****

# 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** | **20 September 2022** | **Kyle Fernish** |  |

## 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

Kyle Fernish

## Interpreting Client Needs

1. What is the value of secure communications to the company?
   1. The value of secure communications to the company are vital. They work in finance and consulting for many different customers. Each customer’s security is vital to them, as each customer has a right to their own security and any breach of that security reflects poorly on them as well as loses them the customer who is harmed. The individual customer is harmed, and secondly after that they will struggle to gain new customers once the world sees that they cannot protect an individual, why anyone or any business trust them afterwards?
2. Does the company make any international transactions?
   1. While there is nothing that explicitly states the company does any international trade in the scenario, any company who conducts financial transactions or assessments must consider international markets. Therefore, it is safe to assume they must be protected against any threats that are relevant to any company who trades domestically or internationally.
3. Are there governmental restrictions about secure communications to consider?
   1. There are multiple restrictions that apply to financial advisement. As most know, insider trading is probably first amongst them. The next would be any scheme to “share” funds amongst accounts, in other words a Ponzi scheme. Therefore, the system must reflect that communication between customers and their individual advisors cannot be shared within the company. Each customer needs to be able to communicate on those terms.
4. What external threats might be present now and in the immediate future?
   1. There are multiple outside threats to be considered with the current program. Most of the program’s current dependencies have known external threat vulnerabilities to multiple attack methods from anyone who would look to breach it. Linked here is the list of those dependencies with known vulnerabilities: [Dependency-Check Report](file:///C:\Users\Kyle\eclipse-workspace\rest-service\target\dependency-check-report.html).
5. What are the modernization requirements that you must consider? For example:
   1. The role of open-source libraries
      1. The role of open-source libraries is not something to consider in this program. While some of the source code to program may be gained from these libraries, it would not be necessary to open-source this program nor rely on them to create it. This program is meant to be secure, and if published to open source it will no longer be secure, plainly stated. Simply because open source means completely publishing the source code and program, which would be opening private information to the world and allowing ANYONE to study the program and asses its vulnerabilities to weaken and attack it.
   2. Evolving web application technologies
      1. Evolving web application technology is something that cannot be totally mitigated. Every day we make strides in technology which means what today’s greatest advancement, is simply tomorrow’s everyday common use program. 20 years ago touch screens were leading edge, today we carry something with that capability in our pocket and take it for granted. This is not an excuse to not prepare for the best we can develop, only a warning that what we accomplish and provide will be passed by at any point in time.

## Areas of Security

* 1. Input validation
     1. Input validation needs to be reconstructed within the program, there is nothing that explicitly states user versus administrator roles within the program.
  2. Cryptography
     1. There is little within the program to reflect the use of cryptography in the program. While there are some private class objects such as the account number and balance, there is little else to show the security methods that are necessary. There is no stated use of secondary validation to the user accounts, nor is there anything for the administrators or employees.
  3. Code Quality
     1. The written code of the program is simply put, weak. It is bare bones as it sits now, while there is less the code the program does operate more efficiently in the terms of statistical analysis; it does not achieve its necessary efficiency for protecting the customer or the company.
  4. Encapsulation
     1. The program does not appear to utilize any form of encapsulation. There are few methods in any declared class, and those classes do not incorporate any previously constructed class. Simply put they operate for their own individual purposes, but do not work together to operate with security or efficiency. There are minimal getter methods and no setter methods written to protect customer account information.

## Manual Review

1. CRUD.java
   1. The CRUD class only uses getter methods to access content data, while it does have to getter methods, there are no setter methods in the first place for those methods to utilize or return any data.
2. Customer.java
   1. The customer class again does not utilize get or set methods to truly secure account information. Without these, the program can be easily manipulated to return anything that an attacker would want to do. There is nothing that incorporates the class where the user would log into their account, while the account information may be a private class object it is not protected from the access to the account itself.
3. DocData.java
   1. The class has a get method for the user ID, but there is no inclusion of any other class for it to pull from. As it sits there are no variables in the class, public or private so there is nothing for the class to use to read into itself.
4. Greeting.java
   1. The class does differentiate user from administrator log in information, anyone who would access the program can use any log in to do anything inside of it. There must be a separation of two users of the program.
5. myDateTime.java
   1. While it seems innocuous to secure this information from anyone outside of the program, date and times reflect when and how any trades could be made by the user if available. Different markets open at different times; therefore trades cannot be made legally. If someone were to manipulate these values, the customer and company could come under scrutiny of illegal trades outside of legal times to do so.

## Static Testing

**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.

**Recommendation**: There is a fix provided by the vendor to fix this vulnerability, Build LMY49F, 6.0 with Security Patch Level of January 1, 2016

**hibernate-validator-6.0.18.Final.jar**

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

**Recommendation:**  Hibernate has published a fix for this issue, Hibernate Validator 6.1.5

**jackson-databind-2.10.2.jar**

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

**Recommendation:** There is an upgraded version from the vendor here, com.fasterxml.jackson.core:jackson-databind:2.11.0.rc1,2.10.5,2.9.10.7,2.6.7.4

**log4j-api-2.12.1.jar**

**Description:** The Apache Log4j API, This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1

**Recommendation:**  Upgrade to Log4j 2.3.2 (for Java 6), 2.12.4 (for Java 7), or 2.17.1 (for Java 8 and later).

In prior releases confirm that if the JDBC Appener is being used it is not configured to use any protocol other than Java.

Note that only the log4j-core JAR file is impacted by this vulnerability. Applications using only the log4j-api JAR file without the log4j-core JAR file are not impacted by this vulnerability.

Also note that Apache Log4j is the only Logging Services subproject affected by this vulnerability. Other projects like Log4net and Log4cxx are not impacted by this.

**logback-core-1.2.3.jar**

**Description:**  In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.

**Recommendation:**  The vendor has generated a newer version of logback-core that mends these vulnerabilities.

**snakeyaml-1.25.jar**

**Description:** The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

**Recommendation:** Upgrade org.yaml:snakeyaml to version 1.31 or higher.

**spring-boot-2.2.4.RELEASE.jar**

**Description:**   
\*\* UNSUPPORTED WHEN ASSIGNED \*\* spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.

**Recommendation:** Build your project using mvn clean install.

Check the contents of your jar file: go to your\_project/target directory and run jar tf your-project.jar command. There should be only log4j-core-2.17.0.jar and no other versions of log4j-core.

**spring-core-5.2.3.RELEASE.jar**

**Description:** A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.

**Recommendation:** The vendor has provided an updated version to fix vulnerabilities, 5.3.20

**spring-web-5.2.3.RELEASE.jar**

**Description:**   
Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. Depending on how the library is implemented within a product, this issue may or not occur, and authentication may be required. NOTE: the vendor's position is that untrusted data is not an intended use case. The product's behavior will not be changed because some users rely on deserialization of trusted data.

**Recommendation:**  The vendor has provided an updated version to fix vulnerabilities, 5.3.20

**tomcat-embed-core-9.0.30.jar**

**Description:** When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.

**Recommendation:** Upgrade to tomcat version 9.0.36.

**tomcat-embed-websocket-9.0.30.jar**

**Description:** When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.

**Recommendation:** Upgrade to tomcat version 9.0.36

## Mitigation Plan

The mitigation plan to fix the discovered vulnerabilities listed above would be to follow the recommendations provided by either upgrading the used versions of programs or look within the systems used and change the syntax within the program itself. Whichever route is more efficient or simply doable would be the simplest route to secure the program.

If any of the recommendations provided do not solve the issues, the next step would be to contact the vendors who published these recommendations and fixes and work with them to solve the problem.