# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[Date]** | **[Your name]** |  |

## 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

[Rodrigues Mavita]

**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?

[**Client’s Needs and Security Requirements**:

* **Value of Secure Communications**:
* Secure communications are crucial for Artemis Financial to protect sensitive financial data and maintain client trust.
* Financial plans, investment strategies, and personal client information require encryption to prevent unauthorized access.
* **International Transactions**:
* Artemis Financial may handle international transactions, requiring compliance with international security standards such as GDPR and PCI-DSS.
* Secure communication channels and data encryption are necessary to protect cross-border data transfers.
* **Governmental Restrictions**:
* Compliance with regulations such as GDPR (for European clients), CCPA (for California clients), and other relevant financial industry regulations.
* Adherence to encryption standards and secure data storage practices mandated by regulatory bodies.
* **External Threats**:
* Potential threats include phishing attacks, data breaches, SQL injection, cross-site scripting (XSS), and man-in-the-middle attacks.
* Immediate future threats may involve sophisticated malware, ransomware attacks, and zero-day vulnerabilities.
* **Modernization Requirements:**
* Adoption of open-source libraries and evolving web application technologies like RESTful APIs.
* Ensuring security measures are in place for integrating third-party libraries and APIs.
* Implementing modern authentication mechanisms (OAuth, JWT) and secure development practices (DevSecOps).]

**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.

[**Relevant Areas of Security**:

* **Authentication and Authorization**:
* Ensuring that only authorized users can access specific resources and perform actions within the application.
* Implementation of multi-factor authentication (MFA) to enhance security.
* **Data Encryption**:
* Encryption of data in transit and at rest to protect sensitive financial information from unauthorized access.
* Use of SSL/TLS for secure communication channels.
* **Input Validation**:
* Preventing SQL injection, XSS, and other input-based attacks by validating and sanitizing user inputs.
* Use of prepared statements and parameterized queries.
* **Access Control:**
* Implementing role-based access control (RBAC) to ensure users have appropriate permissions.
* Regular audits of access control policies and logs.
* **Error Handling and Logging**:
* Secure error handling to avoid revealing sensitive information through error messages.
* Comprehensive logging of security-related events for monitoring and auditing purposes.
* **Third-Party Dependencies**:
* Regularly updating and patching third-party libraries and dependencies to mitigate known vulnerabilities.
* Use of dependency-check tools to identify and address vulnerabilities in third-party components.
* **Secure Configuration**:
* Ensuring secure default configurations for servers, databases, and application settings.
* Regularly reviewing and updating configurations to adhere to security best practices.]

**3. Manual Review**

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

[**Identified Vulnerabilities (7-10 Findings)**:

1. **SQL Injection Vulnerability**:

* Description: Lack of parameterized queries in database operations.
* Solution: Implement prepared statements and parameterized queries.

1. **Cross-Site Scripting (XSS):**

* Description: Unsanitized user inputs displayed in the web interface.
* Solution: Properly encode and sanitize user inputs before rendering them in the UI.

1. **Insecure Authentication Mechanism**:

* Description: Use of outdated or weak password hashing algorithms.
* Solution: Implement strong hashing algorithms like bcrypt or Argon2.

1. **Sensitive Data Exposure:**

* Description: Storing sensitive information (e.g., passwords) in plaintext.
* Solution: Encrypt sensitive data before storing it in the database.

1. **Insufficient Logging and Monitoring**:

* Description: Lack of logging for security-related events.
* Solution: Implement comprehensive logging and monitoring for all critical operations.

1. **Insecure Default Configuration**:

* Description: Default configurations expose unnecessary services and ports.
* Solution: Harden configurations by disabling unused services and ports.

1. **Lack of Input Validation**:

* Description: User inputs are not validated or sanitized.
* Solution: Implement strict input validation and sanitization.]

**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

[**Dependency Check Findings**:

* **Known Vulnerabilities**:
* Name: CVE-2021-44228 (Log4j)
* Description: Remote code execution vulnerability in Log4j library.
* Recommended Solution: Update to the latest patched version of Log4j.
* **Names or Vulnerability Codes**:
* Vulnerability: CVE-2021-45046
* Description: Denial of service vulnerability in Log4j.
* Recommended Solution: Update to the latest patched version of Log4j.
* **Attribution**:
* Identified through NVD and CVE databases.
* Previous documentation and industry reports on Log4j vulnerabilities.

**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.

[**Mitigation Steps**:

* **SQL Injection**:
* Implement prepared statements and parameterized queries for all database operations.
* **Cross-Site Scripting (XSS):**
* Sanitize and encode all user inputs before displaying them in the UI.
* **Insecure Authentication Mechanism:**
* Upgrade to strong password hashing algorithms like bcrypt or Argon2.
* **Sensitive Data Exposure**:
* Encrypt sensitive data before storing it and use secure key management practices.
* **Insufficient Logging and Monitoring**:
* Implement comprehensive logging for all critical operations and monitor logs for suspicious activities.
* **Insecure Default Configuration:**
* Review and harden all default configurations, disabling unnecessary services and ports.
* **Lack of Input Validation:**
* Implement strict input validation and sanitization for all user inputs.
* **Third-Party Dependencies**:
* Regularly update and patch third-party libraries. Use dependency-check tools to identify and address vulnerabilities.
* **Data Encryption**:
* Ensure all data in transit is encrypted using SSL/TLS. Encrypt sensitive data at rest.
* **Access Control:**
* Implement and regularly audit role-based access control (RBAC) policies.]