# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **02/01/2025** | **Hunter Lucas** |  |

## 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

Hunter Lucas

**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?
  1. Secure communications are vital as they protect sensitive financial data, maintain client trust, and ensure compliance with industry standards. This secure exchange of information underpins the company’s commitment to safeguarding financial plans and client details, reinforcing its reputation for reliability and security.
* Are there any international transactions that the company produces?
  1. The potential for international transactions means that additional security measures are necessary. When doing business on a global scale, the company must account for different regulatory environments and ensure that all cross-border communications are encrypted and secure.
* Are there governmental restrictions on secure communications to consider?
  1. There will be many restrictions to consider. Compliance with data protection laws and regulatory guidelines is non-negotiable, as these rules dictate strict standards for data transmission and storage. By adhering to these requirements, we can mitigate legal risks and demonstrate its commitment to robust security practices.
* What external threats might be present now and in the immediate future?
  1. External threats will pose a constant threat, stemming from the issues we face now. With cyber threats evolving rapidly, the company must remain vigilant and update its security measures regularly. Proactive identification and mitigation of these vulnerabilities are essential for protecting both client information and the overall integrity of the financial planning process.
* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?
  1. The new requirements for us will involve integrating new web technologies while managing the risks associated with open-source libraries. Open-source components can enhance functionality and speed up development, but they also require careful oversight to avoid introducing security vulnerabilities. While we must strive for innovation, we must be careful to consider a secure implementation of any modern web technology.

**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 is essential for Artemis Financial’s web application, as it ensures that all user inputs, whether submitted through client forms or received via API requests, are properly sanitized and validated THis mitigates risks like injection attacks and data breaches that could compromise sensitive financial information. Securing the RESTful API endpoints is crucial since these interfaces manage the exchange of confidential data. Implementing strict authentication and access control measures helps prevent unauthorized access and potential data tampering. Cryptography plays a key role in protecting data integrity by encrypting information both during transmission and when stored, which is particularly important for international transactions that must comply with regulatory standards. Lastly, establishing secure client/server communications is vital for preventing man-in-the-middle attacks and ensuring that data flows safely between endpoints, maintaining the overall security of the system and preserving trust.

**3. Manual Review**

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

**Input Validation:**Found in: CRUDController.java  
Why it is a concern: The /read endpoint accepts user input without validation. This leaves our service vulnerable to attacks like injection attacks. Malicious input could be used to manipulate database queries or execute unauthorized scripts.

**Hardcoded Credentials for the Database:**

Found in: DocData.java

Why it is a concern: Hardcoded credentials expose sensitive authentication details in the source code. If an attacker gains access, they could use these credentials to access the database.

**Stack Trace Printing on Error Handling:**Found in: DocData.java

Why it is a concern: When there is an error in the database, the full stack trace is printed. This will expose the entire system and database logs, which will allow an attacker to gain insight into other vulnerabilities.

**Lack of Encapsulation in Customer.java**Found in: customer.java

Why it is a concern: The account balance field is declared with default access, meaning any class in the package can modify it directly. This can lead to the changing of the financial records.

**Incomplete Methods**

Found in: myDateTime.java

Why it is a concern: The incomplete functions found in the file cause a concern for code quality and maintainability. They can cause unexpected behavior if not addressed as the codebase grows.

**Exposed Data:**

Found in: CRUDController.java

Why it is a concern: The /read API endpoint returns a CRUD object with data retrieved from DocData. If it contains sensitive data, this data could be exposed to unauthorized users.

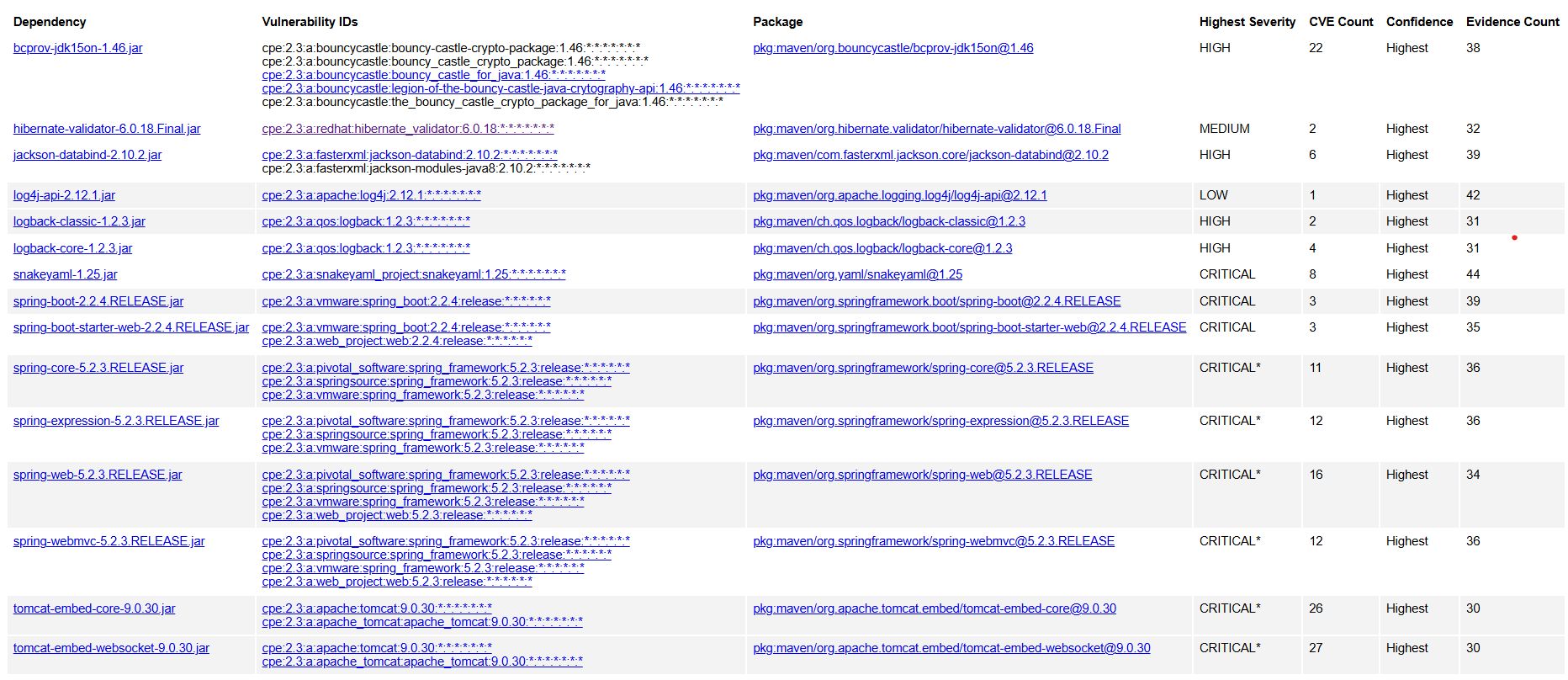
**Encrypting Database Connection:**Found in: DocData.java

Why it is a concern: The database connection does not use encryption. Sensitive financial data transmitted between the application and the database could be intercepted by attackers.

**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



Code: cpe:2.3:a:bouncycastle:bouncycastle\_crypto\_package:1.46  
Description and Recommendation: Cryptographic vulnerability, update to the latest version of Bouncy Castle.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:redhat:hibernate\_validator:6.0.18  
Description and Recommendation: Validation framework vulnerability, upgrade Hibernate Validator to a newer version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:fasterxml:jackson-databind:2.10.2  
Description and Recommendation: Deserialization vulnerability, update Jackson Databind to the latest secure version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:apache:log4j-api:2.12.1  
Description and Recommendation: Logging vulnerability, update to a more secure version of Log4j.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:qos.logback:logback-classic:1.2.3  
Description and Recommendation: Logging vulnerability, upgrade Logback Classic to the latest version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:qos.logback:logback-core:1.2.3  
Description and Recommendation: Core logging library vulnerability, update Logback Core to a more secure version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25  
Description and Recommendation: YAML parsing vulnerability, upgrade SnakeYAML to the latest version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:vmware:spring\_boot:2.2.4:release  
Description and Recommendation: Spring Boot core vulnerability, upgrade to a secure version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:web\_project:web:2.2.4:release  
Description and Recommendation: Web framework vulnerability, update Spring Boot Starter Web to the latest version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release  
Description and Recommendation: Framework vulnerability, update Spring Core to the latest secure version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:springsource:spring\_framework:5.2.3:release  
Description and Recommendation: Expression vulnerability, upgrade Spring Expression to a secure version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:springsource:spring\_framework:5.2.3:release  
Description and Recommendation: Web framework vulnerability, update Spring Web to the latest secure version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:springsource:spring\_framework:5.2.3:release  
Description and Recommendation: MVC vulnerability, upgrade Spring WebMVC to a secure version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:apache:tomcat:9.0.30  
Description and Recommendation: Core embedding vulnerability, update Tomcat Embed Core to the latest version.  
Attribution: Documented in CVE advisories.

Code: cpe:2.3:a:apache:tomcat:9.0.30  
Description and Recommendation: Websocket embedding vulnerability, upgrade Tomcat Embed Websocket to a more secure version.  
Attribution: Documented in CVE advisories.

**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.

The manual review uncovered several critical vulnerabilities within the codebase. In CRUDController.java, the /read endpoint accepts unvalidated user input, exposing the application to injection attacks. DocData.java contains hardcoded database credentials and prints full stack traces on errors, both of which risk exposing sensitive information. In Customer.java, improper encapsulation of the account balance field permits unauthorized modifications, and myDateTime.java includes incomplete methods that could lead to unexpected behavior. The /read endpoint may expose sensitive data retrieved from DocData, and the unencrypted database connection further heightens the risk of data interception.

Static testing via dependency-check identified several known vulnerabilities in third-party libraries. For example, Bouncy Castle Crypto Package suffers from cryptographic issues, while Hibernate Validator and Jackson Databind have input validation and deserialization vulnerabilities, respectively; both require updates. Log4j and Logback Classic are affected by logging issues. Critical vulnerabilities were also found in various Spring Boot components (Spring Boot Starter Web, Spring Core, Spring Expression, Spring Web, and Spring WebMVC) and Apache Tomcat, all of which should be updated to patched versions as documented in CVE advisories and records.

Steps to Mitigate:

Input Validation:

* Implement rigorous input validation for the /read endpoint in CRUDController.java to prevent injection attacks.

Database:

* Remove hardcoded database credentials from DocData.java and store them securely using environment variables or a secure configuration management system.

Error Handling:

* Modify error-handling procedures in DocData.java to avoid printing full stack traces to the user, instead logging them securely.

Encapsulation:

* Update Customer.java by declaring sensitive fields as private and providing controlled access via getter and setter methods.

Code Quality:

* Complete or remove incomplete methods in myDateTime.java to prevent unexpected behavior and improve maintainability.

Data Exposure:

* Review and sanitize data returned by the /read endpoint to ensure that sensitive information from DocData.java is not exposed.

Encryption:

* Implement encryption for the database connection to secure the transmission of sensitive financial data.

Third-Party Dependency Updates:

* Update the following libraries to their latest secure versions:
  + Bouncy Castle Crypto Package
  + Hibernate Validator
  + Jackson Databind
  + Log4j Logback Classic
  + Spring Boot Components (Spring Boot Starter Web, Spring Core, Spring Expression, Spring Web, Spring WebMVC)
  + LogBack Core
  + SnakeYAML
  + Apache Tomcat