# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/25/2025** | **Ben Schmidt** | Updated sections 1, 2 and 3 |
| **1.1** | **3/26/2025** | **Ben Schmidt** | Updated sections 4 and 5 |

## 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

Ben Schmidt

**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?

For Artemis Financial, secure communications are critically important because they handle sensitive financial information for their clients. As a consulting company that develops individualized financial plans, they must protect data related to savings, retirement, investments, and insurance. The value of secure communication lies in maintaining client trust, preventing financial fraud, and ensuring the confidentiality of personal financial details. Given the nature of their business, they likely conduct international transactions, which increases the complexity of their security requirements and exposes them to a broader range of potential cyber threats.

The company must consider several key security challenges, including potential external threats like data breaches, unauthorized API access, and cyber-attacks targeting financial information. Modernization requirements demand careful management of open-source libraries, implementation of robust encryption, and staying current with evolving web application technologies. Governmental restrictions, such as international data protection regulations like GDPR, will also impact their security strategy. Artemis Financial needs to implement comprehensive security measures that protect against injection attacks, cross-site scripting, and unauthorized access while maintaining a flexible and efficient web application infrastructure that can adapt to changing technological landscapes.

**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.

For Artemis Financials’ software application, several key security areas from the vulnerability assessment process flow diagram are crucial. Input validation is essential to prevent malicious data entry and protect financial calculations. Cryptography is vital for securing sensitive customer financial information during transmission and storage. Secure API interactions are critical because the company uses a RESTful web application, requiring robust protection against unauthorized access. Error handling helps prevent information disclosure and maintains system stability. Encapsulation protects financial data by controlling access to sensitive information. Architecture and code reviews provide comprehensive assessments to identify and mitigate potential security vulnerabilities across the application's various components, ensuring the overall protection of the company's digital infrastructure.

**3. Manual Review**

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

The manual code review revealed multiple critical security vulnerabilities in Artemis Financials’ application. The database connection in DocData.java exposes hardcoded root credentials with default username and password, creating an extreme security risk that could allow direct database access by attackers. Input validation is severely lacking across multiple controllers, particularly in the CRUDController, which accepts business name parameters without any sanitization or checking, potentially enabling injection attacks. The customer class demonstrates poor encapsulation with an unprotected account balance variable, allowing direct manipulation of financial data, and the deposit method lacks critical validation to prevent negative or fraudulent transactions. Unimplemented methods in classes like myDateTime create potential functionality gaps that could be exploited, while error handling is minimal, with catch blocks using generic printStackTrace() that might leak sensitive system information. API interactions are not securely implemented, with controllers exposing internal object details and using ineffective data transfer methods that could reveal system internals. These interconnected vulnerabilities create multiple potential attack vectors that could compromise the entire financial application's security and data integrity.

**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 identifies several known vulnerabilities in Artemis Financials’ software application. One of the critical issues is **CVE-2024-21733**, which affects Apache Tomcat. This vulnerability involves the generation of error messages containing sensitive information, potentially exposing internal system details to attackers. The recommended solution is to upgrade Apache Tomcat to version 8.5.64+ or 9.0.44+. This issue has been documented in Apache’s official security advisories, highlighting its risk in web application environments. Another severe vulnerability is **CVE-2022-22965**, known as Spring4Shell, which affects the Spring Framework. This is a remote code execution (RCE) vulnerability that allows attackers to execute arbitrary code on the server if specific configurations are met. To mitigate this, it is recommended to upgrade to the latest patched version of Spring Framework. This vulnerability gained widespread attention when security researchers discovered its exploitability in systems using outdated Spring versions, prompting urgent updates from software vendors. Additionally, **CVE-2023-46589** is another Apache Tomcat vulnerability that can lead to out-of-memory errors during the TLS handshake process. Attackers can exploit this to cause a denial of service (DoS) by overwhelming the server. The fix for this issue is to upgrade Apache Tomcat to version 11.0.0-M21, 10.1.25, or 9.0.90. Security researchers first documented this issue by analyzing how excessive resource consumption in the TLS handshake process could crash servers.

Lastly, **CVE-2020-8022** is a privilege escalation vulnerability affecting Apache Tomcat on SUSE Linux. This issue arises due to incorrect default permissions, allowing attackers with local access to escalate privileges from the tomcat group to root. To address this, it is advised to review and restrict Tomcat’s permission settings and upgrade to a secure version. This vulnerability was first identified through penetration testing reports, demonstrating how attackers could exploit file permission misconfigurations to gain elevated access.

**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 security vulnerabilities identified in Artemis Financial’s software application, several mitigation strategies must be implemented. For **CVE-2024-21733 (Apache Tomcat error message disclosure)**, the solution is to upgrade Apache Tomcat to version 8.5.64+ or 9.0.44+. Additionally, the application should be configured to suppress detailed error messages in production environments, ensuring that sensitive system details are not exposed to potential attackers. To mitigate **CVE-2022-22965 (Spring Framework remote code execution vulnerability)**, the application must be updated to the latest patched version of the Spring Framework. Developers should also review their application configurations to ensure that outdated or vulnerable components are not in use. Implementing strict input validation and sandboxing execution environments can further prevent remote code execution exploits. For **CVE-2023-46589 (Apache Tomcat TLS handshake DoS attack)**, the recommended mitigation step is to upgrade Apache Tomcat to version 11.0.0-M21, 10.1.25, or 9.0.90. Additionally, rate-limiting mechanisms should be enforced to prevent excessive TLS handshake requests, and monitoring systems should be configured to detect and block unusual traffic patterns that could indicate an ongoing denial-of-service attack. To resolve **CVE-2020-8022 (Apache Tomcat privilege escalation on SUSE Linux)**, it is necessary to review and adjust Tomcat’s file permissions to follow the principle of least privilege. System administrators should ensure that the Tomcat service runs with restricted permissions, preventing unauthorized privilege escalation. Regular audits of system permissions and user access should also be performed to identify and mitigate potential security gaps. Beyond these specific fixes, Artemis Financial should adopt a proactive security approach, including regular dependency checks, continuous security monitoring, and penetration testing. Implementing automated security updates and integrating secure coding practices into the development lifecycle will further reduce the risk of future vulnerabilities.