# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **05/22/2024** | **Mitchell Fontaine** | **First Release** |

## 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

Mitchell Fontaine

**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 a financial services provider which means they have access to many forms of sensitive PII. This means secure communication is of utmost importance. There was no mention of international transactions in the scenario, so I assume that there aren’t any that the company produces. There are a few government restrictions on secure communication for Artemis to consider since they are operating in the USA. They should review the SEC, FTC, NIST, and the Gramm-Leach-Bliley Act for any relevant restrictions they need to follow. If Artemis does operate internationally, then it would also be subject to European and other countries’ specific restrictions. Some external threats that might be present now or in the immediate future for Artemis are DDoS attacks, phishing attempts, ransomware attacks, vulnerability exploitation attempts, insider threats, and potentially nation-state attacks in the future due to the ever-changing landscape of cyber-war and the impact of finances on a country’s population. Artemis must take into consideration many things when modernizing such as the cost of upgrading technologies and paying for licenses. Open-source libraries can help reduce the cost of developing their own but there is always the potential threat of someone exploiting the libraries since they are available to everyone. Artemis should also keep their staff updated on new technologies coming out so they don’t fall behind like quantum computing and the use of LLM’s in their web applications.

**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.

Since Artemis is using a RESTful API, the areas of security from the vulnerability assessment process flow diagram that apply to them are Input Validation, Secure API Interactions, Code Error, and Encapsulation. Input Validation applies because the application is taking user input which will have to be sanitized to prevent harm if the user has malicious intent. Secure API Interactions are important to make sure the connection to the user is secure. Code Error applies because if there is an error in the code and it is not handled securely, it could lead to even more vulnerabilities. Encapsulation also applies because there are methods used in the code which need to be protected from any unwanted changes.

**3. Manual Review**

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

Upon inspecting the code base, I noticed a few potential vulnerabilities. The first is the lack of input validation. There is also no sanitation of user input which could lead to buffer overflows. There are also very few classes and methods with error handling. This should be utilized throughout the code base. There is also no POST method to secure the data. Another issue is the many public classes that should be private and use get/set functions. Lastly, there is the issue of user input to the API not being sanitized which can lead to injection-based attacks.

**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

There were thirteen vulnerable dependencies found during the dependency check:

bcprov-jdk15on-1.46.jar

Description: 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.

Identifiers: pkg:maven/org.bouncycastle/bcprov-jdk15on@1.46

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:bouncy\_castle\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

hibernate-validator-6.0.18.Final.jar

Description: Hibernate's Bean Validation (JSR-380) reference implementation.

Identifiers: pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final

cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

jackson-databind-2.10.2.jar

Description: General data-binding functionality for Jackson: works on core streaming API

Identifiers: pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2

cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

log4j-api-2.12.1.jar

Description: The Apache Log4j API

Identifiers: pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1

cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

logback-core-1.2.3.jar

Description: logback-core module

Identifiers: pkg:maven/ch.qos.logback/logback-core@1.2.3

cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

snakeyaml-1.25.jar

Description: YAML 1.1 parser and emitter for Java

Identifiers: pkg:maven/org.yaml/snakeyaml@1.25

cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

spring-boot-2.2.4.RELEASE.jar

Description: Spring Boot

Identifiers: pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE

cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

spring-boot-starter-web-2.2.4.RELEASE.jar

Description: Starter for building web, including RESTful, applications using Spring

MVC. Uses Tomcat as the default embedded container

Identifiers: pkg:maven/org.springframework.boot/spring-boot-starter-web@2.2.4.RELEASE

cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*

cpe:2.3:a:web\_project:web:2.2.4:release:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

spring-core-5.2.3.RELEASE.jar

Description: Spring Core

Identifiers: pkg:maven/org.springframework/spring-core@5.2.3.RELEASE

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:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

spring-web-5.2.3.RELEASE.jar

Description: Spring Web

Identifiers: pkg:maven/org.springframework/spring-web@5.2.3.RELEASE

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:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

spring-webmvc-5.2.3.RELEASE.jar

Description: Spring Web MVC

Identifiers: pkg:maven/org.springframework/spring-webmvc@5.2.3.RELEASE

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:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

tomcat-embed-core-9.0.30.jar

Description: Core Tomcat implementation

Identifiers: pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30

cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

tomcat-embed-websocket-9.0.30.jar

Description: Core Tomcat implementation

Identifiers: pkg:maven/org.apache.tomcat.embed/tomcat-embed-websocket@9.0.30

cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

Mitigation: Upgrade to the latest version.

**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 first thing that needs to be done is to update the dependency libraries. We then need to refactor the code base to sanitize input and output error messages, validate user input, and make better use of encapsulation throughout. We can then recompile and check for any code errors.