# CS 305 Project One Template

**Document Revision History**

| Version | Date | Author | Comments |
|---|---|---|---|
| 1.0 | NOV 17, 2024 | Jerome Talanquines | |

**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**
Jerome Talanquines

**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?

[Include your findings here.]
Artemis Financial has expressed the need for optimizing and implementing effective software security measures to protect their web-based software application. As a developer assisting this consulting company, that deals with financial plans, they handle sensitive customer information related to savings, retirement, investments, and insurance. Therefore, the security of their application holds value. The need to protect their organization from external threats ensures secure communications. Although no specific information is provided about international transactions or governmental restrictions, it is vital to consider these factors while in design for security measures. The modernization requirements should also be cared for, including the role of open-source libraries and evolving web app technologies.

**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.

[Include your findings here.]
Based on my findings and evaluation of the process flow diagram, the following areas proving it's relevancy:
Data Protection : Sensitive data,  such as customers private information, should properly be encrypted during both transit and at rest in order to prevent unauthorized access and data breaches.
Input Validation : Secure input and representations. Proper input validation implanted prevents common vulnerabilities like SQL injection and cross-site scripting (XSS). Output encoding should be applied to projects against these type of attacks when displaying user-generated content.
Session Management : The application should properly manage sessions when addressing client / server properties. Sessions should be maintained and have the ability to prevent session related attacks such as session fixation or hijacking.
Secure Communications : The application should utilize secure communication protocols such as HTTPS to protect data transmission between client and servers, ensuring user experience is satisfactory. Meanwhile, it is a good addition to add the proper encapsulation should be utilized to prevent error.

**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.]
Inspecting the code base's source files and the java files that It contains, I can address several vulnerabilities that may propose issue.
The following are:
Lack of Input Validation
Missing Access Modifiers
Error Handling
Moreover, the code base is not built to be protected from issues such as:
Injection Attacks
Data Exposure Sensitivity
Broken Access Control
Cross Site Scripting
Cross Site Request Forgery
Insecure Deserialization
Security Misconfigurations
Lack of rate limiting and resource throttling
API Key and Token Exposure
Unvalidated Redirects and Forwards
Insufficient logging and monitoring

**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

[Include your findings here.]

| Dependency | Vulnerability IDs | Package | Highest Severity | CVE Count | Confidence | Evidence Count |
|---|---|---|---|---|---|---|
| bcprov-jdk15on The name of the dependency | cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:*:*:*:*:*:*:* | pkg:maven/org.bouncycastle/bcprov-jdk15on@1.46 | HIGH | 20 | Highest | 37 |
| spring-boot-2.2.4.RELEASE.jar | cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:*:* | pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE | CRITICAL | 3 | Highest | 32 |
| logback-core-1.2.3.jar | cpe:2.3:a:qos:logback:1.2.3:*:*:*:*:*:*:* | pkg:maven/ch.qos.logback/logback-core@1.2.3 | HIGH | 2 | Highest | 32 |
| log4j-api-2.12.1.jar | cpe:2.3:a:apache:log4j:2.12.1:*:*:*:*:*:*:* | pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1 | CRITICAL | 5 | Highest | 46 |
| snakeyaml-1.25.jar | cpe:2.3:a:snakeyaml_project:snakeyaml:1.25:*:*:*:*:*:*:*<br>cpe:2.3:a:yaml_project:yaml:1.25:*:*:*:*:*:*:* | pkg:maven/org.yaml/snakeyaml@1.25 | CRITICAL | 10 | Highest | 28 |
| jackson-databind-2.10.2.jar | cpe:2.3:a:fasterxml:jackson-databind:2.10.2:*:*:*:*:*:*:* | pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2 | HIGH | 6 | Highest | 39 |
| tomcat-embed-core-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:*<br>cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.30:*:*:*:*:*:*:* | pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30 | CRITICAL | 27 | Highest | 39 |
| hibernate-validator-6.0.18.Final.jar | cpe:2.3:a:redhat:hibernate_validator:6.0.18:*:*:*:*:*:*:* | pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final | MEDIUM | 2 | Highest | 36 |
| spring-web-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:*:*<br>cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:*:* | pkg:maven/org.springframework/spring-web@5.2.3.RELEASE | HIGH | 8 | Highest | 28 |
| spring-beans-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:*:*<br>cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:*:* | pkg:maven/org.springframework/spring-beans@5.2.3.RELEASE | HIGH | 1 | Highest | 28 |
| spring-webmvc-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:*:*<br>cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:*:* | pkg:maven/org.springframework/spring-webmvc@5.2.3.RELEASE | HIGH | 2 | Highest | 30 |
| spring-context-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:*:*<br>cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:*:* | pkg:maven/org.springframework/spring-context@5.2.3.RELEASE | MEDIUM | 1 | Highest | 28 |
| spring-expression-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:*:*<br>cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:*:* | pkg:maven/org.springframework/spring-expression@5.2.3.RELEASE | MEDIUM | 4 | Highest | 30 |

Hibernate Validators - Library performs bean validation in Java applications. This vulnerability allows attackers to exploit flaws in the library to gain control of the application. Ensure that hibernate validator is updated.
Jacksons Databind - Jackson Databind is a library that provides JSON serialization and deserialization capabilities for Java applications. This vulnerability allows attackers to exploit a flaw in the library to potentially gain control of the application. Ensure that everything is updated.
SnakeYAML - YAML parser and emitter library for Java applications. This vulnerability allows attackers to exploit flaws in the library to potentially gain control of the application. Ensure updating is at the latest version.
Logback-core - Logback is a logging framework for Java applications. This vulnerability allows attackers to exploit flaws in the library to potentially gain control of the application. Updating necessary.
Log4j-api - Apache Log4j is a logging framework for Java applications. This vulnerability allows attacker to exploit a flaw in the library to potentially gain control of the application. Updating necessary.

**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.

[Include your findings here.]
From both of my conducted tests, the steps to mitigate the identified security vulnerabilities, it would be best to adhere to the best practices. Such as using secure libraries, and implementing robust validation and encryption mechanisms. Keep in mind that they are several ways that your application can be compromised so it would be best to understand which types of attacks are most common and how it can be handled. In question 3, I have addressed several types of threats. Mitigation of these attacks can be assisted by knowledge of your resources that help prevent and optimize your framework. The frameworks that are contained in this application and their functionalities are the best way to address the codes vulnerability.