# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **09/26/24** | **Andrew Corrigan** |  |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In this report, identify your security vulnerability findings and recommend the next steps to remedy the issues you have found.

* Respond to the five steps outlined below and include your findings.
* Respond using your own words. You may also include images or supporting materials. If you include them, make certain to insert them in the relevant locations in the document.
* Refer to the Project One Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Andrew Corrigan

**1. Interpreting Client Needs**

Determine your client’s needs and potential threats and attacks associated with the company’s application and software security requirements. Consider the following questions regarding how companies protect against external threats based on the scenario information:

* What is the value of secure communications to the company?
* Are there any international transactions that the company produces?
* Are there governmental restrictions on secure communications to consider?
* What external threats might be present now and in the immediate future?
* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

The fundamental nature of financial planning requires that client information be stored securely. Financial information is highly sensitive and as such, the value of robust encryption and secure communication is immeasurable for clients. Ensuring the safety and integrity of clients financial information is essential to building trust and reducing risks.

Given the scenario, it can be assumed that Artemis Financial interacts with international clients, although it is not explicitly stated.

If assumed correctly, this would require Artemis to comply with regulations such as the Gramm-Leach-Bliley Act, so that client data is stored securely.

The company faces threats any financial institution would such as phishing, SQL injection, and data breaches.

Using open-source libraries requires that the libraries are frequently updated. As web technologies evolve, secure coding practices and regular updates to software need to be implemented.

**2. Areas of Security**

Refer to the vulnerability assessment process flow diagram. Identify which areas of security apply to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

Input Validation: Unvalidated inputs in the API can be found in the application can present a security risk.

APIs: Unsecured API endpoints in the application can expose sensitive data, making the system vulnerable to unauthorized access.

Cryptography: Lack of encryption in the application for data transmission can result in sensitive data being intercepted by attackers.

Client/Server Communication: Insecure communication between the client and server can leave the system vulnerable to attacks.

Code Error Handling: Error handling that exposes detailed information can give attackers insights into the system.

Code Quality: Outdated and insecure dependencies used in the application can introduce vulnerabilities.

Encapsulation: Failure to properly encapsulate sensitive data can allow unauthorized access to critical information.

**3. Manual Review**

Continue working through the vulnerability assessment process flow diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

1. There is a lack of input validation in the CRUDController leaving the application susceptible to SQL injection.

2. In DocData database credentials are hardcoded which can lead to unauthorized access.

3. Parametrized queries are not used in the read\_document method, again leaving the application susceptible to SQL injection.

4. There is no encryption for data transmission.

5. The API endpoint /read lacks access control which can expose sensitive data.

6. Weak error handling can expose sensitive system information.

7. Outdated libraries were found which as discussed above can introduce security risks.

**4. Static Testing**

Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from the dependency-check report. Include the following items:

* The names or vulnerability codes of the known vulnerabilities
* A brief description and recommended solutions provided by the dependency-check report
* Any attribution that documents how this vulnerability has been identified or documented previously

A screenshot of a computer

Description automatically generated

log4j-api-2.12.1.jar: Log4j is vulnerable to remote code execution (RCE) attacks, known as the "Log4Shell" exploit. This allows attackers to take control of the system and execute arbitrary code.

spring-boot-2.2.4.RELEASE.jar: This version of Spring Boot contains multiple vulnerabilities, including authentication bypass and remote code execution risks. Attackers may exploit these flaws to gain unauthorized access or execute arbitrary code on the server.

spring-boot-starter-web-2.2.4.RELEASE.jar: The Spring Boot Starter Web module has multiple critical vulnerabilities that expose the application to attacks, including remote code execution and data exposure.

spring-core-5.2.3.RELEASE.jar: This version of Spring Core is susceptible to multiple vulnerabilities, including authentication issues and remote code execution vulnerabilities.

tomcat-embed-core-9.0.30.jar: Multiple vulnerabilities in Tomcat Embed Core expose the system to remote code execution and server-side request forgery (SSRF). This could allow attackers to exploit the server and execute malicious actions.

tomcat-embed-websocket-9.0.30.jar: Tomcat WebSocket is vulnerable to critical issues that could lead to remote code execution or unauthorized access to sensitive data.

snakeyaml-1.25.jar: SnakeYAML contains a deserialization vulnerability that could be exploited to perform remote code execution.

The solution for all these vulnerabilities stated in the dependency check is simply to update to the latest version, which will resolve the critical issues and protect the system from exploitation.

**5. Mitigation Plan**

Interpret the results from the manual review and static testing report. Then identify the steps to mitigate the identified security vulnerabilities for Artemis Financials’ software application.

To address the variety of concerns, several steps need to be taken to ensure the application is safe, secure, and efficient. The primary action to address the vulnerabilities in Artemis Financials’ software is to update all outdated libraries to their latest versions. This is a straightforward step, which will resolve critical issues and vulnerabilities. Input validation should be implemented across all API endpoints, and parameterized queries should replace direct SQL calls to prevent injection attacks. API endpoints should also be secured with proper authentication. Encrypting data in transit and at rest using will further protect sensitive information, and error handling should be improved to avoid exposing sensitive system details. Hardcoded credentials need to be replaced with secure storage methods and regular monitoring and updating should be performed to keep the application secure against future threats.