****

# 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** | **3/20/22** | **Rico Applewhite** |  |

## 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

Rico Applewhite

## 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?

Secure communication is critical for a company like Artemis Financial. They are a financial company which deal with hundreds of customers assets. This make the company a prime target for hackers who are looking to get rich off exploits. Unsecured communications could mean many customers could face big loses and Artemis Financial could lose the trust and bushiness of their clients and shareholders. The company may have international client, so Chances are good that the company produces international transactions. As for governmental restrictions, The Gramm-Leach-Bliley Act requires financial institutions – companies that offer consumers financial products or services like loans, financial or investment advice, or insurance – to explain their information-sharing practices to their customers and to safeguard sensitive data. Like previous mentions the threat of hackers are a present threat now and as technology advance, so will hackers ability to exploit it. Modernization requirement that mus be considered are using open source libraries to strengthen security,but also understanding their vulnerabilities and making sure the systems web application technologies are evolving with the rest of the world.

## 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.

**Input Validation:** Secure input is applicable to Artemis Finance's software application, creating,accessing and editing clients information should be secure.

**APIs:** Artemis Financial software application must communicate with systems of other financial institutions safely.

**Cryptograph:** Proper cryptography need to be used to encrypt and protects Artemis Financial’s and it’s clients data.

**Client/Server:** Data must be secure as it travel from client to server

**Code Error:** Error in the code leaves the system open to vulnerabilities

**Encapsulation:** To Protects internal data when the system is being accessed.

## 3. Manual Review

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

No privilege restrictions through out the code.

User data can be viewed through out the classes, for example the customer.java. Some information is private but the information is then retrieved and displayed.

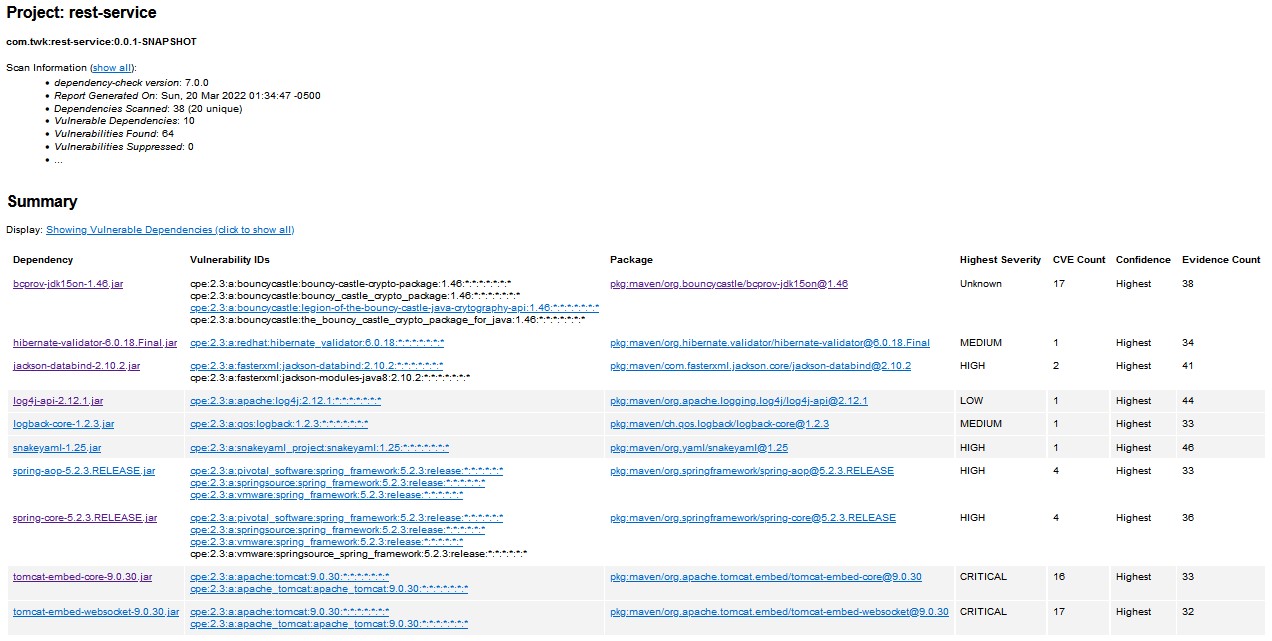
CRUDController.java requests HTML form data to bind to the request parameter. Without validating the HTML form data, harmful data can be introduced into the system.

Error catching in DocData.java needs improving, try/catch may can be exploited.

## 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



**bcprov-jdk15on-1.46.jar**

* CVE-2013-1624

Cryptographic Issues

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.

* CVE-2015-7940

The Bouncy Castle Java library before 1.51 does not validate a point is withing the elliptic curve, which makes it easier for remote attackers to obtain private keys via a series of crafted elliptic curve Diffie Hellman (ECDH) key exchanges, aka an "invalid curve attack."

* CVE-2016-1000338

In Bouncy Castle JCE Provider version 1.55 and earlier the DSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure.

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

* CVE-2020-10693

Improper Input Validation

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.

**jackson-databind-2.10.2.jar**

* CVE-2020-25649

Improper Restriction of XML External Entity Reference

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.

**log4j-api-2.12.1.jar**

* CVE-2020-9488

Improper Certificate Validation

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.

**logback-core-1.2.3.jar**

* CVE-2021-42550

Deserialization of Untrusted Data

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

**snakeyaml-1.25.jar**

* CVE-2017-18640

Improper Restriction of Recursive Entity References in DTDs

The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

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

* CVE-2020-5421

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.

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

* CVE-2019-17569

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.

* CVE-2020-11996

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.**

* CVE-2020-13934

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.

* CVE-2020-13935

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.

* CVE-2020-13943

If an HTTP/2 client connecting to Apache Tomcat 10.0.0-M1 to 10.0.0-M7, 9.0.0.M1 to 9.0.37 or 8.5.0 to 8.5.57 exceeded the agreed maximum number of concurrent streams for a connection (in violation of the HTTP/2 protocol), it was possible that a subsequent request made on that connection could contain HTTP headers - including HTTP/2 pseudo headers - from a previous request rather than the intended headers. This could lead to users seeing responses for unexpected resources.

* CVE-2020-17527

While investigating bug 64830 it was discovered that Apache Tomcat 10.0.0-M1 to 10.0.0-M9, 9.0.0-M1 to 9.0.39 and 8.5.0 to 8.5.59 could re-use an HTTP request header value from the previous stream received on an HTTP/2 connection for the request associated with the subsequent stream. While this would most likely lead to an error and the closure of the HTTP/2 connection, it is possible that information could leak between requests.

* CVE-2020-1935

In Apache Tomcat 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99 the HTTP header parsing code used an approach to end-of-line parsing that allowed some invalid HTTP headers to be parsed as valid. This led 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.

## 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.

First a restriction of privilege is needed to limit certain actions to certain individuals, this will solve the smaller vulnerabilities. One of the biggest vulnerabilities found through static testing is the system’s cryptography, specifically The Bouncy Castle Crypto package. This can be remedy by using a up to date version. The newest version is 1.7. 1.66 and below all have at least one vulnerability as reported on Maven Repository. Our next largest vulnerability reside in the Core Tomcat implementation. Multiple request rather valid or invalid cause many issues here, including denial of service through infinite loops and memory leaks. Proper Input Validation and limiting the number of request a single user or device can perform should be implemented into the code to prevent this. Errors and invalid entries should be handle correctly to avoid the vulnerability of hibernator validation. Spring Core also need to be updated to a version 5.3.14 or higher. Finally make sure the code is error free and all system data is secure.