Security Assessment Report

Version N.1

May, 1, 2023

Table of Contents

1. Summary

This assessment was conducted through a manual review of the code and identifying potential security weaknesses. The major findings included several vulnerabilities such as lack of input validation, insufficient access control, and weak cryptographic practices. To fix these issues, recommendations were made such as implementing proper input validation, access control, and using strong cryptographic algorithms. Additionally, it is recommended to perform thorough testing to ensure the code is secure and all identified vulnerabilities have been addressed.

1. Assessment Scope

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

Tools: Visual Studio Code
OSes: Windows 11, MacOS Ventura
Browsers: Firefox, Google Chrome
Software: .NET Framework, Swagger UI, Sqlite Database, Entity Framework Core

2. Summary of Findings

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

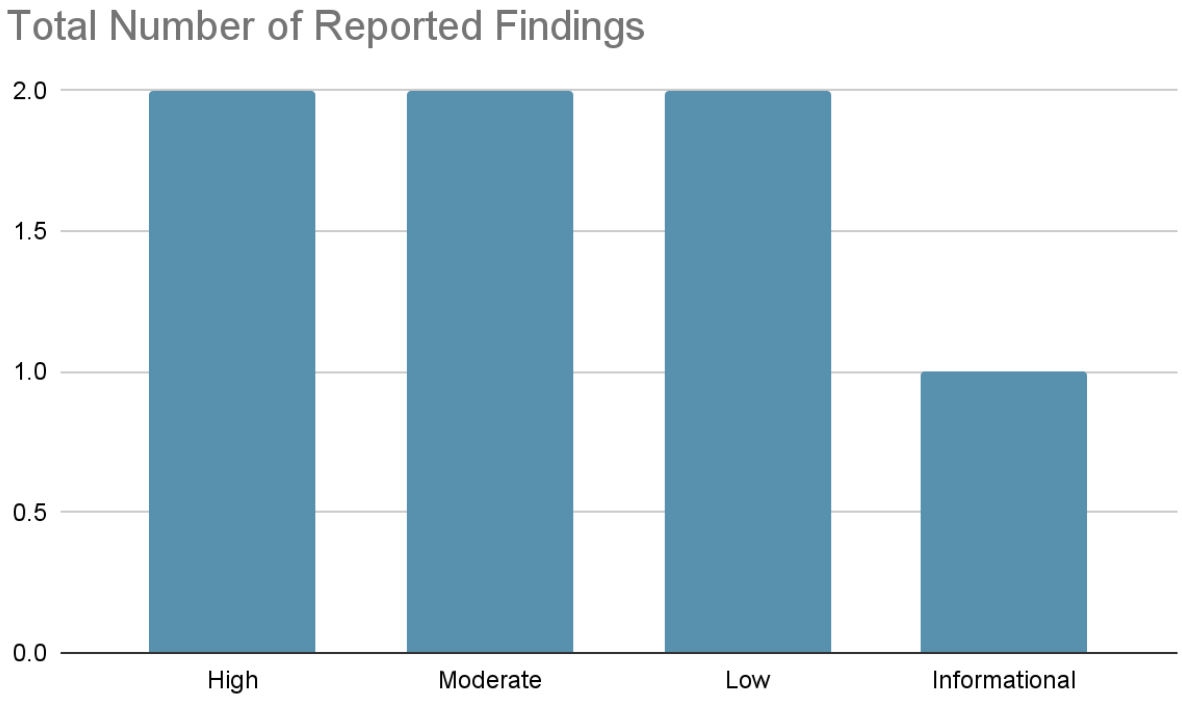


Figure 1. Findings by Risk Level

High Risks: Injection attacks could be a potential issue as there are no input validation or sanitization measures implemented for the GetAllTodos method's parameters.
Broken authentication and authorization could be a concern since the code does not appear to have any implementation of these security measures, which could allow unauthorized access to sensitive data or functionality.

Moderate Risks: Cross-Site Scripting (XSS) could be a potential issue as there is no input validation or output encoding in place for the CreateTodosDto and UpdateTodoDto objects, which could enable attackers to inject malicious scripts. Insufficient logging and monitoring may be a concern as there is no implementation of logging or monitoring, which could make it difficult to detect and respond to potential attacks or security incidents.

Low Risks: Cross-Site Request Forgery (CSRF) could be a potential issue as there is no implementation of CSRF prevention measures, which could allow attackers to trick users into submitting malicious requests. Information leakage could be a concern as sensitive data such as the user ID and other potentially identifying data may be leaked by the application.

Informational Risks: Secure coding practices should be considered, such as implementing input validation, output encoding, and error handling, to help mitigate potential security risks. However, given the current simplicity of the application, these measures may not be a high priority at this time.

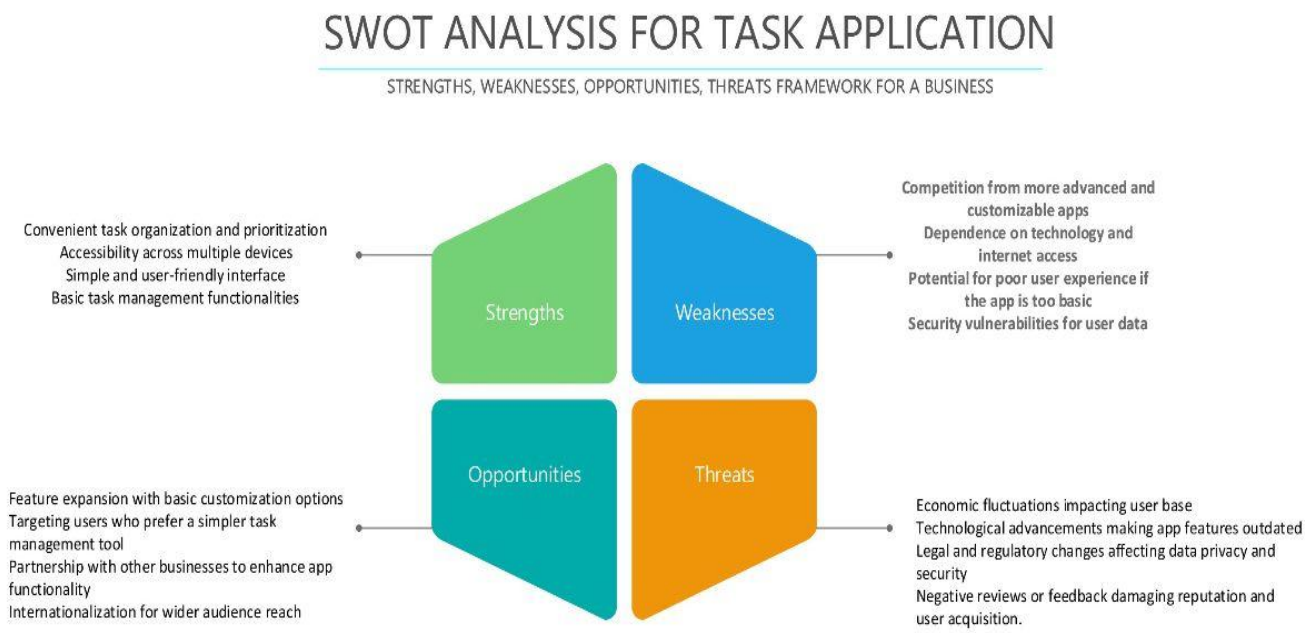


Figure 2. SWOT

3. Summary of Recommendations

Changes that are yet to be made include implementing input validation, access control, and proper error handling, among other things. Once these changes are made, it is important to perform thorough testing to ensure that the code is secure and any vulnerabilities have been addressed.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to do the following:

- Identify potential security vulnerabilities in the given code.
- Provide recommendations for fixing the identified vulnerabilities.
- Improve the overall security posture of the application.
- Increase the confidence of the users in the security of the application.

2. Detailed Findings

The code provided has several security risks that need to be addressed. One significant risk is the lack of authentication and authorization in the API endpoints. If left unaddressed, unauthorized users can access the API endpoints and modify or delete sensitive data, resulting in data breaches, loss of information, and damage to the organization's reputation.

Another risk is the lack of input validation and sanitization. This weakness can result in injection attacks, where attackers can inject malicious code into input fields and execute unauthorized actions. This can lead to data breaches, system crashes, and even complete system compromise, posing a significant risk to the organization's operations.

The use of hard coded secrets, such as database credentials, is another risk in the code. Hard Coding credentials can make it easier for attackers to steal them, and gain unauthorized access to the system. This can result in data breaches, system crashes, and damage to the organization's reputation.

Moreover, the code exhibits a risk of insufficient logging and monitoring. This weakness can make it harder to detect and respond to security incidents. Delayed incident response can increase the severity of the breach, resulting in potential regulatory non-compliance, and increased damage to the organization's operations.

3. Recommendations

SQL Injection: Use parameterized queries or prepared statements, which allow the database to separate the query logic from user input. Another approach is to use an ORM (Object-Relational Mapping) tool like Entity Framework, which automatically sanitizes user input by escaping special characters.

Cross-Site Scripting (XSS): Encoding functions that convert special characters to their HTML entities.

Insufficient Authorization and Authentication: Using password hashing, multi-factor authentication, and role-based access control.

3. Methodology for the Security Control Assessment

3.1.1 Risk Level Assessment

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 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 tests could be created for each endpoint to ensure that the expected output is returned for valid input and that errors are handled correctly. Unit tests could also be created for the database interactions to ensure that data is being retrieved and manipulated correctly. Also, security testing could include penetration testing to identify vulnerabilities in the endpoints, authentication testing to ensure that only authorized users are able to access the endpoints,

and data validation testing to ensure that input is properly sanitized and validated before being processed.

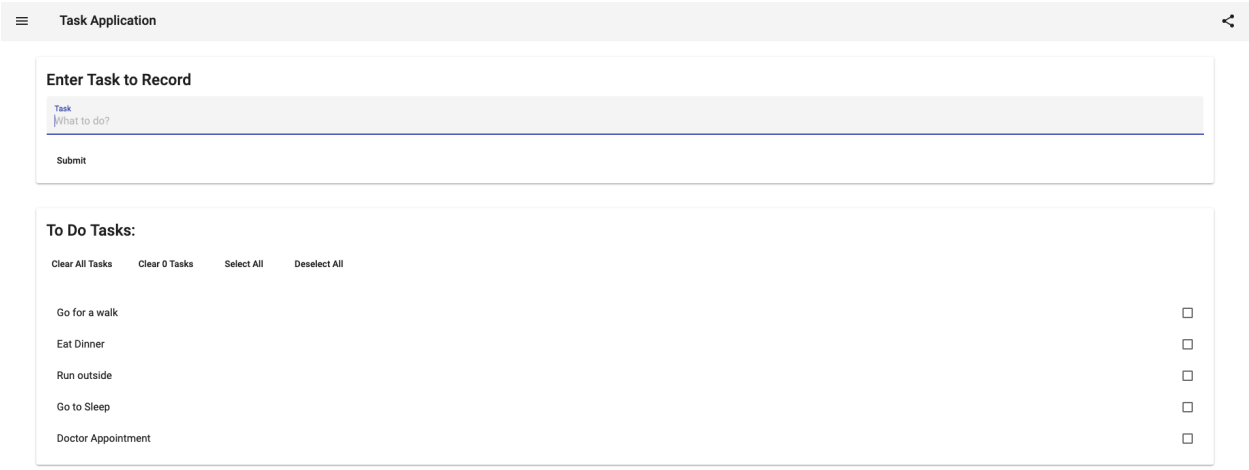
3.1.3 Tools

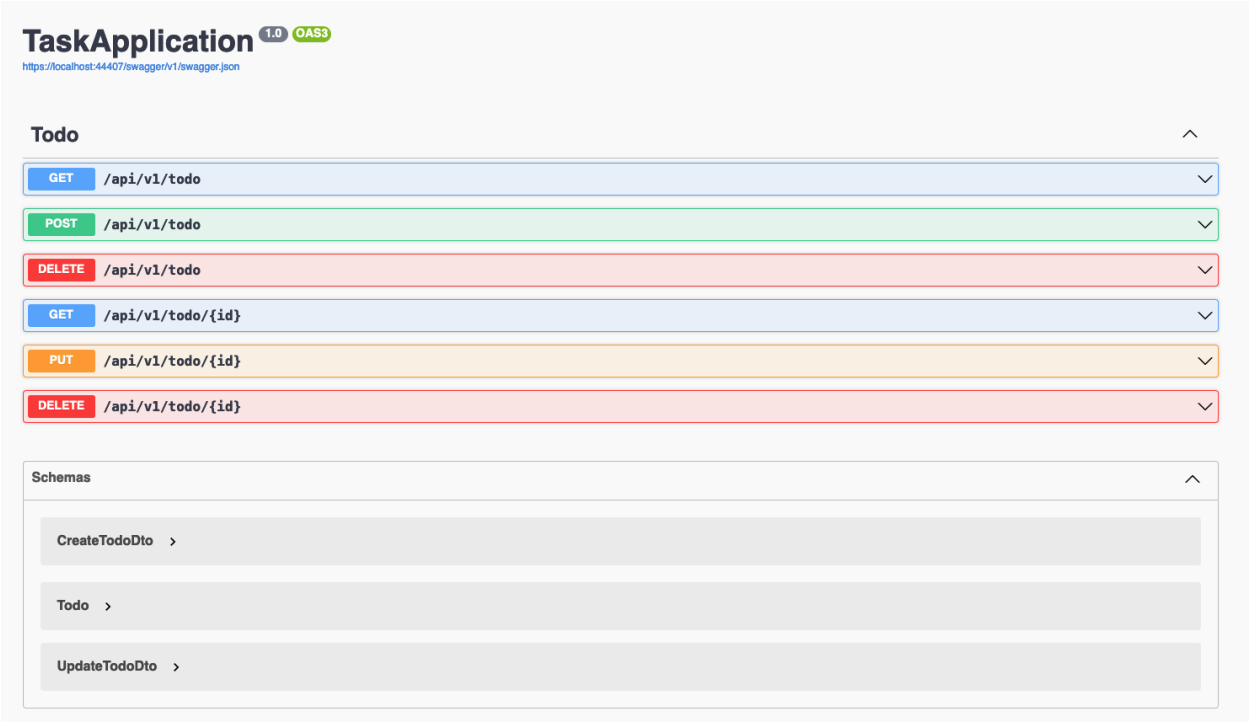
sqlmap: This is an automated tool for SQL injection testing, which can help identify vulnerabilities in web applications that use SQL databases.

BurpSuite: This is a popular web application security testing tool that allows you to intercept, modify, and replay web traffic.

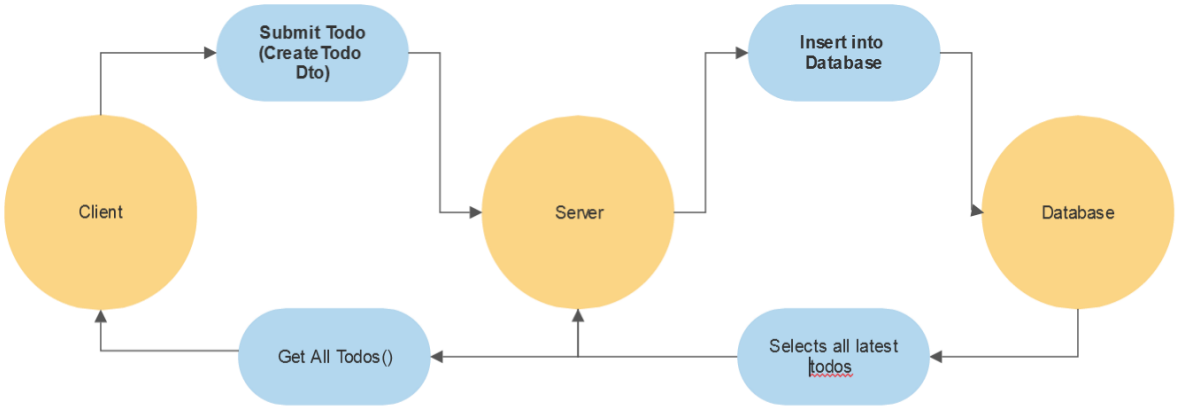
4. Figures and Code

https://github.com/silverlui/TaskApplication





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



5. Works Cited

Github:
https://github.com/silverlui/TaskApplication

Sql Injection:
TryHackMe | SQL Injection

Pentesting:

TryHackMe | Pentesting Fundamentals

Security Testing Tools:

TryHackMe | Introduction to OWASP ZAP