# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/22/2025** | **Dhiraj Gurung** |  |

## 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

Dhiraj Gurung

**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:

1. What is the value of secure communications to the company?
2. Are there any international transactions that the company produces?
3. Are there governmental restrictions on secure communications to consider?
4. What external threats might be present now and in the immediate future?
5. What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

Artemis Financial is responsible for creating financial plans for its customers which is sensitive information, and something its customers would not want leaked. Security, therefore, must be a priority because if a customer’s financial plans and investments were to get leaked they may potentially lose their life savings. Artemis financial may have customers who live internationally or have accounts internationally and as a result they may have to add a secure way to handle international transactions as well. There are laws in place that dictate how much of a person’s information can be shared publicly and how much needs to be kept confidential. It’s important to be mindful of these laws and maintain confidentiality with customers when working together with them. As long as there is money involved there will always be thieves and scammers trying to steal the money. One example, of an external threat could be an injection attack like a SQL injection. Additionally, scammers may try to trick the elderly into giving up their money as well. There needs to be security protocols in place for both external attacks and suspicious account behavior. With applications and technology constantly evolving it’s imperative that the software is being regularly maintained and managed. This includes tasks like keeping the software up to date and fixing any security vulnerabilities that may present itself in the future.

**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:** This is one area of security that needs to be considered because SQL injection attacks are far too common. Software that isn’t coded to protect against these attacks will be easily hacked and customers will lose a lot of money.

**Code Error:** Even with thorough code reviews and following proper coding guidelines some mistakes and bugs could slip through in development. As a result, it is important to have a system in place that can safely handle these errors.

**APIs:** Since the Artemis Financial software will be using the RESTFUL API it is important to secure the connection between the API and the databases otherwise the software will be susceptible to common attacks that target the RESTFUL API.

**Code Quality:** It is important in any coding environment to follow the proper coding guidelines and best practices. However, it is even more essential in the case of Artemis Financial because it handles monetary transactions and those are high value targets for thieves and hackers. Incorporating good coding practices can help reduce the risk of common threats to the software.

**3. Manual Review**

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

|  |  |
| --- | --- |
| **Class** | **Vulnerability** |
| CRUDController.java | * Does not use proper input validation leaving it susceptible to injection attacks |
| GreetingController.java | * There are also no prepared statements used in this class and is also susceptible to injection attacks. |
| DocData.class | * The admin username and password are pretty basic and could easily be guessed |
| myDateTime.java | * Code is incomplete and could lead to issues during launch. |
| Customer.java | * The class uses public datatype to store account balance information which isn’t a secure way of handling this * Additionally, they also use a public void function to handle the deposit which should also be handled more securely |
| N/A | There is a general lack of encryption throughout the code. |
| N/A | There is also a lot of outdated software being used throughout like the java version |

**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 Code** | **Description** | **Attributions** |
| bcprov-jdk15on-1.46.jar | 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. | <https://ossindex.sonatype.org/vulnerability/CVE-2024-34447?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on> |
| hibernate-validator-6.0.18.Final.jar | Hibernate's Bean Validation (JSR-380) reference implementation. | <https://ossindex.sonatype.org/vulnerability/CVE-2023-1932?component-type=maven&component-name=org.hibernate.validator%2Fhibernate-validator> |
| jackson-databind-2.10.2.jar | General data-binding functionality for Jackson: works on core streaming API | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649> |
| log4j-api-2.12.1.jar | The Apache Log4j API | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488> |
| logback-classic-1.2.3.jar | logback-classic module | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-6378> |
| logback-core-1.2.3.jar | logback-core module | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-6378> |
| snakeyaml-1.25.jar | YAML 1.1 parser and emitter for Java | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-1471> |
| spring-boot-2.2.4.RELEASE.jar | Spring Boot | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20873> |
| spring-boot-starter-web-2.2.4.RELEASE.jar | Starter for building web, including RESTful, applications using SpringMVC. Uses Tomcat as the default embedded container | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20873> |
| spring-core-5.2.3.RELEASE.jar | Spring Core | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965> |
| spring-expression-5.2.3.RELEASE.jar | Spring Expression Language (SpEL) | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965> |
| spring-web-5.2.3.RELEASE.jar | Spring Web | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000027> |
| spring-webmvc-5.2.3.RELEASE.jar | Spring Web MVC | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965> |
| tomcat-embed-core-9.0.30.jar | Core Tomcat implementation | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938> |
| tomcat-embed-websocket-9.0.30.jar | Core Tomcat implementation | <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938> |

**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.

In order to ensure that the Artemis Financial software application is secure we must first update all the dependencies to their latest versions as soon as possible. This is because most of the vulnerabilities stem from the fact that we are using outdated software. Another thing we need to implement is the proper input validation to prevent SQL attacks.