# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11/13/2024** | **Bruce Gaudet** | Completed responses for all required sections, addressing client needs, areas of security, manual review findings, static testing results, and mitigation plan for Project One submission. |

## 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

Bruce Gaudet

**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?

Artemis Financial is a consulting company specializing in creating personalized financial plans for clients, including services for savings, retirement, investments, and insurance. Given the nature of its services, the protection of sensitive client information is paramount. Secure communication is essential for Artemis Financial to safeguard customer trust and prevent data breaches, especially during the exchange of financial data between clients and the API.

Artemis Financial may engage in international transactions, which could necessitate compliance with data protection regulations such as the General Data Protection Regulation for European clients. This compliance ensures the secure handling and storage of personal data across borders, emphasizing the importance of encryption in data transmission and storage. Additionally, Artemis Financial must prepare for external threats, including data breaches, SQL injections, and cross-site scripting attacks, which could compromise client data.

The modernization requirements include updating the existing software with secure, current technologies. Since Artemis Financial’s software relies on open-source libraries, it is crucial to monitor and update these dependencies regularly, as vulnerabilities in these libraries can expose the system to potential security risks. The evolving landscape of web application technology also necessitates that Artemis Financial adopt secure coding practices and regularly audit the system to mitigate emerging threats.

**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.

**\*\*\*Artemis Financial’s software application requires attention to several critical areas of security\*\*\***

**Architecture Review:** This area ensures that the application is designed to securely handle sensitive data at all stages, including data entry, storage, and transmission. A secure architecture minimizes data exposure and enables consistent protection across the entire system.

**Input Validation:** Proper input validation is necessary to protect the application from injection attacks, such as SQL injection and XSS. Ensuring all inputs are validated can prevent attackers from exploiting vulnerabilities in user input fields to gain unauthorized access to sensitive information.

**API Security:** As Artemis Financial’s application relies on a RESTful API, it is essential to secure endpoints against unauthorized access and ensure that data requests and responses are protected. Implementing API authentication and authorization measures would further secure client interactions with the API.

**Cryptography:** Cryptography is critical for securing sensitive data both at rest and in transit. Ensuring that sensitive information, such as financial data, is encrypted, helps to protect it from interception and unauthorized access.

**Code Quality:** Maintaining high code quality with secure coding practices helps reduce the risk of vulnerabilities introduced through insecure coding. Regular code reviews and adhering to secure coding standards will ensure the application remains resilient against attacks.

These areas align with Artemis Financial’s requirements for a secure application, addressing the vulnerabilities associated with handling financial data and mitigating risks from evolving security threats.

**3. Manual Review**

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

**In the manual review of the code base, the following seven vulnerabilities were identified:**

**Hardcoded Sensitive Data:** Some credentials and sensitive information appear to be hardcoded in the code base, which could allow unauthorized access if exposed. These should be managed through environment variables instead.

**Lack of Input Validation:** Certain input fields in the API do not validate user inputs adequately, leaving the system open to SQL injection or XSS attacks. Implementing input validation in these fields is necessary.

**Insecure Data Storage:** Sensitive information is stored in plain text in some parts of the application, increasing the risk of data leaks. Encryption should be applied to all sensitive data stored within the application.

**Improper Error Handling:** Some error messages in the code provide detailed information about the system, which could be used by attackers. Limiting error message details would mitigate this risk.

**Outdated Libraries:** Some libraries are outdated and have known vulnerabilities. Regularly updating dependencies to the latest secure versions will reduce exposure to these risks.

**Exposed API Endpoints:** Several API endpoints lack adequate security checks, potentially allowing unauthorized users access. Adding authentication and authorization checks would secure these endpoints.

**Use of Deprecated Functions:** Some functions used in the code base are deprecated and could introduce vulnerabilities. Replacing them with updated, secure alternatives will improve security.

**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

**The dependency-check report identified several critical vulnerabilities in the project’s dependencies:**

**spring-boot-2.2.4.RELEASE:**

**Vulnerability Code:** CVE-2020-5421

**Description:** This vulnerability allows attackers to execute arbitrary code by exploiting deserialization issues in spring-boot.

**Recommendation:** Update spring-boot to a newer, secure version to patch this vulnerability.

**bouncycastle-bcprov-jdk15on-1.46:**

**Vulnerability Code:** CVE-2016-1000344

**Description:** A cryptographic weakness in BouncyCastle allows unauthorized access to data if exploited.

**Recommendation:** Upgrade to a secure version of the BouncyCastle library.

**jackson-databind-2.10.2:**

**Vulnerability Code:** CVE-2020-36518

**Description:** This vulnerability in jackson-databind allows attackers to perform arbitrary code execution through unsafe deserialization.

**Recommendation:** Update to the latest secure version of jackson-databind.

These vulnerabilities highlight the importance of updating dependencies to maintain security. Artemis Financial should ensure that dependencies are regularly scanned and updated to mitigate these risks.

**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.

To address the identified vulnerabilities, Artemis Financial’s software application requires several key mitigation steps. First, any hardcoded sensitive data in the code base should be removed and stored securely in environment variables or configuration files, thereby reducing the risk of unauthorized access. Input validation should be applied across all input fields to prevent injection attacks such as SQL injection and cross-site scripting. This can be achieved by using parameterized queries and sanitizing all user inputs. Additionally, all sensitive data, whether at rest or in transit, should be encrypted using strong cryptographic algorithms, such as AES for storage and HTTPS for transmission, to ensure data confidentiality and integrity.

Error handling must also be enhanced to prevent attackers from gaining insights into the system. By limiting the detail provided in error messages, the application can avoid disclosing potentially sensitive information. The static testing report revealed several outdated dependencies, including spring-boot, bouncycastle, and jackson-databind, which have known vulnerabilities. Updating these libraries to their latest secure versions is essential to prevent exploitation of these vulnerabilities.

Furthermore, implementing robust authentication and authorization mechanisms for API endpoints will ensure that only authorized users can access sensitive data or perform specific actions, thereby enhancing API security. Lastly, deprecated functions should be replaced with secure, updated alternatives to improve both security and code quality. Collectively, these mitigation steps will significantly enhance the security of Artemis Financial’s application, protecting it against unauthorized access, data breaches, and other potential threats.