****

# 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** | **11/8/2022** | **Thomas Fiske** | **Adding Interpreting Client needs, Areas of Security, Manual Review, Static Testing, and Mitigation Plan.** |

## 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

Thomas Fiske

## Interpreting Client Needs

Artemis Financial is a financial consulting company that “develops individualized financial plans for their customers. The financial plans include savings, retirement, investments and insurance.” (SNHU, n.d.) Artemis Financial’s app is web-based and uses REStful API to function.

It is important for a company that handles money to have secure transactions and private information. The first reason being that money has a huge influence on a person’s life and not handling money properly can cause devastating effects to that person. Another reason would be for the companies own financial health and reputation. Keeping information private and in assessable to unprivileged users is vital to the company maintaining its value. The reason for this is because if the private information of the company’s users is not secure, then no one will want to use that company. Another reason would be that the company could be on the hook for breaches of private data if they are found to be negligent in handling that data.

The company handles sensitive data and makes sensitive transactions that must be protected under law. The U.S Securities and Exchange Commission (SEC) has a regulation called “Privacy of Consumer Financial Information (Regulation S-P).” “The Act also requires the Commission to establish for financial institutions appropriate standards to protect customer information (U.S. SEC, 2001).” This act ensures that costumer’s private information is secured when it is given to an investment company. It also states that costumers must be shown a privacy policy so that they know the company’s policy on protecting their information (U.S. SEC, 2001).

External threats are a major concern because of the ramifications associated with what a breach of security can mean for the costumer and the company. SQL injection is a common example of a threat that is used on websites in order to bypass their account input. For example, when logging into the website or application, a bad actor could add pieces to the input that cause the application to have unexpected errors. The bad actor can make use of the errors and may be able to work their way into someone’s account using some tweaking (Manico, J., & Detlefsen, A. 2014). There is also the threat of phishing, where malicious links will be sent through a message system like email that download malware onto their device. Brute force password attacks where bad actors try to guess a user’s password. This can be done with a program or manually. Session hijacking where a bad actor will hijack the session that is being used, basically stealing the connection between the application and the server (ERMProtect, 2022) (Manico, J., & Detlefsen, A. 2014). These are just some examples of the different external threats that are facing companies today. The nature of these threats is ever evolving and it is important to continue learning new techniques to keep up with the new ways that hackers will try to exploit the company.

Modernization can be very beneficial for companies but it can also leave their applications and websites open to more vulnerabilities than in the past. The more technology develops the more ways that it can be manipulated. As applications continue to grow, expand and become more complicated, they will have more vulnerabilities that will need to be discovered.

## Areas of Security

There are several areas of security that are important to this application. The application will use input because the users will need to make accounts in order to keep track of their account numbers, balances and other information. This means that input validation is absolutely vital to this application. As stated previously, user input can be manipulated with SQL attacks or similar. This means that the application will need a way to validate the correct input from the user, and also reject anything that does not fit that criterion. If an application has a reject first functionality, where it only accepts valid input and rejects everything else, this will help protect it from attacks (Manico, J., & Detlefsen, A., 2014).

This application will be using RESTful API, that means that API is a vulnerability for this application. API’s can be attacked using SQL injection, another reason why input validation is very important to this application. There are other attacks that API’s can be attacked with as well and making sure that the API is secure for the application is important to maintain the security of the application.

Cryptography is used to secure information that is only supposed to be seen by a select few people. This would be important to this application as well because it will be dealing with a lot of sensitive information and transactions that will need to be as secure as possible. Using cryptography is one of those ways to make it more secure.

The client/server is important to this application as well. It is important to secure the connection between the client and the server. The reason for this is because if it is not, a bad actor could use the vulnerability to intercept sessions or requests between the client and the server (Manico, J., & Detlefsen, A. 2014). If that were to happen, then they would have access to information and controls that they should not be allowed to have access to.

Code error is also another important piece of the security. It is important to not give bad actors the errors that they would need to be able to find a way to infiltrate the application. Bad actors can take advantage of what it stated in the error messages and can utilize that for different attacks Manico, J., & Detlefsen, A. 2014).

One of the most important parts of this application is code quality. The application will run from code, and its parts of its security will come from its code as well. That is why it is important to have good code quality throughout the application. Bad code quality will leave the application vulnerable to attacks because it will be easier for bad actors infiltrate the application.

## Manual Review

These are the problems that I found inside of the code base for Artemis Financial’s application.

* The CRUD controller class uses “business\_name” as a parameter for the request. This should be changed to the business’s actual name (Artemis Financial).
* DocData.java uses test for a data base and root for a username and password. The data base should be changed to reflect that of the data based used for the application. Root should be changed accept the username and password of the user.
* Dependencies need to be updated.
* There is no authentication for the user to check that their credentials match what is in the data base when logging in.

## Static Testing

Below is a list of dependencies that have vulnerabilities that were found during the static test.

* spring-boot-2.2.4.RELEASE.jar
  + CVE-2013-4152
  + CVE-2013-7315
  + CVE-2016-1000027
  + CVE-2018-11039
  + CVE-2018-11040
  + CVE-2018-1257
  + CVE-2020-5421
  + CVE-2022-22950
  + CVE-2022-22965
  + CVE-2022-22968
  + CVE-2022-22970
  + CVE-2022-27772
* bcprov-jdk15on-1.46.jar
  + CVE-2013-1624
  + CVE-2015-6644
  + CVE-2015-7940
  + CVE-2016-1000338
  + CVE-2016-1000339
  + CVE-2016-1000341
  + CVE-2016-1000342
  + CVE-2016-1000343
  + CVE-2016-1000344
  + CVE-2016-1000345
  + CVE-2016-1000346
  + CVE-2016-1000352
  + CVE-2017-13098
  + CVE-2018-5382
  + CVE-2020-0187
  + CVE-2020-26939
* logback-core-1.2.3.jar
  + CVE-2021-42550
* snakeyaml-1.25.jar
  + CVE-2017-18640
  + CVE-2022-25857
  + CVE-2022-38749
  + CVE-2022-38750
  + CVE-2022-38751
  + CVE-2022-38752
* log4j-api-2.12.1.jar
  + CVE-2020-9488
  + CVE-2021-44228
  + CVE-2021-44832
  + CVE-2021-45046
  + CVE-2021-45105
* jackson-databind-2.10.2.jar
  + CVE-2020-25649
  + CVE-2020-36518
  + CVE-2022-42003
  + CVE-2022-42004
* tomcat-embed-core-9.0.30.jar
  + CVE-2019-17569
  + CVE-2020-11996
  + CVE-2020-13934
  + CVE-2020-13935
  + CVE-2020-13943
  + CVE-2020-17527
  + CVE-2020-1935
  + CVE-2020-1938
  + CVE-2020-8022
  + CVE-2020-9484
  + CVE-2021-24122
  + CVE-2021-25122
  + CVE-2021-25329
  + CVE-2021-30640
  + CVE-2021-33037
  + CVE-2021-41079
  + CVE-2021-43980
  + CVE-2022-29885
  + CVE-2022-34305
  + CVE-2022-42252
* Spring-boot-starter-validation-2.2.4.RELEASE.jar
  + CVE-2022-27772
* hibernate-validator-6.0.18.Final.jar
  + CVE-2020-10693
* spring-core-5.2.3.RELEASE.jar
  + CVE-2016-1000027
  + CVE-2020-5421
  + CVE-2021-22060
  + CVE-2021-22096
  + CVE-2021-22118
  + CVE-2022-22950
  + CVE-2022-22965
  + CVE-2022-22968
  + CVE-2022-22970
  + CVE-2022-22971

Most of these vulnerabilities can be solved with updating the dependencies that are being used. For example, CVE-2013-1624 in bcprov.jdk.146.jar was caused by the implementation between Bouncy Castle Java 1.48 and C#’s library before 1.48. This implementation caused a timing issue which allowed remote hackers to conduct attacks. After the dependency was fixed, updating the dependency would help get rid of this vulnerability (National Vulnerability Database, 2013)

Spring-boot.2.2.4.RELEASE.jar has several vulnerabilities that are recent. There have been four different vulnerabilities found with this dependency this year. These vulnerabilities range from being vulnerable to a denial-of-service attack because an older version relied on data binding to set a multi part file to a field in a model object, to being vulnerable to remote code execution through data binding (National Vulnerability Database, 2022). Tomcat-embed-core-9.0.30.jar, Jackson-databind-2.10.2.jar, spring-boot-starter-validation-2.2.4.RELEASE.jar are just a few examples of the dependencies that have had vulnerabilities found this year. These dependencies should be regularly checked for updates to continue to fix these vulnerabilities.

## Mitigation Plan

The best way to mitigate security vulnerabilities from harming the company is to continually static and manual test the code in order to find vulnerabilities. In the manual and static tests, several vulnerabilities were found. These tests will increase in importance as the application grows and becomes more complicated. Dependencies need to be updated to the most recent version in order to fix the vulnerabilities that are already caught by other developers. This means that spring-boot-2.24.RELEASE.jar should be updated to its current version of 2.7.5., spring-core.5.2.3.RELEASE.JAR should be updated to version 5.3.32, bcprov-jdk15on-1.46.jar should be updated to 1.72. Every other dependency in this application should be updated to its most current version in order to eliminate the vulnerabilities found in the static test.

In order to change the vulnerabilities found in the manual review, a developer will have to go in and change the code in order to make it more secure. One of the biggest things that needs to be added is a way to validate the user information when they log into the application. This security will help stop bad actors from accessing other users accounts and doing damage within those accounts. The CRUD controller class should have its parameter changed in its @requestparam to the name of the business. The reason for this is to leave less room for SQL injection attacks. DocData.java needs to be able to accept usernames and passwords. Right now, they are both set to root, instead of verifying the correct username and password.

## References

SNHU. (n.d.). *Project One Guidelines CS-305*. Sign in to your account. Retrieved November 8, 2022, from https://learn.snhu.edu/d2l/le/content/1196944/viewContent/20833867/View

U.S. SEC. (2001). *Final rule: privacy of consumer financial information (regulation S-P)*. Final Rule: Privacy of Consumer Financial Information (Regulation S-P). Retrieved November 8, 2022, from https://www.sec.gov/rules/final/34-42974.htm

Manico, J., & Detlefsen, A. (2014). *Iron-clad java: Building secure web applications; best practices for secure Java Web Application Development*. Oracle Press.

*External vs. internal cybersecurity risks: Know the difference - cybersecurity: Digital Forensics: Crypto Investigations*. ERMProtect. (2022, October 19). Retrieved November 8, 2022, from https://ermprotect.com/blog/external-vs-internal-cybersecurity-risks-know-difference/

National Vulnerability Database. (1999-2022). NVD. Retrieved November 13, 2022, from https://nvd.nist.gov/vuln/search/results?form\_type=Advanced&results\_type=overview&search\_type=all&cpe\_vendor=cpe%3A%2F%3Avmware&cpe\_product=cpe%3A%2F%3Avmware%3Aspring\_framework&cpe\_version=cpe%3A%2F%3Avmware%3Aspring\_framework%3A2.2.4