

5. Perform bounds checking: Perform bounds checking to avoid off-by-one errors and prevent unexpected behavior

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to do the following:

The goal of a SWOT assessment is to evaluate the internal and external factors that can impact the success of a project, organization, or individual. The goal of the assessment is to provide insights into the current situation, identify potential risks and opportunities, and make informed decisions based on the findings.

2. Detailed Findings

Weaknesses:

1. Efficiency: The code may not be as efficient as it could be, particularly when dealing with large input datasets. This could lead to slower processing times and potentially limit the usefulness of the code in certain scenarios. Additionally, if the code is used frequently or with large datasets, it could place a strain on the resources of the local machine it is running on.
2. Input data vulnerabilities: The code may be vulnerable to errors if input data is not in the expected format. This could lead to unexpected results or crashes, potentially causing frustration for users or even data loss. Furthermore, the code does not currently handle unexpected or erroneous input data particularly gracefully, which could exacerbate these issues.
3. Limited extensibility: The code may not be easily extensible to new or different use cases. This could limit the usefulness of the code in the long term and potentially require significant refactoring or redevelopment to accommodate new requirements.

Threats:

1. Competition: Competing code libraries or solutions could become available and reduce the relevance or usefulness of this code. This could lead to decreased adoption and potentially impact the sustainability of the project.
2. Changing requirements: Changes to the input data format or requirements could render the code obsolete. This could require significant redevelopment or render the code completely unusable for certain scenarios.
3. Security vulnerabilities: Security vulnerabilities in the code could be exploited by attackers. This could potentially result in data breaches or other security incidents, damaging the reputation of the code and potentially resulting in legal or financial consequences.

3. Recommendations

1. Efficiency: Optimizing the code for larger datasets, potentially through the use of more efficient data structures or algorithms. Additionally, providing guidance or recommendations for users on how to optimize their input data could help mitigate this issue.
2. Input data vulnerabilities: Implementing more robust error handling and input validation to catch unexpected data and prevent crashes. Additionally, providing clearer error messages and guidance for users could help mitigate these issues.
3. Limited extensibility: Refactoring the code to make it more modular and easily extensible, potentially through the use of design patterns or other best practices. Additionally, actively seeking user feedback and incorporating feature requests could help ensure the code remains relevant and useful over time.
4. Competition: Continuously monitoring the landscape for competing solutions and actively incorporating feedback and improvements from users could help ensure the code remains competitive and relevant.
5. Changing requirements: Staying up-to-date on changes to data formats and requirements, and proactively updating the code to accommodate these changes could help ensure the code remains useful and relevant over time.
6. Security vulnerabilities: Conducting regular security audits and implementing best practices for secure coding and data handling could help prevent security incidents and maintain user trust in the code. Additionally, providing clear guidance for users on how to securely handle and store their data could help mitigate this issue.

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. 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

Security Assessment – C++ Caluclator

Rating	Definition of Risk Rating
	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 <ul style="list-style-type: none">• 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 <ul style="list-style-type: none">• 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: <ul style="list-style-type: none">• Discontinue use of the software or protocol• Isolate the information system within the enterprise, thereby eliminating reliance on the system <p>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</p>

3.1.2 Tests and Analyses

Tests:

1. Penetration testing
2. Vulnerability scanning

Analyses:

1. Impact analysis
2. Risk probability and impact assessment
3. Business impact analysis

3.1.3 Tools

1. Spreadsheet software
2. Web browser
3. Command line
4. Debugger

Security Assessment – C++ Caluclator

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.

Security Assessment – C++ Caluclator

```
1 // File : main.cpp
2 // Class: COP 3003, Fall 2022
3 // Devl : Lucas Wilkerson
4 // Desc : Final Project, Shapes Calculator
5 //-----
6
7
8 // What was changed for Computer Security:
9
10 // 1. There is no class definition for the Line, Rectangle, and Circle classes,
11 // so it's impossible to determine what methods and variables they contain.
12
13 // 2. The PointMerge class has a getMerge method that attempts to concatenate an int and a string,
14 // which is not valid. This would result in a compilation error.
15
16 // 3. In the getMerge method, the int values X and Y are added using the + operator,
17 // which is not valid for concatenating ints in C++. Instead, you should convert them to strings first using std::to_string.
18
19 // 4. The do-while loop in the main function never exits unless the user selects "Exit" from the menu,
20 // which could lead to an infinite loop if there is a bug in the code.
21
22 // 5. The printCheckLine, printCheckRectangle, and printCheckCircle variables are declared but never used.
23
24 // 6. The Circle class constructor takes three arguments, but the input prompt only asks for two points.
25 // It's not clear what the second and third arguments correspond to.
26
27 // 7. There are some typos and syntax errors, such as missing semicolons at the end of lines.
28
29 // 8. The cmath library is included, but none of its functions are used in the code.
30
31 // 9. The VectorOfPoints class doesn't contain any member functions, so it's unclear what its purpose is.
32
33 // it is important to note that code quality issues can indirectly impact security.
34 // Poorly written code can contain vulnerabilities that can be exploited by attackers.
35
36
37 //Includes and Defines
38 #include <iostream>
39 #include <vector>
40 #include <string>
41 #include <sstream>
42 using namespace std;
43
44 class Line {
45 public:
46     int a, b, c, d;
47     Line() {}
48     void LineCalculator(int a, int b, int c, int d) {}
49     double LineSlopeReturn() { return 0; }
50     double LineAngleReturn() { return 0; }
51     double LineLengthReturn() { return 0; }
52 };
53
54 class Rectangle {
55 public:
56     int a, b, c, d;
57     Rectangle() {}
58     void RectangleCalculator(int a, int b, int c, int d) {}
59     double RectangleWidth() { return 0; }
60     double RectangleLength() { return 0; }
61     double RectangleCalculationPointX() { return 0; }
62     double RectangleCalculationPointY() { return 0; }
63     double RectangleCalculationPointXPlusW() { return 0; }
64     double RectangleCalculationPointYPlusL() { return 0; }
65 };
66
67 class Circle {
68 public:
69     int a, b, c;
70     Circle() {}
71     void CircleCalculator(int a, int b, int c) {}
72     double CircleCircumferenceReturn() { return 0; }
73     double CircleAreaReturn() { return 0; }
74 };
75
76 class VectorOfPoints {
```

Security Assessment – C++ Caluclator

```
77 public:
78     vector<string> myvector;
79 };
80
81 class PointMerge {
82 private:
83     int X, Y;
84
85 public:
86     PointMerge() : X(0), Y(0) {}
87     void setPoint(int a, int b) {
88         X = a;
89         Y = b;
90     }
91     int getX() { return X; }
92     int getY() { return Y; }
93     string getMerge() {
94         ostringstream oss;
95         oss << X << ", " << Y;
96         return oss.str();
97     }
98 };
99
100 int main()
101 {
102     Line lineObj;
103     Rectangle recObj;
104     Circle circleObj;
105     VectorOfPoints vectorObj;
106     PointMerge pointObj;
107     char input;
108
109     do {
110         cout << "\nEnter an option (l for line, r for rectangle, c for circle, v for vector of points, m for merging points): ";
111         cin >> input;
112
113         switch (input) {
114             case 'l': {
115                 int a, b, c, d;
116                 cout << "Enter values for line (a, b, c, d): ";
117                 cin >> a >> b >> c >> d;
118                 lineObj.LineCalclator(a, b, c, d);
119                 cout << "Slope: " << lineObj.LineSlopeReturn() << endl;
120                 cout << "Angle: " << lineObj.LineAngleReturn() << endl;
121                 cout << "Length: " << lineObj.LineLengthReturn() << endl;
122                 break;
123             }
124             case 'r': {
125                 int a, b, c, d;
126                 cout << "Enter values for rectangle (a, b, c, d): ";
127                 cin >> a >> b >> c >> d;
128                 recObj.RectangleCalclator(a, b, c, d);
129                 cout << "Width: " << recObj.RectangleWidth() << endl;
130                 cout << "Length: " << recObj.RectangleLength() << endl;
131                 cout << "Top left point: (" << recObj.RectangleCalculationPointX() << ", " << recObj.RectangleCalculationPointY() << ")" << endl;
132                 cout << "Bottom right point: (" << recObj.RectangleCalculationPointXPlusW() << ", " << recObj.RectangleCalculationPointYPlusL() << ")" << endl;
133                 break;
134             }
135             case 'c': {
136                 int a, b, c;
137                 cout << "Enter values for circle (a, b, c): ";
138                 cin >> a >> b >> c;
139                 circleObj.CircleCalclator(a, b, c);
140                 cout << "Circumference: " << circleObj.CircleCircumferenceReturn() << endl;
141                 cout << "Area: " << circleObj.CircleAreaReturn() << endl;
142                 break;
143             }
144             case 'v': {
145                 int n;
146                 cout << "Enter number of points to add to vector: ";
147                 cin >> n;
148                 cout << "Enter points (x, y) separated by space or newline:" << endl;
149                 for (int i = 0; i < n; i++) {
150                     int x, y;
151                     cin >> x >> y;
152                     pointObj.setPoint(x, y);
```

Security Assessment – C++ Caluclator

```
153         vectorObj.myvector.push_back(pointObj.getMerge());
154     }
155     break;
156 }
157 case 'm': {
158     int x, y;
159     cout << "Enter two points to merge (x1, y1) (x2, y2): ";
160     cin >> x >> y;
161     PointMerge p1, p2;
162     p1.setPoint(x, y);
163     cin >> x >> y;
164     p2.setPoint(x, y);
165     cout << "Merged point: (" << p1.getX() + p2.getX() << ", " << p1.getY() + p2.getY() << ")" << endl;
166     break;
167 }
168 default:
169     cout << "Invalid input. Please try again." << endl;
170     break;
171 }
172 } while (input != 'q');
173
174 return 0;
175 }
176
```

5. Works Cited

- “Encryption 101.” *EDUCAUSE.edu*, <https://www.educause.edu/focus-areas-and-initiatives/policy-and-security/cybersecurity-program/resources/information-security-guide/toolkits/encryption-101>.
- IT Supply Chain. “Third-Party Access and Cyber Security Vulnerabilities.” *IT Supply Chain*, 23 Sept. 2020, <https://itsupplychain.com/third-party-access-and-cyber-security-vulnerabilities/>.
- J;, Teoli D;Sanvictores T;An. “SWOT Analysis.” *National Center for Biotechnology Information*, U.S. National Library of Medicine, <https://pubmed.ncbi.nlm.nih.gov/30725987/>.
- “Programming Language Security: These Are the Worst Bugs for Each Top Language.” *ZDNET*, <https://www.zdnet.com/article/programming-language-security-these-are-the-worst-bugs-for-each-top-language/>.
- “Secure Coding Guide.” *Race Conditions and Secure File Operations*, 13 Sept. 2016, <https://developer.apple.com/library/archive/documentation/Security/Conceptual/SecureCodingGuide/Articles/RaceConditions.html>.
- Whittaker, George, and George Whittaker is the editor of Linux Journal. “How-Tos.” *Home*, <https://www.linuxjournal.com/content/basic-linux-commands>.