# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/19/2025** | **Joseph Roberts** |  |

## 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

Joseph Roberts

**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, a consulting firm specializing in personalized financial planning, is modernizing its operations and enhancing security for its RESTful web API, which handles sensitive client data such as savings, investments, and insurance. Secure communication is paramount to prevent data breaches, unauthorized access, and financial fraud, ensuring compliance with government regulations such as GDPR and U.S. financial laws. Given its involvement in international transactions, Artemis Financial must safeguard cross-border exchanges using robust encryption protocols like TLS (Transport Layer Security) to prevent data interception. However, the company faces various external threats, including phishing, malware, API vulnerabilities, and supply chain risks from third-party integrations. To stay ahead of evolving cybersecurity challenges, Artemis Financial must implement modern security measures such as OAuth 2.0 authentication, AES-256 encryption, multi-factor authentication, and secure software development practices to mitigate vulnerabilities. Strengthening defenses will protect financial assets, maintain compliance, and preserve trust with clients while ensuring seamless operations in a rapidly changing digital landscape

**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 is enhancing its RESTful web API to safeguard client financial data and prevent breaches and fraud. Secure communications and strong encryption are crucial, especially for international transactions and regulatory compliance with GDPR and U.S. financial laws. The company faces external threats like phishing, malware, and vulnerabilities in third-party integrations. Implementing OAuth 2.0 authentication, AES-256 encryption, and multi-factor authentication (MFA) will strengthen security, ensuring data protection, trust, and resilience in an evolving digital landscape.

**3. Manual Review**

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

* Hardcoded Credentials (DocData Class): This creates a serious security risk, as attackers can easily exploit credentials embedded in the code.
* Missing Input Validation(CRUDController Class): The API does not validate user input, this makes it vulnerable to injection and malicious inputs.
* Publicly Accessible Account Balance (Customer Class): Lacks proper encapsulation, allowing direct access and modification from outside the class.
* Lack of Authentication for API Endpoints: The CRUDController and GreetingController expose API endpoints without authentication or access control.
* Improper Handling of SQL Exceptions (DocData Class): This method catches SQL exceptions but does not log or handle errors properly, which can lead to unnoticed security breaches.
* Lack of Rate Limiting for API Calls: The application does not restrict repeated requests, making it vulnerable to denial-of-service (DoS) attacks.
* Weak Object Management (CRUD Class): The CRUD class stores data without proper data protection, increasing the risk of unintended data exposure.

**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

1. [**bcprov-jdk15on-1.46.jar**](file:///C:\Users\saint\eclipse-workspace\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7)CVE count: 22 multiple vulnerabilities – Update to latest version
   1. This library handles cryptographic functions but has known vulnerabilities related to weak encryption algorithms and exposure to timing attacks.
2. **hibernate-validator-6.0.18.Final.jar** CVE Count:2 medium – Update to latest version
   1. This validator is used for data validation, but outdated versions may allow insecure input handling, leading to potential injection attacks**.**
3. **jackson-databind-2.10.2.jar** CVE Count: 6 High – update to latest version
   1. Used for JSON serialization/deserialization, but previous versions have been exploited for remote code execution through deserialization flaws.
4. [**log4j-api-2.12.1.jar**](file:///C:\Users\saint\eclipse-workspace\rest-service\target\dependency-check-report.html#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb)CVE Count: 1 Low - Update to latest version
   1. Has minor security issues related to improper input sanitization.
5. **logback-classic-1.2.3.jar** CVE Count: 2 High – Update to latest version
   1. Logback is another logging framework, but older versions are susceptible to remote exploitation if improperly configured.
6. [**logback-core-1.2.3.jar**](file:///C:\Users\saint\eclipse-workspace\rest-service\target\dependency-check-report.html#l13_864344400c3d4d92dfeb0a305dc87d953677c03c)CVE Count: 4 High - Update to latest version
   1. Similar to Logback Classic, core components can be exploited for data exposure and untrusted logging input.
7. **snakeyaml-1.25.jar** CVE Count: 8 Critical – Update to latest version
   1. Used for processing YAML files, this outdated version is vulnerable to arbitrary code execution and file access manipulation.

**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 strengthen Artemis Financial’s security, several critical measures must be taken. Database credentials should be secured using environment variables instead of hardcoded values, reducing the risk of unauthorized access. Input validation must be enforced across API endpoints to prevent malicious attacks, while OAuth 2.0 and multi-factor authentication (MFA) will enhance access control. Rate limiting should be implemented to counter DoS attacks, and structured error logging will prevent exposing system details. Data security will be reinforced with AES-256 encryption, ensuring financial information remains protected. Additionally, all outdated dependencies must be updated, particularly vulnerable libraries like Jackson Databind and SnakeYAML, to eliminate security weaknesses