****

# 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** | **March 19th 2022** | **Luis Lansberry** |  |

## 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

Luis Lansberry

## 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?

The company benefits from secure communications by giving its clients the needed security to invest their money with them, without fear of loss or hack. It also saves the company from lawsuits in the case a leak happened due to a lack of secure communication

Since the company deals with investments it produces many international transactions in the form of purchasing or selling foreign bonds, stocks and ETFs.

Yes, each state and international body has different regulations for secure communication, such as what information can be stored and for how long is it on record, along with who gets to access said information.

External threats that could affect Artemis Financial would-be hackers, phishers and bugs encountered within their application. Hackers and phishers would be interested in getting clients financial information or their personal information such as social security numbers. The bugs are trickier to tackle, but as an example we can see how RobinHood a popular trading app had a bug which gave its users infinite leverage.  
Also, using a rest API can cause security vulnerabilities such as injections attacks and parameter tampering which must be accounted for when designing the API.

I believe that by implementing open-source libraries we will modernize the application, these libraries are constantly updated and patched when new bugs are discovered, they could be used for some of the public operations while ensuring they don’t have access to sensitive data. Evolving web technologies such as NFTs can be used to ensure the authenticity of certain securities, while also making the ownership data public at real time instead of the current waiting period for disclosing securities which makes the information worthless by the time it is public. These will ensure the app is modernized for future clients.

## 2. Areas of Security

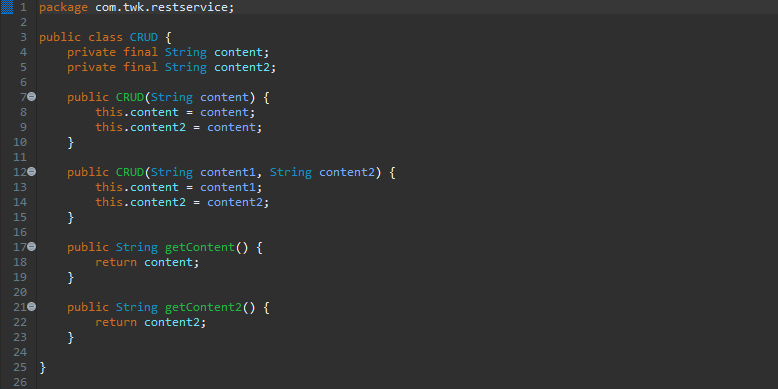
Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

The relevant areas for Artemis Financial’s software are:

* Input Validation: The reasoning behind this is to avoid injection attacks while also defining the parameters by which user input will be validated and how the software must act in case it is invalid input.
* APIs: This selection is a little obvious, since we are going to use a rest API, we must ensure all interactions are secure and all possibilities accounted for.
* Cryptography: Encryption must be used for secure communication, and it is also required by law in some of the markets we will explore.
* Client/Server: We must ensure client server communications are safe and that verification steps are taken to ensure that no client had their account stolen.
* Code Error: As mentioned before, we must account for all possibilities when the software is ran, accounting for these errors ensures there are no holes in the code by which bad actors can access.
* Encapsulation: We must ensure our data and data structures are isolated in case one was compromised, this way we don’t lose everything in the case of a breach.

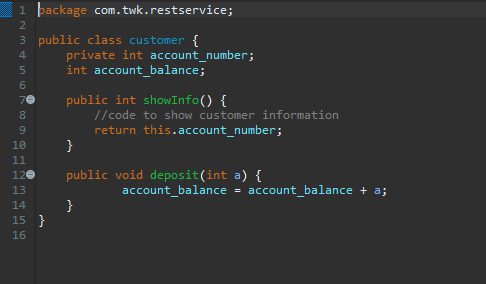
## 3. Manual Review

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

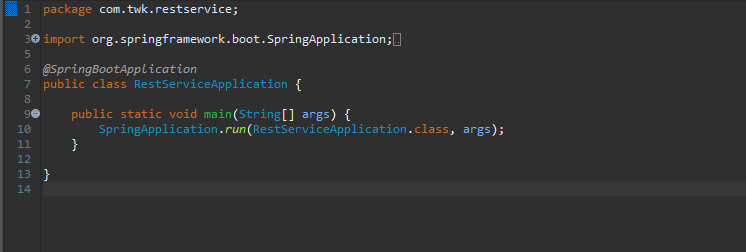




Here we see there are no limitations for user input and no accounting for special characters. There are no parameters on what to do in case of errors or invalid input types.



There is no accounting for unexpected errors and no validation processes to ensure valid access was granted.



No accounting for unexpected errors and the API is just set to run continuously.

Service does not use HTTPS either.

## 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

Description: In the Bouncy Castle JCE Provider version 1.55 and earlier the ECIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.  
Solution: Before applying the update, back up your existing installation, including all applications, configuration files, databases and database settings, and so on.

-hibernate-validator-6.0.18.Final.jar

Description: 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.  
Solution: Update software to hibernate-validator 7.0.0.Alpha2

-jackson-databind-2.10.2.jar

Description: jackson-databind before 2.13.0 allows a Java StackOverflow exception and denial of service via a large depth of nested objects.  
Solution: Optimize software.

-log4j-api-2.12.1.jar  
Description: Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP data source URI when an attacker has control of the target LDAP server. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2.  
Solution: Upgrade to version 2.17.1.

- 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.  
Solution: Upgrade software to a newer version

- snakeyaml-1.25.jar  
Summary: The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.  
Solution: Upgrade software to a newer version.

- spring-aop-5.2.3.RELEASE.jar  
Summary: 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.  
Solution: Upgrade software to a newer version.

- spring-core-5.2.3.RELEASE.jar  
Summary: 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.  
Solution: Upgrade software to a newer version.

- tomcat-embed-core-9.0.30.jar  
Summary: The fix for bug 63362 present in Apache Tomcat 10.1.0-M1 to 10.1.0-M5, 10.0.0-M1 to 10.0.11, 9.0.40 to 9.0.53 and 8.5.60 to 8.5.71 introduced a memory leak. The object introduced to collect metrics for HTTP upgrade connections was not released for WebSocket connections once the connection was closed. This created a memory leak that, over time, could lead to a denial of service via an OutOfMemoryError.  
Solution: Upgrade your tomcat9 packages.

- tomcat-embed-websocket-9.0.30.jar  
Summary: The fix for bug 63362 present in Apache Tomcat 10.1.0-M1 to 10.1.0-M5, 10.0.0-M1 to 10.0.11, 9.0.40 to 9.0.53 and 8.5.60 to 8.5.71 introduced a memory leak. The object introduced to collect metrics for HTTP upgrade connections was not released for WebSocket connections once the connection was closed. This created a memory leak that, over time, could lead to a denial of service via an OutOfMemoryError.

Solution: Upgrade your tomcat9 packages.

## 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.

We should update all dependencies listed above, implement an authentication method for users accessing accounts and strange behavior detectors. We should begin using HTTPS protocols and incorporate parameter limitations for user input and unexpected error catches.