****

# Artemis Financial Vulnerability Assessment Report

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Instructions 3](#_Toc32574609)

[Developer 4](#_Toc32574610)

[1. Interpreting Client Needs 4](#_Toc32574611)

[2. Areas of Security 4](#_Toc32574612)

[3. Manual Review 4](#_Toc32574613)

[4. Static Testing 4](#_Toc32574614)

[5. Mitigation Plan 4](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/27/2024** | **Brandon Barrett** |  |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In the report, identify your findings of security vulnerabilities and provide recommendations for 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 choose to include images or supporting materials. If you include them, make certain to insert them in all 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

Brandon Barrett

## Interpreting Client Needs

1. **Interpreting Client Needs**: Review the scenario to determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Document your findings in your vulnerability assessment report. Consider the scenario information and the following questions regarding how companies protect against external threats:
   1. What is the value of secure communications to the company?
   2. Does the company make any international transactions?
   3. Are there governmental restrictions about secure communications to consider?
   4. What external threats might be present now and in the immediate future?
   5. What are the modernization requirements that you must consider? For example:
      1. The role of open-source libraries
      2. Evolving web application technologies

Secure communications would be at the utmost importance for Artemis Financial due to the client information that the company contains. Information such as savings, investments, and insurance data will be required to be protected from unauthorized users. Secure communications ensure the integrity of client data and it helps to hold trust with the company's clientele.

Artemis Financial deals with individualized financial plans which could consist of international investments or transactions. Secure communications would be important to keep sensitive financial information safe during international transactions.

There may be international regulations to watch out for and adhering to these regulations would be crucial to avoid any legal issues and ensure that the company is abiding by all laws and regulations.

Artemis Financial may be at risk of cyber-attacks, data breaches, and unauthorized access to their web applications. They would also need to be aware of phishing, ransomware, and SQL injection attacks in their financial sector. Adopting security measures to prevent these attacks and training employees on what to look for would help prevent these attacks.

The emphasis on modernizing operations requires an approach to secure communications, international transaction considerations, adherence to governmental regulations, identification of external threats, and the evaluation of modernization requirements such as the role of open-source libraries and evolving web application technologies. Open-source libraries can accelerate development, but it is important to ensure they are up-to-date and adhere to security best practices. Modern frameworks, secure coding practices, and staying current with industry standards is important for overall security and performance.

## Areas of Security

**Areas of Security**: Use what you’ve learned in step 1 and refer to the Vulnerability Assessment Process Flow Diagram provided. Think about the functionality of the software application to identify which areas of security apply to Artemis Financial’s web application. Document your findings in your vulnerability assessment report and justify why each area is relevant to the software application.

* Input Validation – Input validation prevents common web application attacks and it ensures that user inputs are validated. This provides protection for the users.
* API’s – API implementation provides secure endpoints, proper authentication, and validates incoming requests.
* Code Quality – Well reviewed and well written code will reduce the likelihood of allowing vulnerabilities and makes it easier to address security issues.
* Code Error – Handling code properly will help to prevent information being exploited by attackers. The proper handling of crucial information is important to all users and it is important that they trust the organizations that have access to this information.
* Cryptography – This is important for ensuring the integrity of sensitive financial information. Encrypting data at rest and during transmission is important to prevent unauthorized access to the information.

## Manual Review

**Manual Review**: Refer to the seven security areas outlined in the Vulnerability Assessment Process Flow Diagram. Use what you’ve learned in steps 1 and 2 to guide your manual review. Identify all vulnerabilities in the Project One Code Base, linked in Supporting Materials, by manually inspecting the code. Document your findings in your vulnerability assessment report. Be sure to include a description that identifies where the vulnerabilities are found (specific class file, if applicable).

CRUD.Java – There are no apparent security vulnerabilities, however, it should be noted that comments are important, and they should be used to at least let others know what the class is for.

CRUDController.Java – The ‘/read’ endpoint takes a ‘business\_name’ parameter, but it is not used in the method.

Customer.Java – The ‘showInfo’ method returns the account number, but there is no validation or authentication.

DocData.Java – The ‘read\_document’ method contains a commented-out section for connecting to a MySQL database, but the connection is not used.

Greeting.Java – No security vulnerabilities found.

GreetingController.Java – The ‘/greeting’ endpoint generates a greeting based on user input.

myDateTime.Java – No security vulnerabilities found.

RestServiceApplication.Java – No security vulnerabilities found.

## Static Testing

**Static Testing:**Integrate the dependency-check plug-in into Maven by following the instructions outlined in the Integrating the Maven Dependency-Check Plug-in tutorial provided in Supporting Materials. Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Specifically, identify all vulnerabilities in the code base by analyzing results from running the code through a static test.Include these items from the dependency-check report in your vulnerability assessment report:

* 1. The names or vulnerability codes of the known vulnerabilities
  2. A brief description and recommended solutions that are found in the dependency-check report
  3. Attribution (if any) that documents how this vulnerability has been identified or how it was documented in the past

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dependency | Vulnerability | Description | Solution | |
| [bcprov-jdk15on-1.46.jar](file:///C:\Users\brand\eclipse-workspace\CS%20305%20Project%20One%20Code%20Base.zip_expanded\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7) | [**CVE-2016-1000338**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338)  [**CVE-2016-1000342**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342)  [**CVE-2016-1000343**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000343)  [**CVE-2016-1000344**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000344)  [**CVE-2016-1000352**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000352)  [**CVE-2016-1000341**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000341)  [**CVE-2016-1000345**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000345)  [**CVE-2017-13098**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-13098)  [**CVE-2020-15522**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-15522)  [**CVE-2023-33202**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-33202)  [**CVE-2016-1000339**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000339)  [**CVE-2015-7940**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-7940)  [**CVE-2018-5382**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-5382)  [**CVE-2013-1624**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624)  [**CVE-2016-1000346**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000346) | 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. | Conduct a version update. | |
| hibernate-validator-6.0.18.Final.jar | [**CVE-2020-10693**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693) | Hibernate's Bean Validation (JSR-380) reference implementation. | Upgrade Hibernate Validator to 6.0.20. | |
| jackson-databind-2.10.2.jar | [**CVE-2020-25649**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649)  [**CVE-2020-36518**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-36518)  [**CVE-2021-46877**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-46877)  [**CVE-2022-42003**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42003)  [**CVE-2022-42004**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42004)  [**CVE-2023-35116**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-35116) | General data-binding functionality for Jackson: works on core streaming API | Update to current version | |
| log4j-api-2.12.1.jar | [**CVE-2020-9488**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488) | The Apache Log4j API | Upgrade to version 2.13.2 | |
| spring-web-5.2.3.RELEASE.jar | [**CVE-2016-1000027**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000027)  [**CVE-2021-22118**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118)  [**CVE-2020-5421**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421)  [**CVE-2022-22950**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950)  [**CVE-2022-22971**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971)  [**CVE-2023-20861**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20861)  [**CVE-2023-20863**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20863)  [**CVE-2022-22968**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968)  [**CVE-2022-22970**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970)  [**CVE-2021-22060**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060)  [**CVE-2021-22096**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096) | Spring Web  In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter. | Upgrade to current Version | |
| tomcat-embed-core-9.0.30.jar | [**CVE-2020-11996**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996)  [**CVE-2020-13934**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934)  [**CVE-2020-13935**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935)  [**CVE-2020-17527**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527)  [**CVE-2021-25122**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122)  [**CVE-2021-41079**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079)  [**CVE-2022-29885**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885)  [**CVE-2022-42252**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42252)  [**CVE-2023-44487**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-44487)  [**CVE-2023-46589**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-46589)  [**CVE-2020-9484**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484)  [**CVE-2021-25329**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329)  [**CVE-2021-30640**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640)  [**CVE-2022-34305**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305)  [**CVE-2023-41080**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-41080)  [**CVE-2021-24122**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122)  [**CVE-2021-33037**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037)  [**CVE-2023-42795**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-42795) | Core Tomcat implementation  A specially crafted sequence of HTTP/2 requests sent to Apache Tomcat 10.0.0-M1 to 10.0.0-M5, 9.0.0.M1 to 9.0.35 and 8.5.0 to 8.5.55 could trigger high CPU usage for several seconds. If a sufficient number of such requests were made on concurrent HTTP/2 connections, the server could become unresponsive.  An h2c direct connection to Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M5 to 9.0.36 and 8.5.1 to 8.5.56 did not release the HTTP/1.1 processor after the upgrade to HTTP/2. If a sufficient number of such requests were made, an OutOfMemoryException could occur leading to a denial of service.  The payload length in a WebSocket frame was not correctly validated in Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M1 to 9.0.36, 8.5.0 to 8.5.56 and 7.0.27 to 7.0.104. Invalid payload lengths could trigger an infinite loop. Multiple requests with invalid payload lengths could lead to a denial of service. | Upgrade to current Version | | |
|  |  |  |  |  | | |

## Mitigation Plan

**Mitigation Plan:**Interpret the results from the manual review and static testing report. Identify steps to mitigate the identified security vulnerabilities by creating an action list that documents how to fix each vulnerability in your vulnerability assessment report.

After analyzing the test results and reviewing the various security vulnerabilities I can conclude the best plan of action is to upgrade to the current versions of Snakeyaml, hibernator validator, Apache Tomcat, and bouncycastle.