the point
3	Dependency Injection possible	Moderate	High	N	(More Research Required)	Priority- Scales from Low > Moderate > high
4	Memory Leak by continuous program Looping	High	Moderate	Y	(1.1.0)Limit the total amount of shapes allowed to be stored	Difficulty - Scales the same as priority
5	Input crashes if expecting an int and receiving a letter	Moderate	Low	Y	(1.1.0)(Check getPoint and getDimension functions)	Solved - Simple Y/N for yes or no
6	Program not future-proofed for more shapes with the same first letter as anc	Low	Low	Y	(1.2.0) Replaced char input with ints and now check against enums to verify if it is a known input	Solution: (Version which it was implimented) followed by the solution
7						
8						Focus on solving issues with the highest priority first, starting at the lowest difficulty in that priority (in most cases)

Updated Methodology for tracking and logging existing/solved issues in excel.

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


Program requests input to select which function to perform
check that input is an integer
if input is NOT an integer
Output an error message and retry
If the input is NOT exit
Active the switch case corresponding to the integer input
If input is for a function
Perform the function process
If input is for a shape
Program requests dimensional info when creating shape
check that input is an integer
if input is NOT an integer
Output an error message and retry
Save the shape information to a dedicated vector for all shapes

The graph is rather self explanatory. Currently, the only 2 non-shape inputs are to print or exit. Exit simply terminates the program and print goes through the dedicated vector for all shapes and outputs the organized information for each shape based on a dedicated function within the shapes header file. (Ex: printing a circle outputs the shape type, its radius and coordinates (inputted) and its diameter/circumference/area (calculated))

5. Works Cited

"Access Permissions on GitHub." *GitHub Docs*, docs.github.com/en/get-started/learning-about-github/access-permissions-on-github.

Dalal, Animisha. "How to Take Only Integer Input in C++." *CodeSpeedy*, 28 Sept. 2019, www.codespeedy.com/taking-only-integer-input-in-cpp/.

Limit Input to Only Integer - C++ Forum, cplusplus.com/forum/beginner/123439/.

TylerMSFT. "Smart Pointers (Modern C++)." *Microsoft Learn*, learn.microsoft.com/en-us/cpp/cpp/smart-pointers-modern-cpp?view=msvc-170.

Valgrind Home, valgrind.org/.