****

# CS 305 Project One

**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** | **7/17/2022** | **Kelly Lewis** | **Initial Review** |

## Client



## Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found.

Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Kelly Lewis

## 1. Interpreting Client Needs

Determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Consider the following 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 about secure communications to consider?
* What external threats might be present now and in the immediate future?
* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?

Our client, Artemis Financial, has numerous needs based on the operations of the company. These operations include financial plans for savings, retirement, investments, and insurance for their clientele. Due to the sensitivity of the data being utilized by the client, there are obvious security concerns. Currently, there appear to be no international transactions that are produced by the company; however, this should be discussed further as part of both current data transfers and future growth plans. For both domestic and international transactions, more information is required in understanding the restrictions for monetary data transfers. Finally, regarding modernization requirements, updating existing program APIs and Frameworks to current releases should be a priority as a major source of securing the current software.

## 2. Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

Areas of concern and focus:

* Input Validation
  + Ensuring proper input validation, not only from outside input, but internal input from function to function is required to ensuring memory leaking/ overloading does not occur. In addition, ensuring bypasses by overloading are blocked or handled appropriately.
* APIs
  + Ensuring the latest releases of APIs are implemented to correct for found vulnerabilities. Utilizing the latest release is useful in patching holes in security caused by APIs.
* Cryptography
  + Due to the sensitive nature of the data being handled, encrypting is necessary in preventing leaked data from being properly used.
* Client/ Server
  + Due to the use of REST and RESTful APIs, including the current and planned use of the system, ensuring proper client/ server communication and security is important.
* Code Error
  + Ensuring that the code is free of errors, both syntax and logic based errors, is important to prevent exploits or overloads of system functions.
* Code Quality
  + Ensuring proper secure coding practices and patterns is necessary in preventing unwanted breaches to the program security. This also applies to ensuring using the latest patterns for encryption and data flows.
* Encapsulation
  + Encapsulating and creating secure data structures within the code base is a first line defense against unwanted data access.

## 3. Manual Review

Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

Code Review:

Upon manual code review, there is an unused variable in the DocData.java class “con” (line 26) that stands out. Unused variables can lead to security risks as they can eventually be exploited by attackers. Additionally, the use of proper documentation throughout the code base is unseen. Proper documentation leads to easier review and understanding of code uses.

For all other classes:

* CRUD.java
  + The overloading CRUD class that uses two strings could be reworked using better naming. Missing Documentation.
* CRUDController.java
  + Missing documentation.
* Customer.java
  + Review variables and public/ private assignments. Missing Documentation.
* DocData.java
  + Line 6 as stated above. Missing Documentation.
* Greeting.java
  + Check variable assignments. Missing Documentation.
* GreetingController.java
  + Check variable assignments (i.e. use of “final”). Missing Documentation.
* myDateTime.java
  + Check variables and use in setMyDateTime. Check documentation/ missing documentation.
* RestServiceApplication.java
  + Missing Documentation.

Without proper documentation it is quite difficult to follow the applications logic.

## 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 dependency check report. Include the following:

1. The names or vulnerability codes of the known vulnerabilities
2. A brief description and recommended solutions provided by the dependency check report
3. Attribution (if any) that documents how this vulnerability has been identified or documented previously

Identified Vulnerabilities:

* 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.
  + License:
    - Bouncy Castle License: <http://www.bouncycastle.org/licence.html>
  + Vulnerabilities
    - The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169.
* spring-boot-2.2.4.RELEASE.jar
  + Description:
    - Spring Boot
  + License:
    - Apache License, Version 2.0: <https://www.apache.org/licenses/LICENSE-2.0>
  + Vulnerabilities
    - The Spring OXM wrapper in Spring Framework before 3.2.4 and 4.0.0.M1, when using the JAXB marshaller, does not disable entity resolution, which allows context-dependent attackers to read arbitrary files, cause a denial of service, and conduct CSRF attacks via an XML external entity declaration in conjunction with an entity reference in a (1) DOMSource, (2) StAXSource, (3) SAXSource, or (4) StreamSource, aka an XML External Entity (XXE) issue.
* logback-core-1.2.3.jar
  + Description:
    - logback-core module
  + License:
    - http://www.eclipse.org/legal/epl-v10.html, <http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html>
  + Vulnerabilities
    - 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.
* log4j-api-2.12.1.jar
  + Description:
    - The Apache Log4j API
  + License:
    - <https://www.apache.org/licenses/LICENSE-2.0.txt>
  + Vulnerabilities
    - Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. 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
* snakeyaml-1.25.jar
  + Description:
    - YAML 1.1 parser and emitter for Java
  + License:
    - Apache License, Version 2.0: <http://www.apache.org/licenses/LICENSE-2.0.txt>
  + Vulnerabilities
    - The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.
* jackson-databind-2.10.2.jar
  + Description:
    - General data-binding functionality for Jackson: works on core streaming API
  + License:
    - <http://www.apache.org/licenses/LICENSE-2.0.txt>
  + Vulnerabilities
    - A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.
* tomcat-embed-core-9.0.30.jar
  + Description:
    - Core Tomcat implementation
  + License:
    - Apache License, Version 2.0: <http://www.apache.org/licenses/LICENSE-2.0.txt>
  + Vulnerabilities
    - The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.
* spring-boot-starter-validation-2.2.4.RELEASE.jar
  + Description:
    - Starter for using Java Bean Validation with Hibernate
    - Validator
  + License:
    - Apache License, Version 2.0: <https://www.apache.org/licenses/LICENSE-2.0>
  + Vulnerabilities
    - \*\* 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.
* hibernate-validator-6.0.18.Final.jar
  + Description:
    - Hibernate's Bean Validation (JSR-380) reference implementation.
  + License:
    - <http://www.apache.org/licenses/LICENSE-2.0.txt>
  + Vulnerabilities
    - A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages.
* spring-core-5.2.3.RELEASE.jar
  + Description:
    - Spring Core
  + License:
    - Apache License, Version 2.0: <https://www.apache.org/licenses/LICENSE-2.0>
  + Vulnerabilities
    - 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.

## 5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial’s software application.

Mitigation planning will begin with updating APIs and utilizing more/ most recent releases. The majority of identified vulnerabilities exist due to utilization of outdated releases. Numerous other vulnerabilities were identified, however false flags were identified and filtered out of this report. The most critical vulnerable dependencies are: spring-boot-2.2.4.RELEASE.jar, log4j-api-2.12.1.jar, tomcat-embed-core-9.0.30.jar, spring-core-5.2.3.RELEASE.jar. Identified issues include the log4j vulnerability (<https://logging.apache.org/log4j/2.x/security.html>). As identified recently this is a major security risk and should be addressed with the highest priority. Following this, the rest of the critical vulnerabilities identified in this segment should be addressed next. Finally, all of the dependency vulnerabilities identified in the previous section, that have not been addressed, should be the next focus.