# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **07/28/2025** | **Anjel Cobbs** |  |

## 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

Anjel Cobbs

**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 is entrusted with the personal and sensitive information of its customers, including Social Security numbers, bank account details and balances, home addresses, and other confidential data. It is in the company’s best interest to integrate robust security policies with its communication systems to prevent data breaches, build client trust, and mitigate potential legal consequences.

Although Artemis Financial's company description does not mention processing international transactions, it also does not explicitly define its services as domestic. Therefore, we can reasonably assume that international transactions may occur.

Local, federal, and international cybersecurity policies must all be considered when implementing secure communication systems. These policies include the data protection laws and regulations that secure communications must comply with. Artemis Financial’s security system should adhere to the IRS Publication 1075 guidelines to protect customers’ private tax information. It should also comply with the Payment Card Industry (PCI) standards to protect customers’ credit card data.

As a financial institution, Artemis Financial is vulnerable to ransomware attacks, in which attackers encrypt sensitive data and demand payment for its release. The company also faces the threat of phishing attacks, where malware is used to trick users into disclosing sensitive or financial information. Additionally, there is the broader threat of hackers exploiting system vulnerabilities to gain unauthorized access to critical data.

To support Artemis Financial in its modernization efforts, incorporating well-established open-source libraries can significantly reduce development time and costs while enhancing code quality. However, it is essential to regularly test and maintain these libraries to avoid introducing security vulnerabilities. Integrating modern web technologies into the system's design can enhance performance, user experience, and maintainability. Adopting current coding practices, frameworks, and programming languages can strengthen the system’s overall security if they are implemented securely by experienced developers.

**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*

User input should be validated to prevent code injections and cross-site Scripting (XSS), which can compromise the software. All user input should be validated against a whitelist before accessing the application.

*Cryptography*

Encryption is essential to protect sensitive data as it's stored and sent over the network. This way, the data is unreadable without authorized decryption.

*Client/Server*

Protecting the communication channels between the client and server is essential to prevent attackers from intercepting messages containing sensitive data.

*Code Quality*

Employing secure coding practices, such as running dependency checks on libraries and frequent manual scans, helps to identify and prevent

vulnerabilities that attackers can exploit.

*Encapsulation*

Encapsulation promotes access control, ensuring only authorized access to certain system features, such as databases containing customers’ financial details.

**3. Manual Review**

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

[Include your 7–10 findings here.]

* Customer class
  + *account\_balance* is a public variable, so it is vulnerable to unauthorized access/modification
  + *deposit(int a)* method does not validate the input parameter
  + no access control for the *showInfo()* method; sensitive data could be leaked
* DocData class
  + The parameters *key* and *value* are not validated
  + Error pages should not reveal sensitive data
  + Username and password are stored directly in the code (unauthorized access to sensitive data)

**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** | **Solution** |
| CVE-2022-22965 | Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. | Apply updates per vendor instructions. |
| CVE-2020-1938 | Apache Tomcat trusts AJP connections more than similar (HTTP), making them exploitable if accessible to attackers. | Upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. |
| CVE-2025-24813 | Apache Tomcat contains a path equivalence vulnerability that allows a remote attacker to execute code, disclose information, or inject malicious content via a partial PUT request. | Apply mitigations per vendor instructions, follow applicable BOD 22-01 guidance for cloud services, or discontinue use of the product if mitigations are unavailable.  Upgrade to version 11.0.3, 10.1.35 or 9.0.99 |
| CVE-2023-44487 | HTTP/2 contains a rapid reset vulnerability that allows for a distributed denial-of-service attack (DDoS). | Apply mitigations per vendor instructions, follow applicable BOD 22-01 guidance for cloud services, or discontinue use of the product if mitigations are unavailable. |

**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.

Security should be a priority throughout the entire software development lifecycle—from initial design to final release. Regular testing and code reviews are essential for identifying and addressing vulnerabilities early. Developers should stay informed about current secure coding practices to enhance code quality and minimize security risks.

A manual review of the codebase revealed several vulnerabilities. Notably, user input is not being properly validated, and the system lacks adequate access control mechanisms. To mitigate risks such as cross-site scripting (XSS), code injection, and denial-of-service (DoS) attacks, all user input should be validated using a strict whitelisting approach. Additionally, proper authorization checks must be implemented to restrict access to sensitive data and system functions.

The dependency-check report identified several critical-severity vulnerabilities caused by outdated third-party libraries and dependencies. All identified vulnerable libraries should be upgraded to their latest secure version, prioritizing the most vulnerable cases. We should continue to run dependency checks throughout development to ensure the third-party libraries are up to date and secure.

References

*Cybersecurity Law and Regulation for Local Government*. (2022). https://doi.org/10.1002/9781119788317.ch10