****

# 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** | **05/22/2021** | **Richard Rehkemper** | **Vulnerability Assessment report** |

## 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

Richard Rehkemper

## 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:

* Security in the financial realm is of upmost importance. So secure communication between the server and client must be a high priority in protecting financial interactions and client data.
* Transactions may potentially occur internationally and also must be a consideration when developing a secure information transmission.
* Government entities are a consideration when dealing with financial interactions and records. After 180 days in a database, the US government is allowed to access the data without a warrant (<https://us.norton.com/>).
* Current threats involve any kind of external interference with the transfer of communications between the server and the client, or a breach of data storage. The threats can happen in many different forms, such as SQL injection to pull data, or man in the middle (MITM) attacks that interfere with data transference. Future concerns deal with the complexity of the attacks, so updated frameworks, libraries, and scans should be part of the routine maintenance.
* Modernization of a company includes clients accessing their profiles from numerous devices and possibly different operating systems. The convenience of customer interactions comes at a price, however, and adequate validation measures need to be taken while not compromising the end-user experience. The Software-As-A-Service model, for example, can give the end user a simple product that is heavily coded on the back end (https://www.strategy-business.com/).

## 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 – When a client enters their credentials, it needs to be handled properly. If the username and password don’t match, for example, access should not be allowed. This goes hand in hand with proper exception control and should not return any values that put the code, systems, or data at risk.

Cryptogrophy – When data is being stored, or transmitted, it should be encrypted to better protect the information. The use of a secure protocol (https://) is almost necessary and should be expected when dealing with businesses that involve personal data or financial transactions.

Client/Server – The communication and distribution of information from the client and the server needs to remain a high priority in the overall security of the systems. The data often has sensitive information that could cause harm to the individual client or the business if breached. Also, the use of the principle of least privilege should be implemented to establish privilege levels to help sustain proper access to clientele.

Code Quality – The quality of the code plays a few major roles in securing the software. A high quality code is easy to read for the back end developers but has also been designed with security as a high priority since then very beginning, making the overall product more significantly more secure.

Code Error – There are major security concerns in the event that a code has an error or is mishandled in the event of an error. Improper handling of exceptions can open a flood gate of breaches and can present users with access points they shouldn’t have. If a mishandled exception gets manipulated it can be disastrous to the sensitive data.

## 3. Manual Review

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

* There is a lack of exception handling throughout the code. This interferes with code quality and client/server protection. Also, a lack of exception handling can open the door for SQL injection, which can pull up archived files for third party review. Example: in DocData.Java there is a very basic catch on lines 28-30 that could benefit from being more extensive.
* There is a lack of input limiters. Input limiters reduce the attack surface area making it more difficult for attackers and relates to the input validation aspect of the VAPFD. In the Greeting.java the id is indicated as a long type. This may be acceptable for that id length, however there are no limiters to the length of the string provided.
* In the Customer.java class line 5 account balance is a public integer. This should be private to protect the integer value unless accessed through an encapsulated method. This is a code quality and possibly a code error instance of the VAPFD and would allow data to be sensitive data to be divulged without going through the proper channels.

## 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

Below I have included a screenshot of the dependencies identified in the static testing check.

A picture containing text, screenshot, indoor, computer

Description automatically generated

**bcprov-jdk15on-1.46.jar - CVE-2013-1624 NVD published date: 02/05/2013 NVD modified date: 10/30/2018.**

an outdated library would allow a side-channel attack on a “non-compliant” mac. The library should be updated to version 1.8 or greater for the Bouncy Castle Java.

**hibernate-validator-6.0.18.Final.jar - CVE-2020-10693 NVD published Date: 05/06/2020 and NVD modified date: 03/02/2021.**

There is a code bug in the hibernator validator that could allow hackers to get into the system without having sufficient credentials. Typically changing the code can resolve this issue, making it a quality of code problem.

**jackson-databind-2.10.2.jar -** [**CVE-2020-25649**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649)**NVD publish date: 12/03/2020 NVD modified date: 5/14/2021.**

The FasterXML Jackson Databind did not properly secure entity expansion. This could lead to attackers exploiting data. To resolve this issue, removal of the protocol or the functionality through code changes are suggested.

**log4j-api-2.12.1.jar -** [**CVE-2020-9488**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488) **NVD Published Date: 04/27/2020 NVD Last Modified: 05/10/2021.**

There is an improper certificate validation of the host allowing potential Man-In-The-Middle attacks. SmptAppender should be upgraded to version 2.13.2.

[**snakeyaml-1.25.jar**](file:///C:\Users\owner\eclipse-workspace\rest-service\target\dependency-check-report.html#l13_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421) **-** [**CVE-2017-18640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640) **NVD Published Date: 12/11/2019 NVD Last Modified: 04/19/2021.**

The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation. Entity expansion , as mentioned earlier, opens the database up to attackers exploiting data. To resolve this issue, removal of the protocol or the functionality through code changes are suggested.

**spring-core-5.2.3.RELEASE.jar** - [**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421) **NVD Published Date:**  
**09/19/2020** **NVD Last Modified:** **05/13/2021.**

Depending on the browser, earlier versions of the spring core database could allow RFD attacks. RFD attacks allow the attacker to gain control of a virtual machine. Mitigation would involve updating the spring core database and library to versions greater than 5.28.

[**tomcat-embed-core-9.0.30.jar**](file:///C:\Users\owner\eclipse-workspace\Module2.1\target\dependency-check-report.html#l16_ad32909314fe2ba02cec036434c0addd19bcc580) **and** [**tomcat-embed-websocket-9.0.30.jar**](file:///C:\Users\owner\eclipse-workspace\Module2.1\target\dependency-check-report.html#l16_ad32909314fe2ba02cec036434c0addd19bcc580) **–** [**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569)**and** [**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569)**NVD Published Date:** **02/24/2020** **NVD Last Modified:** **01/20/2021**

The tomcat core needs to be updated. 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 (nvd.nist.gov).

## 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.

* Review of the code to ensure that proper exception handling doesn’t allow the chance for SQL injecting or input errors that would allow validations to be bypassed.
* Review the code for any entries to be limited in length (assists with the entity expansion issues) and for the returns to be encrypted as opposed to “plain-text”.
* Update the Spring core library
* Update the Bouncy Castle Java library
* SmptAppender should be upgraded to version 2.13.2.
* Verify code quality to ensure that sensitive data is set to private unless encapsulated.

Resources:

Cooper, L., & Vyas, M. (2019, February 21). 10 principles for modernizing your company’s technology. strategy+business. <https://www.strategy-business.com/article/10-Principles-for-Modernizing-Your-Companys-Technology?gko=6ff05>

*What are some of the laws regarding internet and data security?* (n.d.). Official Site | Norton™ - Antivirus & Anti-Malware Software. https://us.norton.com/internetsecurity-privacy-laws-regarding-internet-data-security.html