

# 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** | **11/2/2023** | **Trent Hesler** | **Initial Assesments** |

## 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

Trent Hesler

## Interpreting Client Needs

There are four main motivations when it comes to hacking software: revenge, activism, government sponsorship, and monetary gain (Kingland, 2022). As a financial institution, that last motivation is of particular importance to Artemis Financial. Artemis Financial’s clients trust them to handle their money and personal information in a safe and secure manner, and having secure software is fundamental to meeting this expectation. Further, as a financial institution operating within the United States, Artemis Financial is subject to many laws and regulations that dictate how they handle customer financial data.

Since Artemis Financial’s software uses a RESTful API that will be accessed remotely over the internet, it is of critical importance to make sure that all communications between the client’s computer and the server hosting the API are secure. To prevent interception of sensitive data, the software should employ encrypted communications through the HTTPS protocol and use secure HTTP response headers.

It is also a good idea to consider the use of open source libraries. Software libraries are collections of code that can be integrated into a program to expand functionality without the developers having to write everything from scratch. The advantage of open source software libraries is that they have been looked over and audited by the software development community at large. Having many eyes looking over a piece of software can improve the security of said software because one developer may catch a vulnerability that another has missed.

## Areas of Security

When we talk about software security, there are seven areas of security to consider when evaluating a project: Input Validation, APIs, Cryptography, Client/Server, Code Error, Code Quality, and Encapsulation. Only some of these seven will apply to any given project.

In the case of Artemis Financial, the security areas of focus are:

* **Input Validation:** It is important to check all input, either entered by hand by the client or extracted from a data source, such as a database, to ensure that it is secure. Many pieces of software have had their security broken by an injection attack, wherein an attacker inserts code or invalid data into an input field in order to make the program behave in a way contrary to its design.
* **APIs:** Since clients will be interacting with this software using the RESTful API, it is critical to ensure that all communications to and from the API are secure and cannot be intercepted.
* **Cryptography:** To secure communications to and from the API, the software will need to make use of industry standard cryptographic protections, such as the HTTPS protocol.
* **Code Quality:** The Code Quality area of security pertains to the use of secure coding practices. While this should be of importance to any software development project, it is especially important here because of the sensitivity of the data being handled and the liability involved with mishandling said data.
* **Code Error:** As alluded to earlier, one way that attackers attempt to circumvent software security is by causing erroneous behavior. This is why it is important to have strong error handling methods and prepare for contingencies.

## Manual Review

As part of this vulnerability assessment, a manual code review was performed. Below are the vulnerabilities discovered through this review:

* In customer.java, the *account\_balance* attribute should be private.
* In DocData.java, the MySQL database connection information is hardcoded.
* The root password for the MySQL server should be changed to something more secure than “root.”
* In GreetingController.java, input validation should be performed on the *name* String that is passed in.
* In CRUDController.java, it appears that *business\_name* is being passed in as a URL parameter. This is unsafe as it can be easily exploited. Recommend passing data in using the POST method.
* The system does not appear to be using HTTPS. This means that all information sent to and from the system is sent in plain text and can be easily intercepted.

## Static Testing

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

* **bcprov-jdk15on-1.46.jar** 
  + **Vulnerabilities (17):**  
    CVE-2013-1624 - Cryptographic Issues  
    CVE-2015-6644 - Information Exposure  
    CVE-2015-7940 - Information Exposure  
    CVE-2016-1000338 - Improper Verification of Cryptographic Signature  
    CVE-2016-1000339 - Cryptographic Issues  
    CVE-2016-1000341 - Private Key Generation Vulnerabilities  
    CVE-2016-1000342 - Improper Verification of Cryptographic Signature  
    CVE-2016-1000343 - Cryptographic Issues  
    CVE-2016-1000344 - Cryptographic Issues  
    CVE-2016-1000345 - Private Key Generation Vulnerabilities  
    CVE-2016-1000346 - Key Management Errors  
    CVE-2016-1000352 - Cryptographic Issues  
    CVE-2017-13098 - Information Exposure Through Discrepancy  
    CVE-2018-5382 - Improper Validation of Integrity Check Value  
    CVE-2020-0187 - Cryptographic Issues  
    CVE-2020-26939 - Information Exposure Through Discrepancy  
    CVE-2023-33201 - Improper Certificate Validation
  + **Recommendation:** Update this dependency to version 1.6 or higher.
* **spring-boot-2.2.4.RELEASE.jar**
  + **Vulnerabilities (3):**   
    CVE-2022-27772 - Exposure of Resource to Wrong Sphere  
    CVE-2023-20873 - Applications deployed to Cloud Foundry could be susceptible to security bypass  
    CVE-2023-20883 - Uncontrolled Resource Consumption ('Resource Exhaustion')
  + **Recommendation:** Update this dependency to version 3.0.7 or higher.
* **logback-core-1.2.3.jar**
  + **Vulnerabilities (1):**   
    CVE-2021-42550 - Deserialization of Untrusted Data
  + **Recommendation:** Update this dependency to version 1.2.9 or higher.
* **log4j-api-2.12.1.jar**
  + **Vulnerabilities (5):**CVE-2020-9488 - Improper Certificate Validation  
    CVE-2021-44228 - Uncontrolled Resource Consumption ('Resource Exhaustion'), Deserialization of Untrusted Data, Improper Input Validation  
    CVE-2021-44832 - Improper Input Validation  
    CVE-2021-45046 - Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')  
    CVE-2021-45105 - Improper Input Validation, Uncontrolled Recursion
  + **Recommendation:** Update this dependency to version 2.17.1 or higher.
* **snakeyaml-1.25.jar**
  + **Vulnerabilities (10):**CVE-2017-18640 - Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')  
    CVE-2021-4235 - Uncontrolled Resource Consumption ('Resource Exhaustion')  
    CVE-2022-1471 - Deserialization of Untrusted Data  
    CVE-2022-25857 - Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')  
    CVE-2022-3064 - Uncontrolled Resource Consumption ('Resource Exhaustion')  
    CVE-2022-38749 - Out-of-bounds Write  
    CVE-2022-38750 - Out-of-bounds Write  
    CVE-2022-38751 - Out-of-bounds Write  
    CVE-2022-38752 - Out-of-bounds Write  
    CVE-2022-41854 - Out-of-bounds Write
  + **Recommendation:** Update this dependency to version 2.2.4 or higher.
* **jackson-databind-2.10.2.jar**
  + **Vulnerabilities (6):**CVE-2020-25649 - Improper Restriction of XML External Entity Reference ('XXE')  
    CVE-2020-36518 - Out-of-bounds Write  
    CVE-2021-46877 - Allocation of Resources Without Limits or Throttling  
    CVE-2022-42003 - Deserialization of Untrusted Data  
    CVE-2022-42004 - Deserialization of Untrusted Data  
    CVE-2023-35116 - Allocation of Resources Without Limits or Throttling
  + **Recommendation:** Update this dependency to version 2.15.3 or higher.
* **tomcat-embed-core-9.0.30.jar**
  + **Vulnerabilities (25):**CVE-2019-17569 - Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')  
    CVE-2020-11996 - Resource Management Issues  
    CVE-2020-13934 - Improper Release of Memory Before Removing Last Reference ('Memory Leak'), NULL Pointer Dereference  
    CVE-2020-13935 - Loop with Unreachable Exit Condition ('Infinite Loop')  
    CVE-2020-13943 - Information Exposure  
    CVE-2020-17527 - Information Exposure  
    CVE-2020-1935 - Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')  
    CVE-2020-1938 - Un(der)secured Protocol  
    CVE-2020-8022 - Incorrect Default Permissions  
    CVE-2020-9484 - Deserialization of Untrusted Data  
    CVE-2021-24122 - Use of Incorrectly-Resolved Name or Reference  
    CVE-2021-25122 - Information Exposure  
    CVE-2021-25329 - Uncontrolled Resource Consumption ('Resource Exhaustion')  
    CVE-2021-30640 - Improper Encoding or Escaping of Output  
    CVE-2021-33037 - Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')  
    CVE-2021-41079 - Loop with Unreachable Exit Condition ('Infinite Loop')  
    CVE-2021-43980 - Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')  
    CVE-2022-29885 - Uncontrolled Resource Consumption ('Resource Exhaustion')  
    CVE-2022-34305 - Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')  
    CVE-2022-42252 - Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')  
    CVE-2023-28708 - Unprotected Transport of Credentials  
    CVE-2023-41080 - URL Redirection to Untrusted Site ('Open Redirect')  
    CVE-2023-42795 - Incomplete Cleanup  
    CVE-2023-44487 - Uncontrolled Resource Consumption ('Resource Exhaustion')  
    CVE-2023-45648 - Improper Input Validation
  + **Recommendation:** Update this dependency to version 10.1.14 or higher.
* **hibernate-validator-6.0.18.Final.jar**
  + **Vulnerabilities (1):**CVE-2020-10693 - Improper Input Validation
  + **Recommendation:** Update this dependency to version 6.1.5 or higher.
* **spring-web-5.2.3.RELEASE.jar**
  + **Vulnerabilities (4):**CVE-2016-1000027 - Deserialization of Untrusted Data  
    CVE2020-5421 - Unauthorized Code Execution   
    CVE-2021-22096 - Improper Output Neutralization for Logs  
    CVE-2021-22118 - Exposure of Resource to Wrong Sphere
  + **Recommendation:** Update this dependency to version 6.0.0 or higher.
* **spring-beans-5.2.3.RELEASE.jar**
  + **Vulnerabilities (1):**CVE-2022-22965 - Improper Control of Generation of Code ('Code Injection')
  + **Recommendation:** Update this dependency to version 5.3.18 or higher.
* **spring-webmvc-5.2.3.RELEASE.jar**
  + **Vulnerabilities (1):**CVE-2021-22060 - Improper Output Neutralization for Logs
  + **Recommendation:** Update this dependency to 5.2.19 or higher
* **spring-context-5.2.3.RELEASE.jar**
  + **Vulnerabilities (1):**CVE-2022-22968 - Improper Handling of Case Sensitivity
  + **Recommendation:** Update this dependency to 5.2.21 or higher.
* **spring-expression-5.2.3.RELEASE.jar**
  + **Vulnerabilities (3):**CVE-2022-22950 - Allocation of Resources Without Limits or Throttling  
    CVE-2023-20861 - Improper Input Validation, Uncontrolled Resource Consumption ('Resource Exhaustion')  
    CVE-2023-20863 - Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')
  + **Recommendation:** Update this dependency to version 6.0.8 or higher.

## Mitigation Plan

Based on the previous findings, I would recommend the following:

* Update all vulnerable dependencies. The versions provided above are the minimum safe version for each dependency, recommend incorporating the most recent version practical for each.
* Make sure class attributes are properly encapsulated.
* Encrypt all communications to and from the software using HTTPS.
* Treat all input as suspicious and validate it.
* Use secure user credentials and avoid hard coding, where possible.
* Pass parameters using the POST method, as opposed to other more vulnerable methods currently in use.

# References

Kingland, J. (2022, June 22). *What Motivates A Hacker?* Retrieved from Veteran Owned Cybersecurity Firm - Blue Team Alpha: https://blueteamalpha.com/cybersecurity/what-motivates-a-hacker/