# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/17/2025** | **Brooke Slampak** | **Assessment** |

## 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

Brooke Slampak

**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?

A comprehensive security strategy is crucial to safeguard Artemis’ customers, reputation, and long-term success. Considering that Artemis is a financial services provider, we must prioritize security, as the nature of the business will most certainly attract cyber-attackers. In addition to being adequately secure, service provided to clientele should be discreet with Personally Identifiable Information (PII) handled with utmost care. Online operations subject Artemis to a range of cyber threats, from minor attacks like denial-of-service (DoS) attacks to severe breaches resulting in financial loss and irreparable damage to the company’s reputation. High-level encryption should be implemented across all databases, as even limited personal information exposure can open the door to possible phishing attacks. In addition, international operations expose Artemis to a wide array of regulations surrounding PII, records retention, contract enforcement, and secure communication protocols. Knowledgeability in these areas in the countries in which Artemis operates is required.

Additionally, Artemis must carefully manage its use of open-source software, as certain licenses require reciprocity, meaning that any modifications to their code and derivative software that implements it must be shared under the same open-source license. This typically means making the source code publicly accessible, ensuring that others can view, modify, and distribute it as well. Open-source transparency could expose vulnerabilities if weaknesses are uncovered, making frequent monitoring for updates to any third-party libraries a necessity, whether open-source or proprietary.

**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.

Artemis Financial has a RESTful web API, therefore focusing on input validation, secure API interactions, code errors, encapsulation, and cryptography should be prioritized. Input validation is crucial to sanitize user data, preventing inputs and injections that could cause errant system behavior. Securing API connections further guarantees the authenticity of users and data exchanges, protecting against unauthorized access. While Spring offers some error-handling mechanisms, it’s essential to implement generic error messages to avoid revealing system details that could give insights about a program and aid attackers. Additionally, applying encapsulation by keeping sensitive variables private and accessing them through getter and setter methods protects data from unauthorized modifications. Another critical aspect is cryptography, ensuring that all data in transit and at rest is encrypted. Implementing encryption enhances data protection across databases and communication channels.

**3. Manual Review**

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

* In the customer class, account\_balance is public. It should be private, with getter and setter methods to control access.
* The pom.xml file requires updating to a newer version of Maven, as older versions may not include all CVEs contained within and these vulnerabilities will go unaddressed.
* The parameters of the Greeting class are private, but GreetingController doesn’t utilize the getter and setter functions of the Greeting class.
* Methods such as the greeting method should sanitize inputs and check their length to prevent issues like buffer overruns.
* Only a few classes and methods check for errors. Error messages should avoid giving away too much information, such as “incorrect password” or “out of bounds,” as this can help attackers find vulnerabilities. Messages need to be reviewed and properly sanitized to avoid leaking details that can be exploited
* The myDateTime and DocData classes have incomplete methods.
* There is a lack of API sanitization that leaves the program vulnerable to the risk of injection, such as the CRUDcontroller class’s CRUD method.

**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 Code | Description | Mitigation |
| bcprov-jdk15on-1.46.jar | cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:  cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46  cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytographyapi:1.46  cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46 | The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms | Upgrade Bouncy castle to version 1.60 or higher |
| tomcat-embed-websocket-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30 | Core Tomcat implementation | Update Tomcat to the newest version |
| tomcat-embed-core-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30 | Core Tomcat implementation | Upgrade to newest Tomcat version |
| spring-core-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release  cpe:2.3:a:springsource:spring\_framework:5.2.3:release  cpe:2.3:a:vmware:spring\_framework:5.2.3:release  cpe:2.3:a:vmware:springsource\_spring\_framework:5.2.3:release | Spring Core | Upgrade to the latest version of Spring |
| logback-classic-1.2.3.jar | cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\* | logback-classic module | Upgrade to logback versions 1.3.14/.4.14 or later |
| spring-boot-2.2.4.RELEASE.jar | cpe:2.3:a:vmware:spring\_boot:2.2.4:release | Spring Boot | Only affects unsupported versions. Upgrade. |
| snakeyaml-1.25.jar | cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25 | YAML 1.1 parser and emitter for Java | Allows entity expansion. Upgrade. |
| jackson-databind-2.10.2.jar | cpe:2.3:a:fasterxml:jackson-databind:2.10.2  cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2 | General data-binding functionality for Jackson: works on core streaming API | Data integrity concerns. Upgrade to latest release. |
| logback-core-1.2.3.jar | cpe:2.3:a:qos:logback:1.2.3 | logback-core module | Arbitrary code execution. Upgrade to the latest version |
| hibernate-validator-6.0.18.Final.jar | cpe:2.3:a:redhat:hibernate\_validator:6.0.18 | Hibernate's Bean Validation (JSR-380) reference implementation | Input validation bypass issue. Upgrade to the latest version. |
| log4j-api-2.12.1.jar | cpe:2.3:a:apache:log4j:2.12.1 | The Apache Log4j API | Allows Man-in-the-middle attack on SMTP. Upgrade. |
| spring-boot-starter-web-2.2.4.RELEASE.jar | cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\* cpe:2.3:a:web\_project:web:2.2.4:release:\*:\*:\*:\*:\*:\* | Starter for building web, including RESTful, applications using Spring | Upgrade to the latest version of Spring |
| spring-expression-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | Spring Expression Language (SpEL) | Upgrade to the latest version of Spring |
| spring-web-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\* | Spring Web | Upgrade to the latest version of Spring |
| spring-webmvc-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\* | Spring Web MVC | Upgrade to the latest version of Spring |

**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.

* Maven dependency check version should be updated to the current version.
* Implement sanitized error messages and input validation.
* Complete incomplete methods.
* Implement encapsulation practices by fully utilizing getter and setter methods.