# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/25/2025** | **Dexter Melton** | **Initial Creation** |

## 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

Dexter Melton

**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 communication is critical for a financial institution. Ensuring the correct regulations and defenses allows Artemis Financial to build client trust, ensure industry compliance, and prevent potential loss. It is not explicitly stated whether Artemis Financial operates internationally. Still, since Global Rain works with customers worldwide, it would be safe to assume that there are international transactions and ensure proper security methods are in place. There are numerous regulations on communication security for financial institutions, such as the Securities and Exchange Act, Digital Operational Resilience Act, and FTC Safeguards Rule, to name a few. Depending on Artemis Financial's region, which regulations specifically apply will be determined. External threats that could be present now are phishing attacks, API exploitation, and Man-in-the-Middle attacks. Future attacks could comprise zero-day vulnerabilities and third-party attacks, as well as leveraging new uses of AI for nefarious means. Ensuring all libraries being used are updated to the latest version and that coding practices are keeping pace with the evolving technology and necessary security methods are in place will go a long way to mitigating potential risks with the current modernizations.

**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
  + Ensuring proper input validation prevents attacks like SQL injection and cross-site scripting.
* APIs
  + Since the application uses a RESTful API, it needs to be secured with the proper authentication and authorization
* Cryptography
  + Since the focus is on secure communication, ensuring proper encryptions are used to pass data is essential.
* Error Handling
  + Ensuring that the proper error responses are in place prevents the potential exposure of sensitive information to attackers
* Code Quality
  + Anytime there are significant upgrades or modernization to software, it is crucial to ensure secure code practices are utilized, and the overall code quality is maintained.
* Encapsulation
  + When handling sensitive data, encapsulation is vital to ensure the proper separation between the application structures to prevent unintended access by users or attackers.

**3. Manual Review**

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

* CRUDController
  + There is no validation on the business\_name value
  + Does not require authentication for access
* DocData
  + The username and password are hard-coded
  + The read\_document method initializes a database connection but does not close it
  + There is a partially developed catch block, but it does not handle errors securely
* GreetingController
  + There is no validation for the parameter “name”
  + The project uses Java 1.8, which is out of date
  + Does not require authentication for 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

|  |  |  |  |
| --- | --- | --- | --- |
| **Dependency** | **Vulnerability** | **Description** | **Mitigation** |
| **bcprov-jdk15on-1.46.jar** | Outdated version | May contain cryptographic weaknesses or vulnerabilities to attacks. | Update to the latest version of BouncyCastle (e.g., 1.74 or newer). |
| **spring-boot-2.2.4.RELEASE.jar** | Outdated version | May lack security patches for known vulnerabilities. | Upgrade to a supported version of Spring Boot (e.g., 3.x or later). |
| **logback-classic-1.2.3.jar** | Vulnerable to insecure logging practices | Potential for log injection or sensitive data exposure. | Update to the latest version and configure secure logging practices (mask sensitive data). |
| **logback-core-1.2.3.jar** | Insecure default configurations | May expose sensitive information or allow exploitation. | Use a secure configuration and the latest version of Logback. |
| **log4j-api-2.12.1.jar** | Vulnerable to Log4Shell (CVE-2021-44228) | Allows remote code execution via crafted input. | Update to a safe version (e.g., 2.17.1 or newer). |
| **snakeyaml-1.25.jar** | Remote code execution vulnerability | Insecure deserialization of YAML data can lead to code execution. | Upgrade to the latest version of SnakeYAML. |
| **jackson-databind-2.10.2.jar** | Vulnerable to deserialization attacks | Insecure handling of data deserialization may allow remote code execution. | Update to a secure version (e.g., 2.15.x or later). |
| **tomcat-embed-core-9.0.30.jar** | Outdated version | May lack security patches and be vulnerable to attack vectors like directory traversal. | Upgrade to the latest version of Tomcat (e.g., 10.x or later). |
| **hibernate-validator-6.0.18.Final.jar** | Vulnerable to validation bypass attacks | May allow attackers to bypass data validation and exploit the system. | Update to the latest version and use strict input validation rules. |
| **spring-web-5.2.3.RELEASE.jar** | Outdated version | May lack fixes for known vulnerabilities, exposing APIs to attacks. | Upgrade to the latest version of Spring Web. |
| **spring-beans-5.2.3.RELEASE.jar** | Outdated version | May expose the application to dependency injection vulnerabilities. | Update to a supported and patched version. |
| **spring-webmvc-5.2.3.RELEASE.jar** | Outdated version | Potential exposure to insecure configurations or unpatched issues. | Upgrade to a secure and maintained version of Spring WebMVC. |
| **spring-context-5.2.3.RELEASE.jar** | Outdated version | May lack security fixes for vulnerabilities in application context configuration. | Update to the latest version and review context configurations for security. |
| **spring-expression-5.2.3.RELEASE.jar** | Expression injection vulnerability | May allow malicious expressions to execute unauthorized actions. | Upgrade to the latest version and validate all expressions before processing. |

**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.

Several key mitigation steps should be implemented to enhance the security of the application. First, ensure proper input validation and authorization mechanisms are added to both the GreetingController and CRUDController to prevent unauthorized access and malicious input. The error handling in the catch block of the DocData file should be completed to properly manage and log exceptions securely, reducing the risk of sensitive information exposure. Additionally, move the hardcoded database username and password to the application.properties file for secure storage and consider encrypting these credentials. A connection pooling mechanism should be implemented to manage database connections efficiently and prevent resource exhaustion. Finally, refer to the Maven vulnerability table and update all dependencies to their recommended secure versions to mitigate vulnerabilities associated with outdated libraries. These measures will significantly improve the application's overall security posture.