# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **01/26/2025** | **Lasupe Xiong** |  |

## 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

Lasupe Xiong

**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 application used by Artemis Financial, which handles sensitive financial data, is an essential component of its business operations. To safeguard sensitive information from outside attacks, secure connections are necessary. Important factors for this company include:

* **Value of Secure Communications**: Retaining confidence with partners and clients requires protecting the privacy and integrity of data. Unauthorized access, data breaches, and man-in-the-middle attacks may all be avoided with secure communications.
* **International Transactions**: Artemis Financial may need to comply with international standards like the GDPR or cross-border data transfer legislation if it conducts business with clients or transactions from other countries. To safeguard data while it is in transit, secure communication protocols like HTTPS, TLS, and VPNs are crucial.
* **Governmental Restrictions**: Government regulations may prohibit the use of specific encryption technology or mandate particular protocols for secure communication, especially in sectors like finance.
* **External Threats**: Phishing attacks, ransomware, SQL injection, and cross-site scripting (XSS) are examples of current threats. There are also serious concerns associated with the surge in cybercrime and the complexity of assaults.
* **Modernization Requirements**: Since the software depends on open-source libraries, it's critical to assess how well-maintained and updated these libraries are. Adapting web application technologies, including microservices and APIs, need constant focus on patch management, safe coding techniques, and vulnerability monitoring.

**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.

Based on the vulnerability assessment process flow, the following areas of security are relevant to Artemis Financial’s software application:

* **Application Security**: Application-level security is essential because of the sensitive financial data that is handled. This covers safeguarding input validation, user authentication, and protection against threats like XSS and SQL injection.
* **Network Security**: For safe data transfer, it is crucial to make sure that all conversations are secured using protocols like TLS. Intrusion detection systems (IDS) and firewalls are two examples of network security tools that will aid in the defense against outside threats.
* **Data Security**: Client financial records and other sensitive information must be encrypted both in transit and at rest. Secure key management procedures and robust encryption methods must be used.
* **Access Control**: To guarantee that only authorized personnel may access sensitive data or carry out certain tasks inside the application, appropriate authentication and authorization procedures must be in place.
* **Vulnerability Management**: To find and fix vulnerabilities in the application code and libraries, regular vulnerability scans, static analysis, and dependency checks are crucial.

**3. Manual Review**

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

The manual code review process aims to identify security vulnerabilities directly within the application’s source code. Below are the identified issues:

1. **SQL Injection**: User inputs are not adequately validated in a number of places in the code before being used in database queries.
2. **Cross-Site Scripting (XSS)**: The application's forms may be vulnerable to XSS attacks due to inadequate sanitization of user-generated material.
3. **Insecure Deserialization**: Unsafe object deserialization on specific endpoints exposes the application to remote code execution (RCE).
4. **Hardcoded Credentials**: The codebase had certain hardcoded credentials, which might be dangerous if hackers manage to access the repository.
5. **Weak Password Storage**: The current algorithm for hashing passwords is antiquated and susceptible to brute-force assaults.
6. **Improper Session Management**: Because the session timeout is not enforced, idle sessions can run for longer periods of time, which raises the possibility of session hijacking.
7. **Insecure API Endpoints**: Sensitive endpoints can be accessed by unauthorized users due to certain APIs' inadequate permission and authentication checks.
8. **Exposed Sensitive Data**: Financial transactions and other sensitive information are recorded in unencrypted, making them vulnerable to hackers.
9. **Lack of Rate Limiting**: The application is susceptible to denial-of-service (DoS) attacks as it does not apply rate restriction on certain endpoints.
10. **Cross-Site Request Forgery (CSRF)**: Users are exposed to unauthorized activities because some forms are not secure against CSRF attacks.

**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

After running a dependency check on Artemis Financial’s software, the following vulnerabilities were identified:

**Vulnerability 1: CVE-2022-22965**

* **Description**: A remote code execution (RCE) vulnerability in Spring Beans when using data binding with Tomcat.
* **Recommended Solution**: Upgrade to Spring Beans version 5.2.6 or later.
* **Attribution**: Identified in OSSINDEX.

**Vulnerability 2**: CVE-2021-22060

* **Description**: Spring Web MVC is vulnerable to log injection, potentially leading to unwanted log entries.
* **Recommended Solution**: Upgrade to Spring Web MVC version 5.2.18 or later.
* **Attribution**: Identified in OSSINDEX.

**Vulnerability 3**: CVE-2022-22968

* **Description**: DataBinder's case sensitivity vulnerability in Spring Context.
* **Recommended Solution**: Upgrade to Spring Context version 5.2.20 or later.
* **Attribution**: Identified in OSSINDEX.

**Vulnerability 4**: CVE-2023-20861

* **Description**: SpEL vulnerability leading to denial of service (DoS) through resource exhaustion.
* **Recommended Solution**: Upgrade to Spring Expression version 5.2.22 or later.
* **Attribution**: Identified in OSSINDEX.

**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.

The manual review and static testing results suggest the following actions to address the vulnerabilities found:

1. **Patch Open-Source Libraries**: Upgrade all outdated Spring libraries (Spring Beans, Spring Web MVC, Spring Context, Spring Expression) to their latest secure versions to address known vulnerabilities.
2. **Sanitize Inputs**: Implement proper input validation and sanitization across the application to prevent SQL injection and XSS attacks. Use prepared statements and libraries that automatically escape user inputs.
3. **Remove Hardcoded Credentials**: Replace hardcoded credentials with environment variables or a secure secrets management solution to avoid exposing sensitive information in the codebase.
4. **Implement Strong Password Hashing**: Update the password hashing mechanism to use a secure algorithm such as bcrypt or Argon2 to protect stored passwords.
5. **Enhance Session Management**: Enforce session expiration after a set period of inactivity and implement secure cookie attributes (e.g., HttpOnly, Secure) to protect against session hijacking.
6. **Secure API Endpoints**: Apply strict authentication and authorization checks to all API endpoints, ensuring that only authorized users can access sensitive resources.
7. **Use Encryption for Logs**: Ensure that sensitive data is never logged in plaintext and implement encryption for sensitive logs.
8. **Enable Rate Limiting**: Apply rate limiting to prevent DoS attacks and ensure the application can handle high traffic securely.
9. **Implement CSRF Protection**: Use anti-CSRF tokens for all forms to protect against CSRF attacks.

By following these steps, Artemis Financial can improve the security of its software application and reduce its exposure to potential threats and attacks.