****

# 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** | **11/13/21** | **Jacob Valdiviez** |  |

## 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

Jacob Valdiviez

## 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?

Artemis Financial is a financial consultant company that specializes in producing specialized financial plans for savings, investments, insurance and retirement. Our client is planning to modernize their operations. This may include: secure APIs and current secure coding practices. Security is important for Artemis Financial. They want the most current and effective software security. They currently have a RESTful API, and would like to have the organization protected from vulnerabilities.

## 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:

The software application would consider secure input when users logon to the system and interact with the system. This should prevent any unwanted code injections into the system.

API:

The software application is going to utilize a RESTful API. This would ensure that client and server requests are secure.

Cryptography:

Although the client hasn’t mentioned anything about secure messages, but this should be necessary for a secure connection to financial data. This should protect customer and company information from being viewed.

Client / Server:

A well built client-server architecture is necessary to conform to the criteria of a RESTful API. This is also necessary for a secure system dealing with financial information.

Encapsulation:

This ensures that the system is layered and organized to what server is responsible for security. This will make information into hierarchies and make that information invisible to unwanted authorization.

Code Quality:

The client wants a “modernized” software. This includes: best coding practices with security and readability. This also lowers the chances of developing bugs that may be exploited by attackers.

## 3. Manual Review

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

The mysql database username and password needs to be updated to a more secure password, and a secure username is needed too. The method for depositing money to an account should accept a float instead of an integer. Also, showing the customer’s information should not be a public method. A random name should be generated for a directory reference. Especially, “/read” is used to call the database. This could be enumerated by an attacker.

## 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

A dependency check was performed to find any vulnerable dependencies.

bcprov-jdk15on-1.46.jar

The implementation in the Bouncy Castle library does not consider timing side-channel attacks on MAC check operation. It allows remote attackers to do distinguishing attacks and plaintext-recovery attacks. One of the ways to prevent such attacks is to not map categories. (NIST, n.d.-a)

log4j-api-2.12.1.jar

Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. SMTPs can be intercepted by a man-in-the-middle attack which leaks any log messages sent by the appender. To prevent the attack upgrade the version to 2.13.2. (NIST, n.d.-b)

snakeyaml-1.25.jar

The alias feature in Snake YAML allows entity expansion during a load operation. To fix it, upgrade to version 1.26. (NIST n.d.-c)

jackson-databind-2.12.2.jar

Entity Expansion was not secured properly. This allows vulnerability to XML external entity attacks. To fix it, upgrade to version 3.9. (NIST, n.d.-d)

tomcat-embed-core-9.0.30.jar

The refactoring in the Apache Tomcat introduced a regression. The result of the regression was that Transfer-Encoding headers were incorrectly processed leading to a HTTP Request Smuggling. This happens to a specific kind of reverse proxy. To fix it upgrade tomcat to 9.0.31 or later. (NIST, n.d.-e)

hibernate-validator-6.0.18.Final.jar

A bug in the message processor enables invalid EL expressions to be evaluated as if they were valid. This bug allows attackers to pass through input sanitation controls in error messages. To fix it, upgrade hibernate to 6.0.20. (NIST, n.d.-f)

spring-core-5.2.3.RELEASE.jar

The protections against RFD attacks can be bypassed depending on the browser used through the use of jsessionid path. It is recommended to update the Spring components to at least 5.1.18. (NIST, n.d.-g)

spring-jcl-5.2.3.RELEASE.jar

The protections against RFD attacks may be bypassed depending on the browser used through the use of jsessionid path. It is recommended to update the Spring components to at least 5.1.18. (NIST, n.d.-g) This is the same CVE as the Spring core.

## 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.

Through manual review, I noted that the mysql password and username are not correctly configured. To mitigate the vulnerability of unauthorized access of the database, the database must be configured with a secured username and password. This includes random characters, numbers and special characters. Always maintain good coding practices. This includes: making sure fields are private, and sensitive information should not be accessed by public methods. Categorize information using random names and don’t map them. It is found that the directory “/read” accesses the database. This is especially dangerous for one of the dependency vulnerabilities found in Bouncy Castle. A dependency check was ran in the code to check if there are any vulnerable dependencies. The best way to mitigate them, is to upgrade the dependencies to a newer version that has the vulnerability resolved. Always watch out for input validation. Check for escaped characters or code injections. Don’t rely on the dependencies to check for you. Add that extra layer of security to ensure. Utilize a RESTful API. Ensure the application follows the criteria of a RESTful API. Ensure information received and transferred between client and server is encrypted. Using hash functions and using encryption certificates can help ensure the information is encrypted. This protects the financial data of the customer and of the company. Make sure that there are layers of code where unintended areas of the program cannot access. Keep things private and public where they are supposed to be. Coding methods that only do what they are functioned to do. This can also ensure that good coding practices are met, and less confusing to read. Thus, prevents bugs or flaws in the code that an attacker can potentially exploit.

NIST. (n.d.-a). *NVD - CVE-2013-1624*. National Vulnerability Database. Retrieved November 16, 2021, from https://nvd.nist.gov/vuln/detail/CVE-2013-1624

NIST. (n.d.-b). *NVD - CVE-2020-9488*. National Vulnerability Database. Retrieved November 16, 2021, from https://nvd.nist.gov/vuln/detail/CVE-2020-9488

NIST. (n.d.-c). *NVD - CVE-2017-18640*. National Vulnerability Database. Retrieved November 16, 2021, from https://nvd.nist.gov/vuln/detail/CVE-2017-18640

NIST. (n.d.-d). *NVD - CVE-2020-25649*. National Vulnerability Database. Retrieved November 16, 2021, from https://nvd.nist.gov/vuln/detail/CVE-2020-25649

NIST. (n.d.-e). *NVD - CVE-2019-17569*. National Vulnerability Database. Retrieved November 16, 2021, from https://nvd.nist.gov/vuln/detail/CVE-2019-17569

NIST. (n.d.-f). *NVD - CVE-2020-10693*. National Vulnerability Database. Retrieved November 16, 2021, from https://nvd.nist.gov/vuln/detail/CVE-2020-10693

NIST. (n.d.-g). *NVD - CVE-2020-5421*. National Vulnerability Database. Retrieved November 16, 2021, from https://nvd.nist.gov/vuln/detail/CVE-2020-5421