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

Security Assessment Report

Version N.1

April 29, 2023

Table of Contents

1. Summary

The goal of this security assessment is to find as many security holes in the BitmapEx project possible and report on them for the intent of fixing them. Major findings include possible entrypoints for code injection, discovering access control issues, and risk of buffer overflow from use of raw pointers.

1. Assessment Scope

Tools, platforms, OSes, and software used in testing include the Visual Studio 2019 Community Edition IDE, GitHub platform, and Windows 10 Home OS. The software tested on is the BitmapEx project. Major limitations in testing include no financial resources, limited human and time resources, and limited security testing knowledge.

2. Summary of Findings

Of the findings discovered during our assessment, 6 were considered High risks, 4 Moderate risks, 2 Low, and 0 Informational risks. The SWOT used for planning the assessment are broken down as shown in Figure 1.

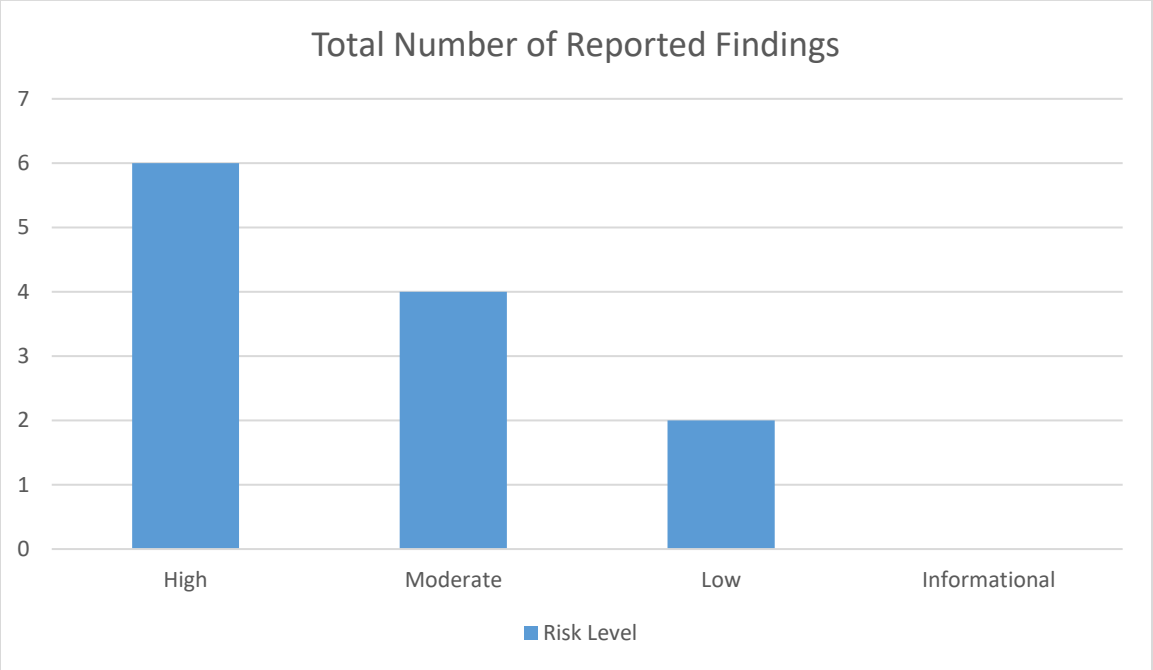


Figure 1. Findings by Risk Level

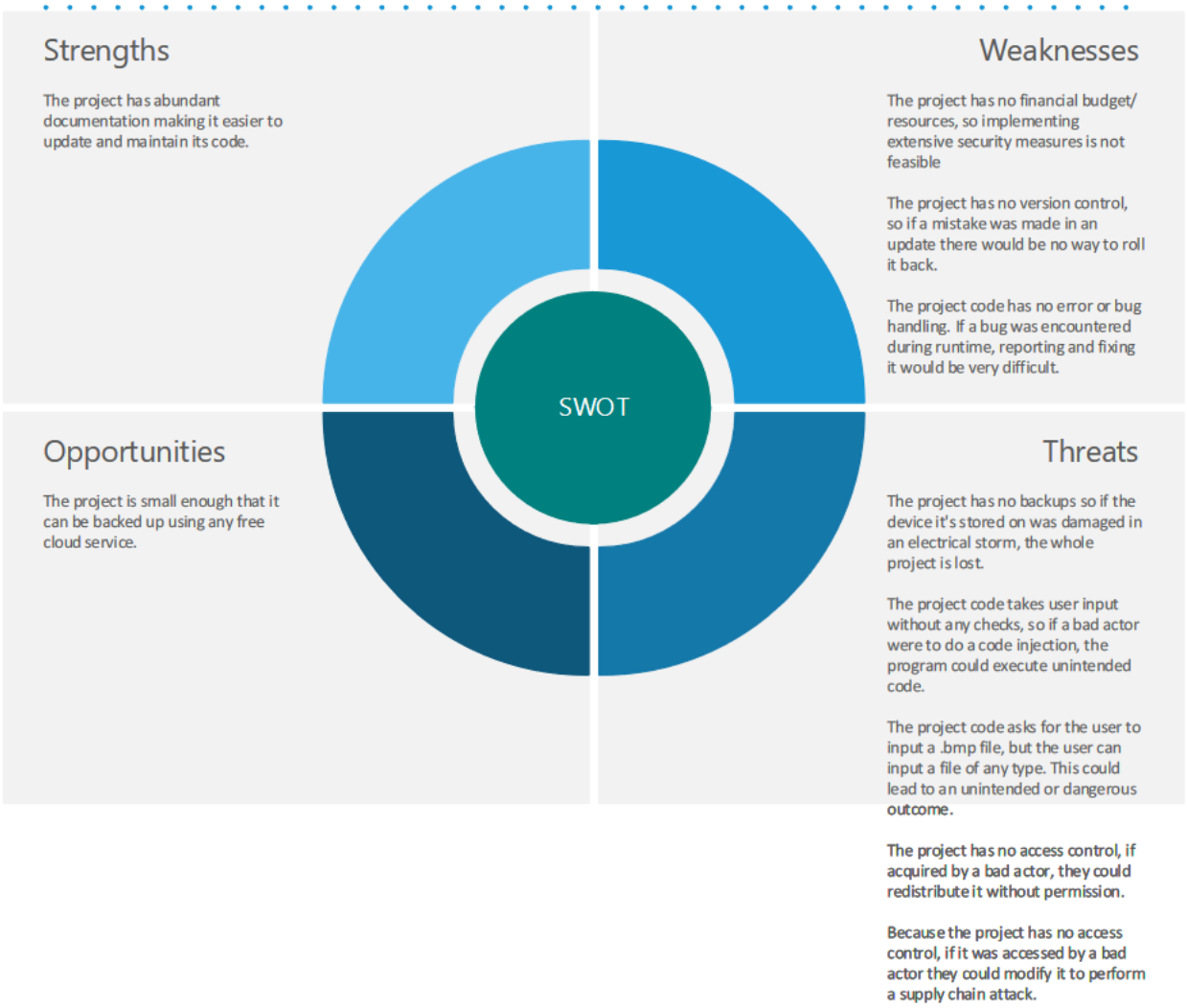


Figure 2. SWOT

Major issues detected within the project include: Lack of access control measures. Anyone with access to the source code has full ability to read, write, modify, and redistribute. Lack of project backup could result in complete project loss. Program takes in inputs with no restrictions/sanitization/whitelisting which could result in buffer overflow, code injection, and program failure.

3. Summary of Recommendations

Changes made include fixes to protect against code injection and type conversion code injection by using string sanitization methods. Complete project loss avoided by implementing backups. Access controls, version history, and logging implemented through hosting the project on the GitHub online platform. Disposed of any sensitive project documentation that is no longer needed.

Changes that still need to be made include restructuring the project to use smart pointers over raw pointers to avoid buffer overflows.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to become aware of all of the this project's risks, threats, and vulnerabilities. Also, to learn how to address these issues and to keep security and regulation compliance in mind when developing future projects.

2. Detailed Findings

Findings from the assessment include:

- [High Risk]: The project's code has no access control measures.
- [High Risk]: Anyone with access to the source code can read/write/modify it freely.
- [High Risk]: Anyone with project access can redistribute it.
- [Moderate Risk]: Project has no version control so no way to revert changes.

  The project has no access control measures, with anyone who has access to it being able to read, write, modify, and redistribute it freely opening it up to the possibility of a supply chain attack. Along with that, were any tampering to be done to the source code there would be no way to revert to an older version.

- [High Risk]: No backup could cause complete project loss.

  Lack of a backup could cause complete project loss in the event of the device the code is contained on being damaged beyond repair.

- [High Risk]: Users can load an unintended file format into the program
- [High Risk]: Users can cause a buffer overflow because program uses raw pointers
- [Moderate Risk]: Program takes user input as-is with no checks
- [Moderate Risk]: Program's string inputs have no sanitization/whitelisting
- [Moderate Risk]: Program's char inputs have no protection against type conversion code injection

  No checks or countermeasures were taken against the possibility of code injection or buffer overflow in the source code. All inputs were taken directly through std::cin and the input buffer was never cleared between inputs. Raw pointers are used instead of smart pointers. If an attacker attempted to execute unintended code through the inputs, there would be no resistance.

- [Low Risk]: Program has no logging for error handling
- [Low Risk]: Program has no logging for what users pass through it

  The program has no accounting in place against program instability or potential attacks. Lack of logging will make tracking down errors or suspicious user behavior nearly impossible.

3. Recommendations

Fixes against these vulnerabilities include:

- Hosting the project on a secure online platform such as GitHub to apply access controls to protect against tampering, version history to rollback any unintended changes, accounting through logging, and protection against redistribution.

- Implementing both physical and online secured backups for the source code and updating them as necessary to prevent a severe/total loss of project code in the possibility of theft or hardware failure.

- Implementing proper input sanitization and using more secure C++ data structures over ones from C as well as replacing all raw pointers used in the code with smart pointers will protect the program from buffer overflows, code injection, and memory issues it is currently vulnerable to.

- Implementing proper logging methods for the program itself to promote error detection and protection against improper usage of the program which could lead to total program failure during runtime.

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 **Error! Reference source not found.** 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

Rating	Definition of Risk Rating
	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

Accurate business risk findings according to Table 1:

- [High Risk]: The project's code has no access control measures
- [High Risk]: Anyone with access to the source code can read/write/modify it freely
- [High Risk]: Anyone with project access can redistribute it
    - The three risks above could lead to a supply chain attack which would be seriously damaging to the business.
- [High Risk]: Users can cause a buffer overflow because program uses raw pointers
    - This risk could possibly lead to the system crashing or possibly creating an entry point for a cyber attack. In a business scenario this could lead to a possible data breach as a buffer overflow could give an attacker control over the system.
- [Low Risk]: Program has no logging for error handling
- [Low Risk]: Program has no logging for what users pass through it
    - The two risks above are not integral to program stability, and the logging itself would not include any sensitive information such as PI or PII and would not be a risk to the business if compromised.

Inaccurate findings:

- [High Risk]: Users can load an unintended file format into the program
    - The risks above, while it is a high risk to the program's functionality, would only be a moderate business risk. Were it to be exploited the program would only fail and stop functioning.
- [Moderate Risk]: Program takes user input as-is with no checks
- [Moderate Risk]: Program's string inputs have no sanitization/whitelisting
- [Moderate Risk]: Program's char inputs have no protection against type conversion code injection
    - The three risks above would only be moderate risk for this project, but for a business scenario could easily lead to a data breach or loss of business data. Because their consequences would go beyond just program failure, they would be high business risks.

3.1.2 Tests and Research

Tests were completed using whitebox testing. Research into some of the security issues were found from reading the Cpp Core Guidelines provided by Bjarne Stroustrup and Herb Sutter. Others were learned during the CEN 3078 Computer Security FGCU course.

3.1.3 Tools

This was completed using Visual Studio 2019.

4. Figures and Code

Examples of insecure code:
Possible type conversion code injection point:

```
// Menu Choice
char choice;
std::cout << "\nWould you like to: ";
std::cout << "\nEncrypt a message into a bitmap file [E]";
std::cout << "\nDecode a message from this bitmap file [D]";
std::cout << "\n>";
std::cin >> choice;

if (std::tolower(choice) == 'e') {              // Encrypt
```

Possible string code injection points:

```
std::string inFileName, outFileName;
outFileName = "dummy.bmp";

std::cout << "\nEnter bitmap file name to read: ";
std::cin >> inFileName;
```

```
std::string encryptedFileName;
std::cout << "\nEnter new encrypted bitmap filename to write to: ";
std::cin >> encryptedFileName;

std::string messageToEncrypt;
std::cout << "\nEnter message to encrypt into " << encryptedFileName << ".\n" << ">";
std::cin >> messageToEncrypt;
```

Raw pointers used instead of smart pointers:

```
EncryptedBMFile* encryptedBMFile = reinterpret_cast<EncryptedBMFile*>(&bmFile);
```

```
EncryptedBMFile* decodedBMFile = reinterpret_cast<EncryptedBMFile*>(&bmFile);
```

Extra Figures:
Risk Assessment Matrix:

		Probability ---------->				
Severity		Frequent	Probable	Likely	Possible	Rare
|	Emergency	The project's code does not have any access control measures Anyone who has access to the project can read/write/modify it freely				No backup causing complete project loss
|	Major		Anyone who has access to the project can redistribute it without permission	User can load an unintended file format into the program User can output an unintended file format from the program		
|	Moderate		Program takes user input as-is with no checks		Project has no version control in case of a mistake	
|	Minor	Program does not log what files the user passes through it			Program has no error codes/bug handling in place	
|	Negatable					
V						

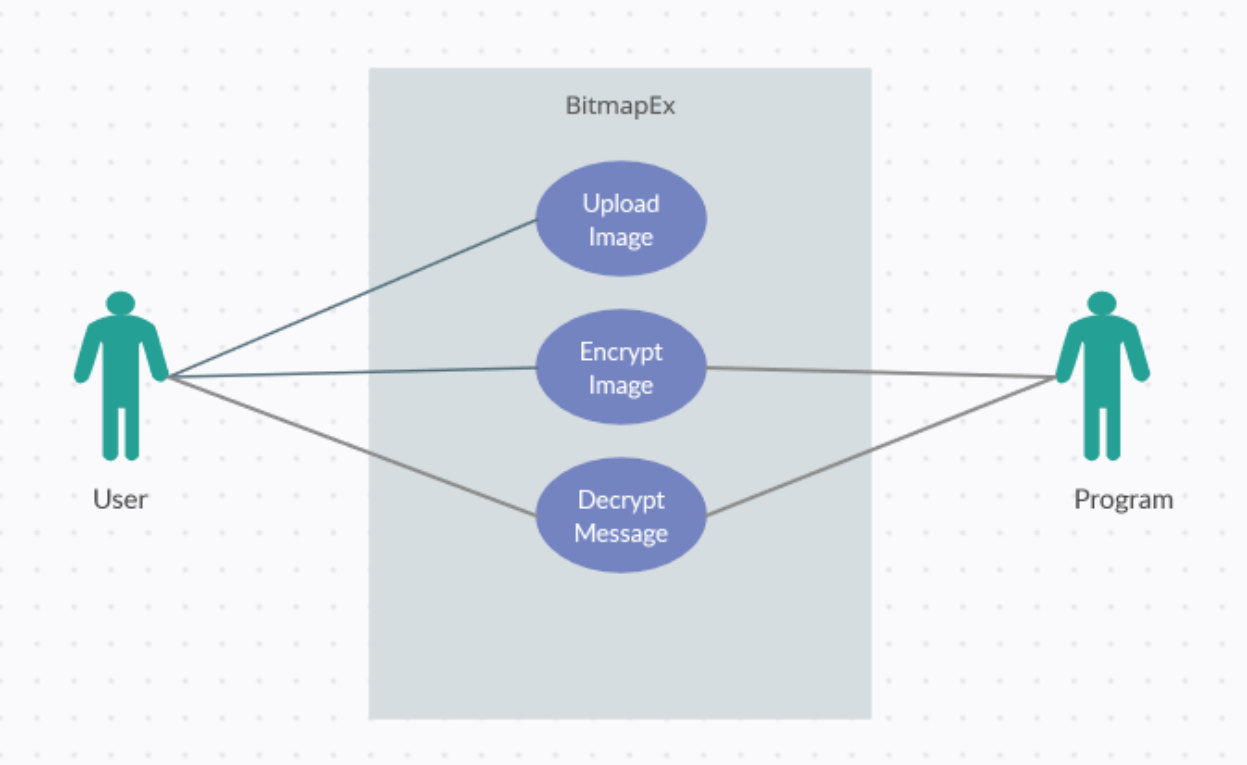
Access Control and Network Security Checklist:

Security Assessment – BitmapEx

Access Control and Network Security Checklist						
ID	Security Control	Applicable (Y	Complete (Y/N/I	To be Done Priority (High, Mid	Ease of fix (Easy, Moderate, Difficult, No	Full Description of processes to add
1	A cloud based platform is being used for access control with only public available items being readable by general public.	Y	Y	Mid	Easy	The cloud platform to be used is Github which provides project writing access to only the creator and trusted collaborators.
2	The cloud based platform provides for hiding information and this is used to protect sensitive information and code.	Y	Y	Mid	Easy	Any sensitive information and code can be stored in a separate private repo in Github.
3	OS access controls are used to only allow authorized changes to be made to code.	Y	N/A			N/A
4	Platform user groups are used to only allow changes to be made to code by authorized	Y	Y	Low	Easy	Only trusted individuals would be added as a trusted collaborator to be given write access.
5	Backup Policy is in place and being used.	Y	Y	High	Easy	Backups will be handled both physically and through a cloud based platform, Google Drive.
6	Third-Party libraries used in code are up-to-date and have been checked to ensure no security issues exist.	N	N/A			N/A
7	Physical Security of actual computer code is stored on is adequete	Y	Y	Mid	Difficult	Project code stored on my computer will be encrypted and password protected. Cloud-based code backups will require 2FA to access through Google Drive.
8	Accounting: Logging is integrated into the code itself (for exceptions, errors, and user input failures at minimum)	Y	N	Mid	Moderate	Logging will be integrated into the project to account for and report any erroneous outcomes to a text file.
9	Accounting: Process includes logging (tracking of changes, user making changes, access attempts, etc)	Y	Y	Low	Moderate	Accounting involving changes made and who made them will be handled through Github version and access control.
10	PKI and other encryption and authentication methods are used to connect to cloud	Y	Y	Mid	Easy	SSL is the authentication method used to connect to GitHub, and the website is ceritifed by DigiCert.
11	Internal Actor threats are accounted for and policies/planning is in place for these.	N	N	Low	Moderate	I do not plan to collaborate on this project with anyone, so trust policies and plans are not needed.
12	Standard Unit Testing used	Y	Y	Mid	Moderate	Unit testing was already completed during the project's creation.
13	Security Testing used (the type varies)	Y	N	High	Moderate	Security testing was not performed during the project's creation, and security tests will be made for its existing functions.
14	Any sensitive project documentation that is no longer useful is stored securely or disposed of	Y	Y	Low	Easy	All sensitive documentation pertaining to the project that is no longer used has been moved to a private repository or deleted permanently, depending on its relevancy.

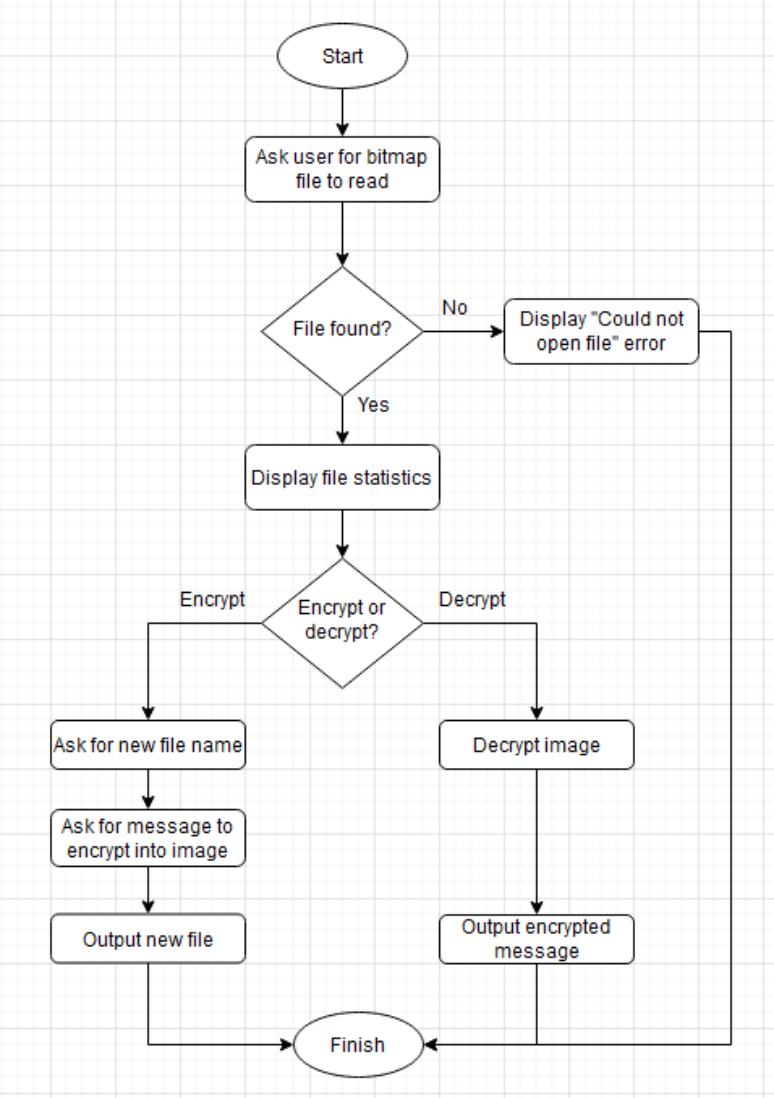
4.1.1 Process or Data flow of System use-cases, security checklist, graphs, etc.

Use case diagram of BitmapEx system:



Security Assessment – BitmapEx

Flowchart of BitmapEx system:



In the BitmapEx program, the user is first asked by the system to select an appropriate bitmap file. If the file isn't found, then the "file not found" error will be displayed and the program will finish. If the file is found and can be read, the properties of the file are displayed. The user is then given a choice to encrypt a message into a new copy of the file or to decrypt the message contained in the current file.

If encrypting:

> The user is then asked for a new filename to write the new encrypted bitmap file to, and then asked for the message to encrypt as well. The system then outputs the new encrypted bitmap file.

If decrypting:

> The system will decrypt the image and print out the decoded message for the user to read.

The program then finishes.

5. Works Cited

Stroustrup, Sutter (2015) CppCoreGuidelines [Markdown Documentation].
https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md