# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **07/20/25** | **Josef Zoucha** | **Assessed security concerns** |

## 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

Josef Zoucha

**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 client, Artemis Financial, is a firm that manages sensitive client data including savings plans, retirement plans, insurance details and investment records, secure communication and data protection are critical to maintaining client trust. Having secured communications can protect customer data whether its in transit or at rest, and we can ensure compliance with any data privacy regulations we need to follow.

Artemis Financial may produce international transactions based on the nature of the company, if so, the financial services would need to follow encryption protocols and compliance.

Some external threats that can be or are a problem are:

* Injection attacks.
* Insecure Input validation
* Broken Access Control
* Sensitive data exposure
* Weak authentication handling

Some modernization needs to consider would be to update and use secure, open-source libraries and to follow secure coding standards.

**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: Input values are not sanitized consistently which can create risks of injection.
* Secure Error Handling: Some errors may show unwanted internal logic.
* Securing Coding Practices: There are areas that would benefit from encapsulation.
* Secure API Interactions: API endpoints should need authentication and sanitize user input.
* Encryption: Sensitive data should be encrypted.
* Architecture Review: There is a lack of layered defenses in the application.

**3. Manual Review**

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

1. No input Validation (ClientController.java) – User data is accepted without being sanitized, which creates risk of injection attacks.
2. Hardcoded credentials (DBUtil.java) – Credentials are embedded directly in the code.
3. Weak exception handling (ClientDAO.java) – No logging or proper error messages.
4. Exposed stack traces (Many controllers share this) – printStackTrace() can be exploited to show internal logic.
5. Lack of access control (ClientController.java) – No role or token-based checks on endpoints.
6. Insecure database queries (ClientDAO.java) – SQL queries are built from string concatenation which creates risk of injection.
7. Sensitive data exposure (Client.java) – Client objects might expose sensitive fields.
8. No encryption for stored data (DBUtil.java, ClientDAO.java) – Sensitive data is not hashed or encrypted at rest.
9. Weak logging practices (ClientController.java) – Application lacks secure logging for tracking access.

**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

Vulnerability Description Recommended Solution Reference

|  |  |  |  |
| --- | --- | --- | --- |
| CVE-2020-9488 | Vulnerability in log4j-core:2.13.1 allows attackers to manipulate log output. | Upgrade to log4-jcore:2.17.1 or higher to prevent injection. | NVD – CVE-2020-9488 |
| CVE-2021-29425 | Insecure deserialization in commons-io:2.6 allows remote code execution. | Upgrade to commons-io:2.11.0 or higher. | NVD – CVE-2021-29425 |
| CVE-2017-18352 | In commons-beanutils, attackers can exploit deserialization flaws with class injection. | Avoid using vulnerable bean classes or update. | NVD – CVE-2017-18352 |
| CVE-2022-22965(Spring4Shell) | A critical vulnerability in certain versions of Spring Framework that enables remote code execution. | Upgrade to Spring Framework 5.3.18+ or 5.2.20+. | Spring Advisory |
| CVE-2019-2692 | A vulnerability in mysql-connector-java can potentially expose user credentials with local access. | Update to mysql-connector-java:8.0.19+. | NVD – CVE-2019-2692 |

Multiple libraries in the project are outdated and are critical components to the application.

**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.

1. No input Validation – Implement input validation using a validation framework.
2. Hardcoded credentials – Move DB credentials to encrypted configuration files.
3. Weak exception handling – Catch exceptions and log securely with a centralized logger.
4. Exposed stack traces – Replace printStackTrace() with a safe and generic error message and log internally.
5. Lack of access control – Implement token-based authentication.
6. Insecure database queries – Use ORM frameworks or prepared statements.
7. Sensitive data exposure – Encrypt or sensitive fields using AES or SHA-256.
8. No encryption for stored data – Use database encryption or encrypt sensitive data.
9. Weak logging practices – Integrate a secure logging framework and follow least-privilege logging.