# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **09/17/2025** | **James Ehrle** |  |

## 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

James Ehrle

**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?

The value of secure communications will be high to the Artemis Financial because they are handling financial plans that include saving, retirement, investments, and insurance which is considered private information. They will need to communicate with the client to ensure that they have all their information to set up these financial plans, this would possibly include personal information, banking information and possible user id/passwords which all could be the motivation of an attack. This software is for entrepreneurs, business, and government agencies around the world. Since it is around the world there will be international transactions that will be processed for the customers. There are some governmental restrictions that they will need to consider on secure communications and that is the Graham-Leach-Bliley Act, and it is a financial reconstruction law for financial institutions that they must publicly disclose their information-sharing practices. Another one that will need to be considered is if passed the regulations by the European Union that considers the privacy of its citizens. Some current threats that could be Brute Force attacks due to a lack of input validation or 2 factor authorization or limit on tries. Another could be hijacking of the communication due to a lack of encryption or man in the middle. Another threat that we will need to watch for is SQL injunction attacks. The modernization requirements to consider would be input validation, access control, updated and strengthened encryptions and secure API. Utilizing a DAO using user parameterization techniques to keep the application safe against injection.

**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.

Following the vulnerability assessment process flow diagram the areas of security that apply to Artemis Financials software application are input validation, APIs, cryptography, client/server, code error, and code quality.

**Input Validation:** if input validation isn’t used in the different fields, then it could leave the software open to vulnerabilities such as SQL injunction. Input validation could be in the form of limiting what characters are able to be used as well as the number of characters.

**APIs:** Since this software will be utilizing RESTful web application programming interface, we will need to make sure that it has strong secure API interaction. This will allow more secure data access from the APIs.

**Cryptography:** Since the company will be dealing with financial information for clients, encryption will be very important to make sure that the information isn’t stolen. This will ensure that even if someone was to intercept any messages, they wouldn’t be able to read them if the cypher is strong.

**Client/Server:** communication between the client/server will need to be taken into consideration since they will be working with business, entrepreneurs, and governments with sensitive financial information. This could cause a security vulnerability if the client side isn’t secure, they will be communicating with the clients through virtual methods since it is a global market.

**Code Error:** Coding without errors is important because errors can cause vulnerability to the web application. If the access control isn’t coded properly, it could allow access to the wrong user and cause problems. Also, if the input validation isn’t correctly coded that can cause it to not validate correctly.

**Code Quality:** Quality is a very important part because it will ensure that the correct and most up to date coding practices are being used so that it can ensure there is no vulnerability due to using outdated or less advanced coding techniques.

**3. Manual Review**

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

After reviewing the code, the vulnerabilities that I found are:

* **There is not an input validation method for the username.**
* **There is not an input validation method for the Id.**
* **There is not an input validation method for the password.**
* **Incorrect dependency check version 5.3.0 was being used rather than the newest version 12.1.0.**
* **Restcontroller is not incorporated properly due to the springframework not being incorporated correctly.**
* **SpringFramework API is not properly incorporated which is causing it to not be called properly throughout the code.**
* **Requestmapping is not incorporated properly due to the springframework not being incorporated correctly.**
* **Requestparam is not incorporated properly due to the springframework not being incorporated correctly.**
* **Getmapping is not incorporated properly due to the springframework not being incorporated correctly.**

**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:

A screenshot of a website

Description automatically generated

A screenshot of a computer

Description automatically generated

* 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.

Vulnerabilities: CVE-2024-34447, CVE-2016-1000338, CVE-2016-1000342, CVE-2016-1000343, CVE-2024-29857, CVE-2016-1000341, CVE-2016-1000345, CVE-2024-30171, CVE-2020-15522, CVE-2020-0187, CVE-2023-33202, CVE-2020-26939, CVE-2023-33201, CVE-2015-7940, CVE-2018-5382, CVE-2013-1624, CVE-2016-1000346, CVE-2015-6644.

This is due to the bouncy castle using improper validation for encoding of signature. This could allow the introduction of “invisible” data into a signed structure. The verification of cryptographic signature is improper. Possible solutions is to apply the update.

* hibernate-validator-6.0.18.Final.jar:

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

Vulnerabilities: CVE-2025-35036, CVE-2023-1932, CVE-2020-10693

This is due to needing to update the hibernate validator because the older versions could allow an attacker to access sensitive information or execute arbitrary java code. Possible solution to this is to update to the most current version.

* jackson-core-2.10.2.jar:

Core Jackson processing abstractions (aka Streaming API), implementation for JSON

Vulnerabilities: CVE-2025-52999, CVE-2025-49128

This is due to using an older version of Jackson Data Processor. This can cause the program to throw a StoackoverflowError if a user parses an iput file and it has deeply nested date. A possible fix to this would be to use the most updated version.

* jackson-databind-2.10.2.jar:

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

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

This is due to using an outdate version of Jackson-databind which could allow attackers to cause a denial of service. A possible solution for this would be to update the version to the most current. Possible correction to this problem is to update to get the critical patches.

* log4j-api-2.12.1.jar:

The Apache Log4j API

Vulnerabilities: CVE-2020-9488

This is due to improper validation of certificate with host, this could cause the connection to be intercepted by a man-in-the-middle attack. Possible correction to this problem is to update to get the critical patches.

* logback-classic-1.2.3.jar:

logback-classic module

Vulnerabilities: CVE-2023-6378, CVE-2021-42550

This is due to not having the most updated version, this has a serialization vulnerability in the logback receiver component that is part of the current version that would allow the attacker to mount a denial-of-service attack by sending poisoned data. A possible fix to this problem would be to use the most current version.

* logback-core-1.2.3.jar:

logback-core module

Vulnerabilities: CVE-2023-6378, CVE-2021-42550, CVE-2024-12798, CVE-2024-12801

This is a serialization vulnerability that can allow the attacker to mount a denial-of-service attack by sending poisoned data. Possible fix to this is to update to the most current version.

* snakeyaml-1.25.jar:

YAML 1.1 parser and emitter for Java

Vulnerabilities: CVE-2022-1471, CVE-2017-18640, CVE-2022-25857, CVE-2022-38749, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854, CVE-2022-38750.

This is due to the SnakeYaml’s constructor class not restricting types which can be instantiated during deserialization. This could lead to an attacker using remote code execution. Possible fix for this is to use SnakeYaml’s safeConstructer when parsing untrusted content and upgrading to the most current version.

* spring-aop-5.2.3.RELEASE.jar:

Spring AOP

Vulnerabilities: CVE-2022-22965, CVE-2024-22259, 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

This can be caused by the Spring MVC not running on JDK 9+ which could cause remode code execution via data binding. Possible fix for this would be to upgrade to the most recent JDK.

* spring-boot-2.2.4.RELEASE.jar:

Spring Boot

Vulnerabilities: CVE-2023-20873, CVE-2022-27772, CVE-2023-20883

This is from using previous version of spring boot which are unsupported versions and could be susceptible to a security bypass. Possible fixes would be updated to using the most current version.

* 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.

Vulnerabilities: CVE-2023-20873, CVE-2022-27772, CVE-2023-20883

This is due to the spring boot version being older and an unsupported version. The could be susceptible to a security bypass. The possible solution to this would be to use the most current version.

* spring-context-5.2.3.RELEASE.jar:

Spring Context

Vulnerabilities: CVE-2022-22965, CVE-2024-22259, 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, CVE-2025-22233

This is due to using an older version of JDK, which may be vulnerable to remote code execution via data binding. To fix this a newer version of JDK will need to be used. This would also require it to run on tomcat as a WAR deployment.

* spring-core-5.2.3.RELEASE.jar:

Spring Core

Vulnerabilities: CVE-2022-22965, CVE-2025-41249, CVE-2025-41242, CVE-2024-22259, 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

This is due to using outdate FDK remote code execution, this can cause attackers to have remote code execution via data binding. The solution for this would be to update the JDK version to the most current and make sure the most updated Spring Framework is being used.

* spring-expression-5.2.3.RELEASE.jar:

Spring Expression Language

Vulnerabilities: CVE-2022-22965, CVE-2024-22259, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2024-38808, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096

This is due to using outdate FDK remote code execution, this can cause attackers to have remote code execution via data binding. The solution for this would be to update the JDK version to the most current and make sure the most updated Spring Framework is being used.

* spring-web-5.2.3.RELEASE.jar:

Spring Web

Vulnerabilities: CVE-2016-1000027, CVE-2022-22965, CVE-2024-38809, CVE-2025-41249, CVE-2024-22243, CVE-2024-22262, CVE-2024-22259, CVE-2021-22118, CVE-2025-41234, CVE-2024-38828, 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

This is due to using a outdated version of spring framework that suffers from a potential remote code execution if used for java deserialization of untrusted data. The way to solve this would be to use the most updated version of Spring Framework.

* spring-webmvc-5.2.3.RELEASE.jar:

Spring Web MVC

Vulnerabilities: CVE-2022-22965, CVE-2024-38816, CVE-2024-22259, 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

This is due to using outdate FDK remote code execution, this can cause attackers to have remote code execution via data binding. The solution for this would be to update the JDK version to the most current and make sure the most updated Spring Framework is being used.

* tomcat-embed-core-9.0.30.jar:

Core Tomcat Implementation.

Vulnerabilities: CVE-2020-1938, CVE-2024-50379, CVE-2024-52316, CVE-2024-56337, CVE-2025-24813, CVE-2025-31651, CVE-2025-49124, 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-2024-24549, CVE-2024-34750, CVE-2024-38286, CVE-2025-48988, CVE-2025-48989, CVE-2025-49125, CVE-2025-52434, CVE-2025-52520, CVE-2025-53506, CVE-2025-46701, CVE-2020-9484, CVE-2021-25329, CVE-2021-30640, CVE-2025-55668, CVE-2024-23672, CVE-2022-34305, CVE-2023-41080, CVE-2021-24122, CVE-2021-33037, CVE-2023-42795, CVE-2023-45648, CVE-2024-21733, CVE-2024-54677, CVE-2019-17569, CVE-2020-1935 , CVE-2020-13943, CVE-2023-28708, CVE-2021-43980

This is because an HTTP connection isn’t being used and it could be exploited in ways that may be surprising if connections are available. To help solve the use the most update version of tomcat will help because the issues have been solved in the security fixes.

* tomcat-embed-websocket-9.0.30.jar:

Core tomcat implementation.

Vulnerabilities: CVE-2020-1938, CVE-2024-50379, CVE-2024-52316, CVE-2024-56337, CVE-2025-24813, CVE-2025-31651, CVE-2025-49124 , 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-2024-24549, CVE-2024-34750, CVE-2024-38286, CVE-2025-48988, CVE-2025-48989, CVE-2025-49125, CVE-2025-52434, CVE-2025-52520, CVE-2025-53506, CVE-2025-46701, CVE-2020-9484, CVE-2021-25329, CVE-2021-30640, CVE-2025-55668, CVE-2024-23672, CVE-2022-34305, CVE-2023-41080, CVE-2021-24122, CVE-2021-33037, CVE-2023-42795, CVE-2023-45648 , CVE-2024-21733, CVE-2024-54677, CVE-2019-17569, CVE-2020-1935, CVE-2020-13943 , CVE-2023-28708, CVE-2021-43980

This is due to using an outdated version of tomcat which can lead to input validation error as well as improper error handling. The possible fix to this would be to update using the most recent tomcat 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.

Based on the results from the manual review and the static testing report there are some steps that we need to take to mitigate the security vulnerabilities. First, there need to be input validations incorporated with in the program, anywhere there is an input that is need. By doing that it will minimize the ability to for attacks such as SQL injection or buffer overflow. Another item we need to look at is encryption, by encrypting the important information with a strong cypher it will keep the financial information secure. This will help reduce attacks such as man-in-the-middle where the attacker would get the information. Another thing that we need to look at is using the most updated versions of all the software so we can make sure any security patch is being used to avoid any issues that have already been fixed by just using updated versions. We will also need to make sure we are looking at code quality to make sure that we are using the most updated libraries to ensure it isn’t allowing for any patches to the security that are fixed in new libraries or updated coding procedures. Lastly code errors when using APIs, if there is error it could cause security vulnerabilities.

Reference:

(n.d.). Dependency-check. Dependency-Check. file:///Users/jamesehrle/Downloads/rest-service/target/dependency-check-report.html#l3\_7fd00bcd87e14b6ba66279282ef15efa30dd2492