# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11-17-2024** | **Xander O’Hara** | **First Draft** |

## 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

Xander O’Hara

**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’s software needs to have strong security to protect sensitive financial data. And the main points needed are:

1. **Secure Communications**:  
   Secure protocols like SSL/TLS are needed to keep data safe during transmission.
2. **International Transactions**:  
   Since they operate globally, their encryption and storage methods must meet regulations like GDPR, HIPAA, and PCI DSS.
3. **Government Restrictions**:  
   Encryption standards need to comply with government rules, like U.S. export controls on encryption.
4. **External Threats**:  
   Current risks include phishing, ransomware, and SQL injection. Future threats might involve AI-driven or automated attacks targeting open-source software.
5. **Modernization Requirements**:  
   They need secure, updated tools and frameworks to keep up with advancing technology, like APIs, microservices, and containerized systems.

**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.

Based on the vulnerability assessment flow, these are the following areas that are needed for Artemis Financial’s app:

1. **Input Validation**:  
   Prevents attacks like SQL injection and cross-site scripting (XSS), which are common in applications handling user inputs.
2. **APIs**:  
   Secure APIs are important for safe communication between components without exposing sensitive data.
3. **Cryptography**:  
   Protects data both when it’s sent and stored.
4. **Client/Server**:  
   Protecting communication between distributed systems avoids issues like man-in-the-middle attacks.
5. **Code Quality**:  
   Writing clean and reliable code reduces errors and security gaps.
6. **Encapsulation**:  
   Protecting internal data structures and functions from unauthorized access keeps the application secure.

**3. Manual Review**

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

**SQL Injection Risk in read\_document Method (docdata.java):**

* **Problem:** The method does not sanitize inputs, leaving it vulnerable to SQL injection.
* **Solution:** Use prepared statements with parameterized queries to mitigate this risk.

**Hardcoded Database Credentials (docdata.java):**

* **Problem:** The Connection object contains hardcoded credentials ("root", "root"), which could be exploited.
* **Solution:** Store credentials securely in environment variables or configuration files.

**Improper Error Handling in read\_document Method (docdata.java):**

* **Problem:** The catch block only prints the stack trace, which is insufficient for error handling.
* **Solution:** Log errors securely using a logging framework like Logback or SLF4J and avoid exposing stack traces in production.

**Insufficient Input Validation in CRUDController (CRUDController.java):**

* **Problem:** The @RequestParam parameter lacks validation, which could lead to injection attacks or processing invalid data.
* **Solution:** Validate and sanitize all user inputs on the server side.

**Lack of Access Modifiers in Class Fields (customer.java):**

* **Problem:** The account\_balance field is not marked as private, allowing unintended access or modification.
* **Solution:** Use private or protected access modifiers to restrict access to internal class fields.

**Unnecessary Exposure of Data in CRUDController (CRUDController.java):**

* **Problem:** The CRUD method exposes doc.toString() directly, which could leak sensitive information.
* **Solution:** Filter and return only necessary and safe data fields.

**No Validation in setMyDateTime Method (myDateTime.java):**

* **Problem:** The method accepts time values without ensuring they are within valid ranges (e.g., seconds and minutes must be 0–59).
* **Solution:** Add validation logic to check that the input values are valid before setting them.

**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 IDs** | **Description** | **Solution** |
| --- | --- | --- | --- |
| bcprov-jdk15on-1.46.jar | [cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Abouncycastle&cpe_product=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api&cpe_version=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api%3A1.46) | The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7. | Upgrade to the latest stable version of bcprov-jdk15on to address known vulnerabilities. |
| spring-boot-2.2.4.RELEASE.jar | [cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_boot&cpe_version=cpe%3A%2F%3Avmware%3Aspring_boot%3A2.2.4) | Spring Boot (framework for Java applications.) | Update to a newer version of Spring Boot, preferably 2.4.0 or later, where vulnerabilities are fixed. |
| logback-core-1.2.3.jar | [cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aqos&cpe_product=cpe%3A%2F%3Aqos%3Alogback&cpe_version=cpe%3A%2F%3Aqos%3Alogback%3A1.2.3) | logback-core module (Core module of Logback, a Java logging framework.) | Update to version 1.2.9 or newer to mitigate existing vulnerabilities. |
| log4j-api-2.12.1.jar | [cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Alog4j&cpe_version=cpe%3A%2F%3Aapache%3Alog4j%3A2.12.1) | The Apache Log4j API (Logging framework commonly used in Java applications.) | Upgrade to Log4j 2.17.1 or later to fix critical security issues like Log4Shell. |
| snakeyaml-1.25.jar | [cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Asnakeyaml_project&cpe_product=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml&cpe_version=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml%3A1.25)  [cpe:2.3:a:yaml\_project:yaml:1.25:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Ayaml_project&cpe_product=cpe%3A%2F%3Ayaml_project%3Ayaml&cpe_version=cpe%3A%2F%3Ayaml_project%3Ayaml%3A1.25) | YAML 1.1 parser and emitter for Java | Use version 1.30 or later, which resolves critical vulnerabilities in older versions. |
| jackson-databind-2.10.2.jar | [cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Afasterxml&cpe_product=cpe%3A%2F%3Afasterxml%3Ajackson-databind&cpe_version=cpe%3A%2F%3Afasterxml%3Ajackson-databind%3A2.10.2) | General data-binding functionality for Jackson: works on core streaming API | Update to version 2.13.4 or newer to fix deserialization vulnerabilities. |
| tomcat-embed-core-9.0.30.jar | [cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Atomcat&cpe_version=cpe%3A%2F%3Aapache%3Atomcat%3A9.0.30)  [cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_tomcat&cpe_product=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat&cpe_version=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat%3A9.0.30) | Core Tomcat implementation (Core implementation of Apache Tomcat, a Java web server.) | Upgrade to Tomcat 9.0.65 or newer, as newer versions patch critical vulnerabilities. |
| hibernate-validator-6.0.18.Final.jar | [cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aredhat&cpe_product=cpe%3A%2F%3Aredhat%3Ahibernate_validator&cpe_version=cpe%3A%2F%3Aredhat%3Ahibernate_validator%3A6.0.18) | Hibernate's Bean Validation (JSR-380) reference implementation. | Upgrade to version 6.1.5.Final or later to address security vulnerabilities. |
| spring-web-5.2.3.RELEASE.jar | [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3)  [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) | Spring Web (module for web development.) | Update to Spring Web 5.3.25 or newer to mitigate identified vulnerabilities. |
| spring-beans-5.2.3.RELEASE.jar | [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3)  [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) | Spring Beans (Dependency injection and bean configuration framework.) | Upgrade to Spring Beans 5.3.25 or later to resolve existing vulnerabilities. |
| spring-webmvc-5.2.3.RELEASE.jar | [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3)  [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) | Spring Web MVC (module for Spring framework.( | Update to Spring Web MVC 5.3.25 or newer to resolve security issues. |
| spring-context-5.2.3.RELEASE.jar | [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3)  [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) | Spring Context (Application context and dependency injection module for Spring.) | Use version 5.3.25 or newer to address potential vulnerabilities. |
| spring-expression-5.2.3.RELEASE.jar | [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3)  [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) | Spring Expression Language (SpEL) (for Spring framework.) | Upgrade to Spring Expression 5.3.25 or later for enhanced security. |

**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.

#### Steps for Fixing Issues:

1. **Input Validation**:  
   Add server-side validation for all inputs to block malicious payloads. Libraries like OWASP ESAPI can help sanitize data.
2. **Update Dependencies**:  
   Upgrade outdated libraries to newer, secure versions. Use tools like Dependabot to track future updates.
3. **Improve Session Management**:  
   Use secure tokens, regenerate them frequently, and set strict expiration rules. Tools like JWT can simplify this.
4. **Use Encryption**:  
   Secure sensitive data with AES-256 encryption and use TLS 1.3 for communications.
5. **Authentication and Authorization**:  
   Add multi-factor authentication (MFA) and limit access using role-based access control (RBAC).
6. **Better Logging**:  
   Remove sensitive information from logs and monitor for suspicious activity in real time.
7. **Testing**:  
   Perform regular code reviews and static analysis to catch issues before deployment.