****

# 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** | **[Date]** | **[Your name]** |  |

## 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

Scot C. Batton

## Interpreting Client Needs

Artemis Financial is looking to modernize the operations of their company and of their web-based software application.

Artemis Financial needs secure messaging and storage of internal and external information regarding customers personal information and financial plans (savings, retirement, investments, and insurance.)

Artemis is also looking to use the most current and effective software security for their business.

They are also in need of Global Rain’s expertise in how to protect their RESTful web application programming interface (API) endpoints.

Artemis is seeking protection from external threats, which include data and information breaches, malware and ransomware attacks, and all other types of external cyber threats that can do harm to the system and its clients.

The company must investigate internal transaction laws and local governments laws in regard to secure communications and restrictions that may be in place.

Evolving use of technologies such as databases and blockchains are modernizations that should be considered for traceability and cloud-based systems for storage.

## Areas of Security

There are several areas of security that Artemis can focus on immediately to begin securing a better network and product for its clients.

**Input Validation.**

Validating that all information input into the system is properly validated can prevent security vulnerabilities such as SQL Injection and cross-site scripting.

**Secure API Interactions.**

With the use of a RESTful API, implementation of secure ways to interact with other API’s can prevent unauthorized access to secure and private data and from leakage of data to those without direct access.

**Encapsulation.**

The use of secure and sophisticated data structures can prevent unauthorized access. Algorithms may also protect data while in transit and at rest from unauthorized access.

**Client/Server.**

The safe passage of information is paramount for any company that is dealing in financial and personal information of their clients. Artemis needs to protect information traveling over networks and can use HTTPS for security during transport.

## Manual Review

After doing a manual review of the code there are multiple vulnerabilities that are of concern and seem to be shared across multiple classes.

**CRUDController Class** has the name parameter passed to the docDate class directly and without input validation.

**Customer Class,** deposit does not have input validation or handle exceptions.

Account balance is also unsecure and is shared as plain text in the code.

**CRUD Class** has no input validation on content 1 or 2

Get() method has no access control and information is shared as plain text.

**DocData** has no input validation or exception handling.

This class also stores in plain text.

Greeting class has no input validation on id and content.

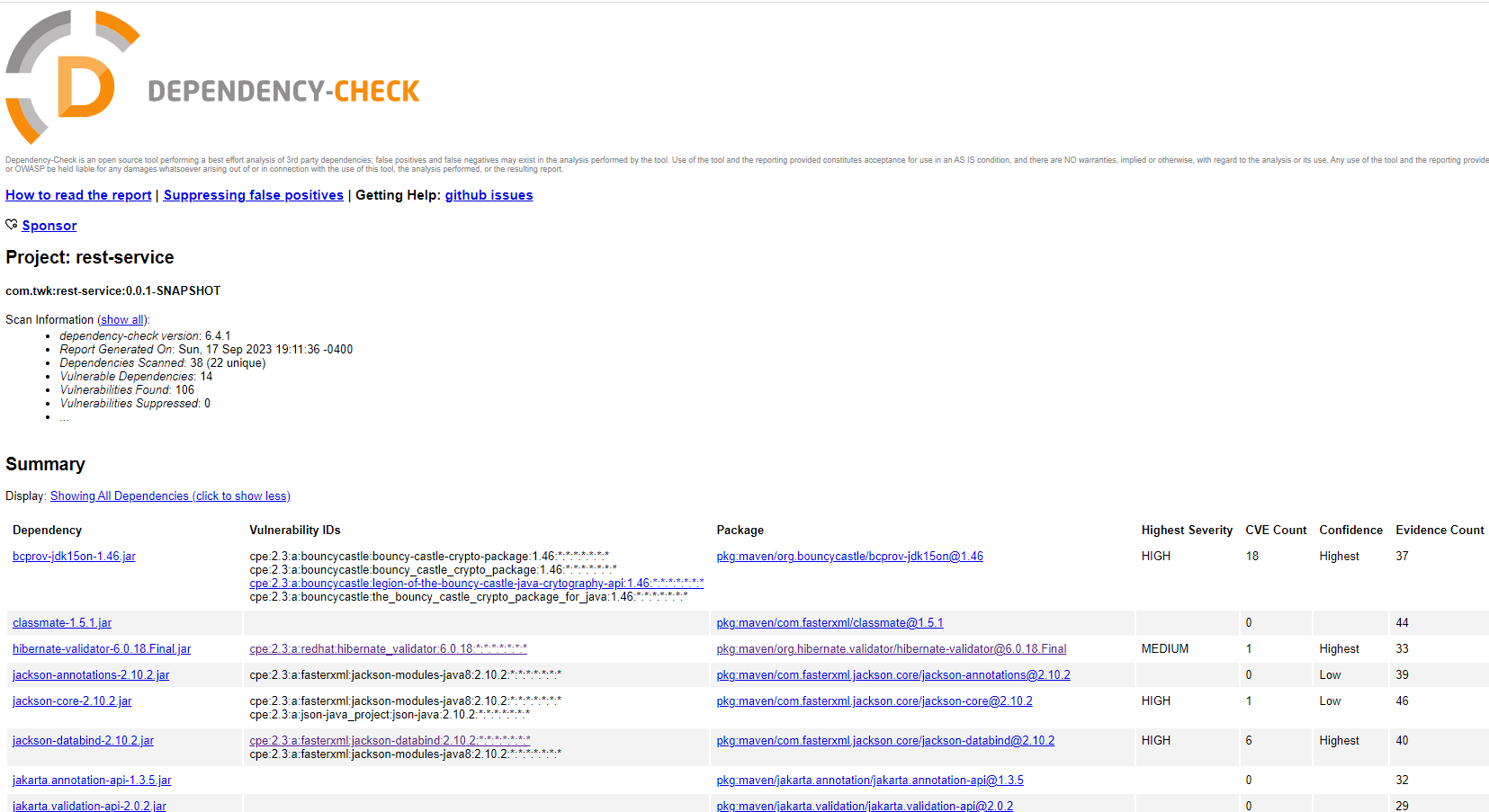
Get(id) and Get(content) have no access control checks.

**Greeting Controller** class has no input validation on name and no error handling.

Greeting() method has no access control checks.

**myDateTime** has SpelExpressionParser() which can lead to code injection

## Static Testing



[logback-core-1.2.3.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l12_864344400c3d4d92dfeb0a305dc87d953677c03c)

logback-core module

1.2.7 and prior versions allow attack to write privileges to edit configurations.

Solution is to upgrade to current version.

[jackson-core-2.10.2.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l5_73d4322a6bda684f676a2b5fe918361c4e5c7cca)

Streaming API

Stack overflow in XML allows DoS attacks.

Upgrade to current version

[jackson-databind-2.10.2.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l6_0528de95f198afafbcfb0c09d2e43b6e0ea663ec)

Data-binding functionality for Jackson

Flaw where it does not have entity expansion secured properly and allows vulnerability to XML external entity attacks.

Update to current version

[log4j-api-2.12.1.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l11_a55e6d987f50a515c9260b0451b4fa217dc539cb)

Apache of Log4j API

Could allow for man in the middle attacks.

Update to current version

[tomcat-embed-core-9.0.30.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l20_ad32909314fe2ba02cec036434c0addd19bcc580)

Core Tomcat Implementation

Refactoring present in 9.0.30 version introduced a regression.

Solution is to update to a current version.

[tomcat-embed-websocket-9.0.30.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l22_33157f6bc5bfd03380ebb5ac476db0600a04168d)

Core Tomcat Implementation

Refactoring present in 9.0.30 introduced regression.

Solution is to upgrade to current version.

[bcprov-jdk15on-1.46.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7)

Java implementation of cryptographic algorithms.

Flaw allows for remote attackers to obtain private keys.

Update to newer version

[hibernate-validator-6.0.18.Final.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492)

Hibernate’s Bean Validation

Flaw allows invalid EL expressions to be evaluated as valid.

Update to current version

[spring-boot-starter-web-2.2.4.RELEASE.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l16_ec75d01d212b5229c16d872fb127744c0ed46ed8)

starter for building web

v2.2.11 and prior subject to temporary directory hijacking

solution is to update to current version.

[spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l15_225a4fd31156c254e3bb92adb42ee8c6de812714)

spring-boot

vulnerable to temporary directory hijacking prior to v2.2.11

solution is to update to current version.

[spring-core-5.2.3.RELEASE.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l17_3734223040040e8c3fecd5faa3ae8a1ed6da146b)

Spring core

Suffers from potential remote code execution.

[spring-web-5.2.3.RELEASE.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l18_dd386a02e40b915ab400a3bf9f586d2dc4c0852c)

Spring Web

Suffers from potential remote code execution.

Update to current version

[spring-webmvc-5.2.3.RELEASE.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l19_745a62502023d2496b565b7fe102bb1ee229d6b7)

Spring Web MVC

5.3.0 and older versions, possible for user to provide malicious input.

Solution is to update to current version.

[snakeyaml-1.25.jar](file:///C:\Users\satur\OneDrive\EclipseJava2\CS%20305%20Project%20One%20Code%20Base%20(1).zip_expanded\rest-service\target\dependency-check-report.html#l14_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421)

parser and emitter for Java

allow entity expansion during a load operation.

Update to current version.

## Mitigation Plan

After careful consideration of the static testing report and a manual review of the code, there are steps that can be taken to mitigate the identified security vulnerabilities and to allow the clients needs to be met.

* In the dependency-check report, each of the vulnerabilities showed that the current versions being used were out of date and did not provide security effectiveness. The first step in mitigating risk to the company will be to update everything involved to its most current version.
* Each class within the code base did not account for input validation, so we will implement input validation for all the user-supplied inputs within the code base.
* Input must also be sanitized to prevent injection attacks.
* Ensure that data in not stored in plain text and that access controls are in place.
* Implement access controls, authentication, and authorization.
* Program proper error handling.