# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **19JUL24** | **Benjamin cleary** |  |

## 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.

## Developer

Benjamin Cleary

**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 derives a major value from maintaining secure data storage and access to the financial data of their customers. That customer financial data is vital to their business process and what they offer to customers in terms of services. If the customers cannot trust that their private financial data is secure, they will not use Artemis Financial services. With any web based financial service, it must be assumed that international transactions occur and that international customers exist. With this comes the additional regulatory regimes associated with where the data and customers are stored and manipulated. Depending on the nation in question, requirements for secure communications, and any governmental backdoors, will differ. External threats will also keep evolving and must be guarded against. While zero day exploits cannot be anticipated, every effort must be made to maintain a robust security posture for data transfer, storage, and manipulation. Financial data is a prime target for data exfiltration, manipulation, and intercept. Even if the data is not used specifically against an individual account, that data can be used to facilitate identity theft or blackmail, both of which have impacts beyond just the transaction or data in question. While efforts by the software and security fields continue to enhance the security of data and communications, newer technology and open source libraries will continue to offer new and “exciting” exploits that must be diligently found and corrected. This goes hand in hand with other security measures like applying security patches as they become available and the concept of least privilege.

**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.

There are multiple areas of security that must be considered when looking at Artemis Financial’s web applications. Secure API interactions stands out as the most obvious. As a web based API the interactions within the API must be robust and secure so that the structure of the API is not used against customers or the company. Another area is Cryptography. Financial data is incredibly sensitive and care must be taken to ensure that a strong encryption suite is used to protect that data in transit, at rest, and in use. Client/Server interactions is another important are of security. These interactions can be intercepted, manipulated, or change system behavior and must be properly secured to maintain the integrity of those interactions. Code Quality and Encapsulation are areas that relate directly with how the software code is structured. Maintaining secure coding practices in how data in classes are stored and manipulated tries directly in with Encapsulation and can prevent serious damage even if other areas of security fall short.

**3. Manual Review**

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

Vulnerabilities found as part of the manual review include:

myDateTime.java

1. myDateTime class is public and unless required somewhere else, should be a protected class
2. int variables seconds, minutes, and hour are set to default and should be set to private

Customer.java

1. Customer class is public and should be set to protected or private
2. Int variable account\_balance is public and should be set to private
3. Private int variable account\_number should be set to private final to prevent changes after assignment

CRUD.java

1. The first constructor takes one string variable and assigns it to both content variables. This should either remove the assignment to content2 or remove the constructor

DocData.java

1. getID method executes without verifying that ID variable is set before execution that could cause a run time error or denial of service.

**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

The vulnerable dependencies packages are as follows with their associated description, recommendations for remediation, and CVE code:

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.

Recommended remediation: update to current version (1.70) and to receive required certificates out of band and stored so that they can be verified against provide certificates during operation

CVE codes: **CVE-2024-34447, CVE-2016-1000338, CVE-2016-1000342, CVE-2016-1000343, CVE-2024-29857, CVE-2016-1000344, CVE-2016-1000352, CVE-2024-30171, CVE-2016-1000341, CVE-2016-1000345, CVE-2017-13098, CVE-2020-15522, CVE-2020-0187, CVE-2023-33202, CVE-2020-26939, CVE-2023-33201, CVE-2016-1000339, CVE-2015-7940, CVE-2018-5382, CVE-2013-1624, CVE-2016-1000346, CVE-2015-6644**

hibernate-validator-6.0.18.Final.jar

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

Recommended remediation: update to current version (8.1.0.1Final)

CVE codes: **CVE-2020-10693**

jackson-databind-2.10.2.jar

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

Recommended remediation: update to current version (2.17.2)

CVE codes: **CVE-2020-25649**, **CVE-2020-36518**, **CVE-2021-46877**, **CVE-2022-42003**, **CVE-2022-42004**, **CVE-2023-35116**

log4j-api-2.12.1.jar

The Apache Log4j API

Recommended remediation: update to current version (2.23.1)

CVE codes: **CVE-2020-9488**

logback-core-1.2.3.jar

logback-core module

Recommended remediation: update to current version (1.5.6)

CVE codes: **CVE-2023-6378**, **CVE-2021-42550**

snakeyaml-1.25.jar

Snake YAML file parser and constructor

Recommended remediation: update to current version (2.2) and do not accept untrusted YAML files for parsing

CVE codes: **CVE-2022-1471**, **CVE-2017-18640**, **CVE-2022-25857**, **CVE-2022-38749**, **CVE-2022-38751**, **CVE-2022-38752**, **CVE-2022-41854**, **CVE-2022-38750**

spring-boot-2.2.4.RELEASE.jar

Spring Boot

Recommended remediation: update to current version (6.1.11) and run Spring boot as executable (default) and not on a Tomcat server or as part of a WAR deployment

CVE codes: **CVE-2023-20873**, **CVE-2022-27772**, **CVE-2023-20883**

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

Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container

Recommended remediation: update to current version (6.1.11) and run Spring boot as executable (default) and not on a Tomcat server or as part of a WAR deployment

CVE codes: **CVE-2023-20873**, **CVE-2022-27772**, **CVE-2023-20883**

spring-core-5.2.3.RELEASE.jar

Spring Core

Recommended remediation: update to current version (6.1.11) and run Spring boot as executable (default) and not on a Tomcat server or as part of a WAR deployment

CVE codes: **CVE-2022-22965**, **CVE-2021-22118**, **CVE-2020-5421**, **CVE-2022-22950**, **CVE-2022-22971**, **CVE-2023-20861**, **CVE-2023-20863**, **CVE-2022-22968**, **CVE-2022-22970**, **CVE-2021-22060**, **CVE-2021-22096**

spring-web-5.2.3.RELEASE.jar

Spring Web

Recommended remediation: update to current version (6.1.11) and run Spring boot as executable (default) and not on a Tomcat server or as part of a WAR deployment

CVE codes: **CVE-2016-1000027**, **CVE-2022-22965**, **CVE-2024-22243, CVE-2024-22262, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096**

spring-webmvc-5.2.3.RELEASE.jar

Spring Web MVC

Recommended remediation: update to current version (6.1.11) and run Spring boot as executable (default) and not on a Tomcat server or as part of a WAR deployment

CVE codes: **CVE-2022-22965**, **CVE-2021-22118**, **CVE-2020-5421**, **CVE-2022-22950**, **CVE-2022-22971**, **CVE-2023-20861**, **CVE-2023-20863**, **CVE-2022-22968**, **CVE-2022-22970**, **CVE-2021-22060**, **CVE-2021-22096**

tomcat-embed-core-9.0.30.jar

Core Tomcat implementation

Recommended remediation: update to current version (11.0.0-M22) and configure servers to close connections that exceed concurrent stream limits

CVE codes: **CVE-2020-1938**, **CVE-2020-11996**, **CVE-2020-13934**, **CVE-2020-13935**, **CVE-2020-17527**, **CVE-2021-25122**, **CVE-2021-41079**, **CVE-2022-29885**, **CVE-2022-42252**, **CVE-2023-44487**, **CVE-2023-46589**, **CVE-2020-9484**, **CVE-2021-25329**, **CVE-2021-30640**, **CVE-2022-34305**, **CVE-2023-41080**, **CVE-2021-24122**, **CVE-2021-33037**, **CVE-2023-42795**, **CVE-2023-45648**, **CVE-2024-21733**, **CVE-2019-17569**, **CVE-2020-1935**, **CVE-2020-13943**, **CVE-2023-28708**, **CVE-2021-43980**

tomcat-embed-websocket-9.0.30.jar

Core Tomcat implementation

Recommended remediation: update to current version (11.0.0-M22) and configure servers to close connections that exceed concurrent stream limits

CVE codes: **CVE-2020-1938**, **CVE-2020-8022**, **CVE-2020-11996**, **CVE-2020-13934**, **CVE-2020-13935**, **CVE-2020-17527**, **CVE-2021-25122**, **CVE-2021-41079**, **CVE-2022-29885**, **CVE-2022-42252**, **CVE-2023-44487**, **CVE-2023-46589**, **CVE-2020-9484**, **CVE-2021-25329**, **CVE-2021-30640**, **CVE-2022-34305**, **CVE-2023-41080**, **CVE-2021-24122**, **CVE-2021-33037**, **CVE-2023-42795**, **CVE-2023-45648**, **CVE-2024-21733**, **CVE-2019-17569**, **CVE-2020-1935**, **CVE-2020-13943**, **CVE-2023-28708**, **CVE-2021-43980**

**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.

Based on the outcomes of the manual review and static testing there are a couple recommended steps to mitigate the found vulnerabilities. First, ensuring that the most recent external packages are utilized in the program. This will mitigate found vulnerabilities but setting a process in place to check for vulnerabilities periodically and patching as required will ensure that the most robust version of outside dependencies are utilized and limit exposure to future vulnerabilities. Utilizing concepts like least privilege and least resources will assume limit the surface area available for exploitation. Ensuring that ports and connections are strictly configured for only the required permissions and purposes will help prevent some of the issues noted in the findings. Finally, proper encapsulation and access to data stored in the program will prevent system errors, denial of service due to runtime errors, prevent unintended execution of code or unpredictable behavior of the system.