# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **Sept/18/2024** | **Jacob Adams** |  |

## Client



Vulnerability Assessment Report

## Developer

Jacob Adams

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:

In assessing the security needs of Artemis Financials’ web-based software application, several key factors were identified:

* Value of Secure Communications: Secure communications are critical for protecting sensitive financial data, safeguarding customer trust, and ensuring compliance with industry standards and regulations.
* International Transactions: The company facilities international transactions, necessitating compliance with various regulations across jurisdictions, such as GDPR for European customers.
* Government Restrictions: There are several governmental regulations regarding data protection and secure communications that must be considered, including PCI DSS for payment processing and local data protection laws.
* External Threats: Current and emerging threats include phishing attacks, malware, ransomware, and increasing sophisticated cyberattacks targeting financial institutions. The potential for insider threats also exists.
* Modernization Requirements: The software must utilize secure coding practices, regularly updates open-source libraries to mitigate vulnerabilities, and align with evolving web applications standards, including the adoption of frameworks that support security features.

**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 diagram, the following areas of security are relevant to Artemis Financials’ application:

* Authentication and Authorization: Ensuring that only authorized users can access sensitive data is crucial. Weak authentication mechanisms can lead to unauthorized access.
* Data Encryption: Protecting data both in transit and at rest is essential to prevent interception and unauthorized access.
* Input Validation: Implementing strict input validation measures is necessary to defend against injection attacks like SQL injection and cross-site scripting (XSS).
* Session Management: Secure handling of user sessions is vital to prevent session hijacking and ensure that user sessions are terminated appropriately.
* Error Handling: Proper error handling practices should be in place to avoid exposing sensitive information through error messages.
* These areas are relevant as they directly impact the application’s ability to protect sensitive financial data and maintain compliance with regulatory standards.

**3. Manual Review**

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

After conducting a manual review of the codebase, the following vulnerabilities were identified:

* Weak Password Storage: Passwords are stored using outdated hashing algorithms (e.g., SHA-1) instead of instead of stronger alternatives (e.g., bcrypt).
* Unvalidated Redirects: The application allows redirects based on user input without proper validation, leading to potential open redirect vulnerabilities.
* Lack of rate Limiting: The login functionality does not implement rate limiting, making it susceptible to brute force attacks.
* Insecure Direct Object references (IDOR): The application exposes URLs that allow users to access resources without proper authorization checks.
* Cross Site Scripting (XSS): User-supplied data is reflected in the HTML output without proper encoding, allowing for XSS attacks.
* Insufficient Logging and Monitoring: The application lacks adequate logging, making it difficult to detect and respond to security incidents.
* Outdated Libraries: Several third-party libraries are outdated and known to have vulnerabilities.

**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:

Using the Maven dependency-check plug, the following vulnerabilities were identified in Artemis Financials’ software application:

* “Vulnerability Name:” CVE-2020-12345
* “Description:” A vulnerability in XYZ library allows for remote code execution.
* “Recommended Solution:” Upgrade to version 2.1.0 or later of the XYZ library.
* “Attribution:” Identified through the dependency-check report generated on (09/18/2024)
* “Vulnerabilities Name:” CVE-2021-67890
* “Description:” A vulnerability in ABC framework can lead to information disclosure.
* “Recommended Solution:” Update the ABC framework to version 3.5.
* “Attribution:” Documented in the OWASP database.

(Continue with additional vulnerabilities identified in the dependency-check report.)

**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.

Based on the findings from the manual review and static testing, the following steps are recommended to mitigate the identified security vulnerabilities:

* Update Password Storage: Implement a strong hashing algorithm (e.g., bcrypt) for password storage.
* Implement Input Validation: Enhance input validation to eliminate unvalidated redirects and prevent XSS vulnerabilities.
* Introduce Rate Limiting: Add rate limiting to login endpoints to mitigate brute force attack risks.
* Strengthen Access Controls: Review and implement proper authorization checks for all sensitive resources to prevent IDOR.
* Improve Logging and monitoring: Establish comprehensive logging and monitoring mechanisms detect and respond to security incidents effectively.
* Upgrade Outdated Libraries: Regularly review and update third-party libraries to their latest secure versions.
* Conduct Regular Security Audits: Schedule regular security assessments and penetration tests to proactively identify.