# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **05/31/2024** | **MINNU** |  |

## 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

MINNU JOHN

**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?

Value of Secure Communications: Ensuring secure communications is crucial to protect sensitive financial information of clients.

International Transactions: If Artemis Financial handles international transactions, it must comply with international security standards and regulations.

Governmental Restrictions: Depending on the jurisdiction, there may be specific regulations such as GDPR for Europe or HIPAA for health-related financial data in the US.

External Threats: Current threats include phishing, ransomware, and zero-day exploits. Future threats might evolve with technology, requiring constant vigilance.

Modernization Requirements:

Open-source Libraries: Ensure they are up-to-date and free of known vulnerabilities.

Evolving Web Technologies: Use secure and modern web development frameworks and technologies.

**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.

Architecture Review: Analyze the overall architecture to ensure it follows best practices for security.

Input Validation: Ensure all user inputs are validated to prevent injection attacks.

APIs: Secure API interactions to protect data in transit.

Cryptography: Use strong encryption methods for data storage and transmission.

Client/Server: Ensure secure communication between client and server.

Code Error Handling: Implement proper error handling to avoid exposing sensitive information.

Code Quality: Maintain high coding standards to prevent security flaws.

Encapsulation: Use secure data structures to protect data integrity.

**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. Injection Flaws: Look for SQL injection or other injection vulnerabilities.

2. Broken Authentication: Ensure authentication mechanisms are robust.

3. Sensitive Data Exposure: Check if sensitive data is properly encrypted.

4. XML External Entities (XXE): Verify XML parsing is secure.

5. Security Misconfiguration: Ensure security settings are configured correctly.

6. Cross-Site Scripting (XSS): Check for unsanitized user inputs in web pages.

7. Insecure Deserialization: Ensure data deserialization is handled securely.

8. Using Components with Known Vulnerabilities: Ensure all libraries and dependencies are up-to-date.

9. Insufficient Logging and Monitoring: Verify logging mechanisms are in place and monitored.

10. Improper Error Handling: Ensure error messages do not expose sensitive information.

**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

Known Vulnerabilities: List the names or codes of the identified vulnerabilities.

Descriptions and Recommendations:

Attribution: Document how the vulnerability was identified, such as through a known CVE database or security advisory.

**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 Financial’s software application.

1. Injection Flaws: Implement parameterized queries and input validation.

2. Broken Authentication: Use multi-factor authentication and secure password storage (e.g., bcrypt).

3. Sensitive Data Exposure: Ensure all sensitive data is encrypted at rest and in transit.

4. XML External Entities (XXE): Disable external entity processing in XML parsers.

5. Security Misconfiguration: Regularly review and update security configurations.

6. Cross-Site Scripting (XSS): Use output encoding and sanitization libraries.

7. Insecure Deserialization: Validate and sanitize all serialized data before processing.

8. Using Components with Known Vulnerabilities: Regularly update dependencies and use tools to monitor for vulnerabilities.

9. Insufficient Logging and Monitoring: Implement comprehensive logging and monitor for suspicious activities.

10. Improper Error Handling: Ensure that error messages do not expose stack traces or sensitive information.