# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11/15/2024** | **Ashlyn Saucier** | **Initial version of vulnerability assessment report** |

## 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

Ashlyn Saucier

**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?
* Secure communications are vital for Artemis Financial as they handle sensitive financial information. This security ensures that data transmitted between the company and its clients is protected from interception and unauthorized access. Given the nature of their services, including savings, retirement, investments, and insurance, maintaining client confidentiality and trust is paramount. Secure communications help mitigate risks associated with data breaches, which can lead to financial losses and damage to the company’s reputation.
* Artemis Financial likely engages in international transactions when dealing with clients who reside in different countries or when offering financial products that require cross-border service. Such transactions necessitate compliance with various international regulations and standards that protect consumer data and govern secure financial communication.
* There are governmental regulations that Artemis Financial must adhere to, especially if they operate in multiple jurisdictions. In the United States, for example, the Gramm-Leach-Bliley Act (GLBA) mandates that financial institutions protect consumer data. Similarly, if operating in the European Union, they must comply with GDPR, which imposes stringent requirements on data protection and privacy.
* Current external threats include cyberattacks such as phishing, ransomware, and Distributed Denial of Service (DDoS) attacks, which pose risks to financial services firms. In the near future, increased sophistication in attack methods, such as social engineering tactics and zero-day vulnerabilities, may heighten the risks faced by Artemis Financial. The rise of geopolitical tensions can also raise cyber espionage threats against financial institutions.
* As part of modernizing its operations, Artemis Financial needs to consider adopting newer technologies and frameworks that enhance security. This includes migrating to cloud-based solutions with strong security features, implementing multi-factor authentication, and regularly updating their software stack to ensure they utilize the latest security patches. Additionally, the adoption of secure coding practices and the use of open-source libraries should be carefully assessed to ensure they are vetted for security vulnerabilities.

**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.

* Input Validation: Ensures that all user inputs are properly validated to prevent attacks like SQL injection and cross-site scripting (XSS). This is crucial given the financial nature of the services provided. Proper validation helps maintain data integrity and security.
* APIs: As Artemis Financial uses RESTful APIs, securing these interfaces is crucial to prevent unauthorized access and data breaches. This involves implementing authentication and authorization mechanisms, which must be integrated to protect sensitive operations.
* Cryptography: Proper encryption of sensitive data both at rest and in transit is essential to protect client information from unauthorized access. This is particularly important for all financial data exchanged through their systems.
* Code Quality: Ensures that the code adheres to secure coding standards, minimizing vulnerabilities that could be exploited by attackers, ensuring that best practices are followed throughout the development process.

**3. Manual Review**

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

* SQL Injection: Found in `UserDao.java` at line 45. The application concatenates user input directly into SQL queries without parameterization.
* Hard-coded Secrets: Found in `Config.java` where sensitive information, such as API keys, is hard-coded, risking exposure.
* Insufficient Exception Handling: In `ServiceLayer.java`, exceptions are not properly logged, making it difficult to track issues.
* Insecure API Endpoints: Discovered that some endpoints do not require authentication and can be accessed publicly.
* Unvalidated Redirects: In `RedirectController.java`, user input is used to determine redirects without validation, exposing the application to potential phishing attacks.
* Outdated Dependencies: Several dependencies, such as `jackson-databind`, are known to have vulnerabilities; specifics noted in the dependency check report.
* Lack of Input Sanitization: Input from forms is not sanitized, opening up potential cross-site scripting (XSS) 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
* bcprov-jdk15on-1.46.jar: Vulnerabilities like CVE-2024-34447 and others, indicating potential security risks in cryptographic functionality.
* jackson-databind-2.10.2.jar: Identified vulnerabilities include CVE-2020-25649, impacting the library's ability to safely deserialize untrusted data.
* spring-boot-2.2.4.RELEASE.jar: Critical vulnerabilities, including CVE-2023-20873, indicate weaknesses that could be exploited by attackers.
* A screenshot of a computer

  Description automatically generated

**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.

* SQL Injection: Refactor code to use prepared statements in `UserDao.java` to securely handle user input.
* Hard-coded Secrets: Move sensitive information to environment variables or secure configuration management tools to keep them out of the source code.
* Exception Handling: Implement thorough logging for exceptions in `ServiceLayer.java` using a structured logging approach to aid debugging and visibility.
* Secure API Endpoints: Implement token-based authentication for all API endpoints to restrict unauthorized access and use role-based access control where applicable.
* Input Sanitization: Implement input validation and sanitization on user inputs across all forms to prevent XSS attacks.
* Dependency Updates: Regularly check for and update dependencies to their latest secure versions. For example, upgrade `jackson-databind` and `bcprov-jdk15on`.
* Conduct Regular Security Audits: Schedule routine security assessments and code reviews to ensure ongoing compliance with security strategies and to identify any new vulnerabilities.