**Logo, company name

Description automatically generated**

Food Ordering Application for Restaurants

**Security Assessment Report**

Version N.1

May 1, 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 figures of code 7](#_Toc127779456)

[5. Works Cited 7](#_Toc127779457)

# Summary

During this assessment, there were many vulnerabilities found within the application including: unsecured software, lack of encryption, and no security plan in place. These vulnerabilities pose a risk to the Restaurant and its security. This could lead to unauthorized access, user data theft, and other malicious attacks. It is recommended to fix these vulnerabilities as soon as possible by: implementing encryption on the users email and passwords, creating a detailed security plan, and reevaluating all code to create a more refined application.

## Assessment Scope

The scope of the assessment includes:

- Investigating data storage

- Evaluating policies and procedures in place for incident response

- Evaluating backup systems in place

- Access control

Windows 10 was the only OS used for testing.

Limitations include:

- Limited budget for testing

- Lack of access to multiple operating systems for testing and analysis

- Limited team to work on testing

## Summary of Findings

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

Figure 1. Findings by Risk Level

**SWOT ANALYSIS**

Text

Description automatically generated

Figure 2. SWOT

After careful investigation, major security issues were found within this application. First, user data is stored in a very unsecure manner, increasing the risk of data being compromised. The application also lacks user authentication which allows for unauthorized users to create accounts using other people’s email address. Next, the repository is very unorganized, making version control nearly impossible to track. Finally, the lack of a robust security plans leaves the application open to potential threats without any guide for how to navigate them.

## Summary of Recommendations

A few changes have been made to this application to ensure higher security. First, a Github repository was created to track version control and to have a web-based copy of the application. Next, a more robust login system was created to handle keeping track of user information. Encryption for this is nonexistant and needs to be updated to a complex system. Using a third-party company to handle credit card transactions is highly recommended to avoid liability. Creating a security plan before the application hits public eye is very important to ensure customer safety and trust in the 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 assignment details
* Determine if the application was securely maintained.
* Identify vulnerabilities
* Evaluate security plan

## Detailed Findings

* Unsecure Password Storage
  + A lack of encryption is a very serious vulnerability found within this application. It leaves users’ sensitive data exposed to potential attackers. It is vital that this is fixed as soon as possible.
* Faulty Input Validation
  + There are multiple instances where the application asks for user input and allows for invalid input. This opens the opportunity for attackers or even well-meaning users that just input the wrong information to crash the app by overloading it. DOS and Injection attacks are also risky possibilities.
* No security plan
  + There is no security plan set in place for any potential security threats. It is important to develop one because it minimizes damage caused in case of an incident, ensures thorough employee training, and can avoid legal issues if the application needs to comply with specific regulations.
* Public Repository
  + Storing the application in a public repository is a huge vulnerability in the application. The user’s information is stored in a public, unencrypted file for the public to see. This is considered an urgent fix to make.
* Version Control
  + The repository is not utilized to its fullest extent and every change was made to the main branch. This makes tracking version control very difficult, which can make reverting back to previous versions if need be.
* Use of a Text file for information
  + The user’s information is stored in a text file that lacks security features found within using a proper database. This is a weakness of the program because it is vulnerable to unauthorized people manually messing up the file. Text files aren’t suitable to store large amounts of data as needed to store mass amounts of user information.
* No Authorization when creating an account
  + Currently there is no system in place to verify the person who is signing up for the account owns the email they used to sign up with. The current system would allow anyone to sign up with any email address, causing issues when the actual owner of the email account tries to sign up with the rewards program. This would be a weakness in the program.

## Recommendations

* Unsecure Password Storage/Use of a Text File for Information
  + Utilizing a database would fix these vulnerabilities and make for a more organized user information storage. Modern database systems handle encryption well and keep data integrity way better than standard text files. Databases would also prevent issues when multiple people make an account at the same time.
* Faulty Input Validation
  + It is vital to make sure input validation handles all invalid input with no errors. Leaving this causes security vulnerabilities like injection attacks and buffer overflows which are detrimental to the application.
* No security plan
  + Creating a security plan utilizing the results of this security assessment is a high priority. This ensures that if any attacks occur, there are steps in place to mitigate the risk and protect the company and the users.
* Public Repository/Version Control
  + It is recommended that the entire GitHub repository is reorganized and set to private. There are branches made but they are not actually utilized; everything is being pushed and pulled to the main branch. By using branches, it helps with version control and allows developers to work simultaneously without affecting other branches.
* User Authorization
  + Implementing robust user authorization is vital for protecting individual users. By sending a verification email to the user to finish setting up the account, it ensures that users are who they say they are. If they don’t confirm verification after a set amount of time, it would be considered safe to delete the potential account since the user isn’t whom they say they are.

# 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 |

**Risk Assessment:**

Chart, treemap chart

Description automatically generated

**Figure 3**

It was found that encryption for user’s information was very weak. There exists a high possibility that an unauthorized user could access a users account and cause issues for the user. By utilizing a third-party system, it would eliminate the need to store user’s card information within the application, eliminating the risk. It was found that phishing emails would more likely go after customers over employees since there is no difference between employee accounts and user accounts.

**3.1.2 Tests and Analyses**

This was completed using:

* Penetration testing
* Whitebox testing
* Research into system vulnerabilities

Whitebox testing was done by code review, unit testing, and integration testing. The source code was reviewed for and inconsistencies or defective code. Each input possible was tested with different cases and input types to see how well it holds up to intentional wrong input. It was found with many inputs, that entering the wrong information breaks the system.

Based upon research done by Ji-Jian Chin, I have considered other vulnerabilities not previously covered. I found that the application is vulnerable to Impersonation Orders. There exists no system to verify the users email is actually their own so it is possible for an attacker to use a stranger’s email as their own.

**3.1.3 Tools**

No external tools were used for testing.

# Figures and Code

### Process or Data flow of System

Use-Case Diagram

Diagram

Description automatically generated

Flow Chart

Diagram

Description automatically generated

Risk Assessment:

Chart, treemap chart

Description automatically generated

SWOT:

Text

Description automatically generated

### Other figure of code

Shows the initial lack of encryption for the users data.

Graphical user interface, application

Description automatically generated

# Works Cited

* <https://github.com/juliachickonski/SecurityFinalProject>: code repository
* <https://github.com/codergopher/CPP-Login-Program>: Used for assistance with the Login System and Encryption
* Week 3 Powerpoint: Risk\_to\_threat.ppxt
* Lampson Butler W., Computer Security in the Real World
* Ju-Juan Chin, Vulnerabilities in Online Food Ordering Website
* [(PDF) Vulnerabilities in Online Food Ordering Website (researchgate.net)](https://www.researchgate.net/publication/338637802_Vulnerabilities_in_Online_Food_Ordering_Website#:~:text=Two%20of%20the%20most%20widely,cause%20to%20the%20victim%20business.)