****

# 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** | **09/15/2022** | **Caleb VanDerMaas** |  |

## 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

Caleb VanDerMaas

## Interpreting Client Needs

1. What is the value of secure communications to the company?
   1. “Security is everyone’s responsibility”, this is the motto of the company, Global Rain, that I work for. In this example the client, Artemis Financial, is looking to protect their application that is responsible for automating personal financial plans for peoples’ savings, retirement, investments, and insurance. Secure communications for the company is their bread and butter. Without it, they could not build their users trust and no one would want you use their services.
2. Does the company make any international transactions?
   1. The scenario doesn’t make any specific mention of international transactions; however, I don’t think it’s a bad idea to consider protecting for international transactions given this is a web based application and will be open to most everyone on the internet. The only reason I would not consider spending time on this is if the user must be associated with some kind of US bank in which case, we would not need to consider any international transactions.
3. Are there governmental restrictions about secure communications to consider?
   1. There are many governmental restrictions around peoples’ personal banking information that must be respected in order to not be prosecuted by the law. For example, the Federal Deposit Insurance Corporation has a rulebook for the dos and don’ts of handling banking information.
4. What external threats might be present now and in the immediate future?
   1. There are plenty of black hat hackers and criminals that would love to get their hands on people’s personal banking information. One of the easiest ways for these hackers to get their hands on this info is through an interception. While the data is being transferred over the internet, the hacker can intercept this information and use it for criminal activity. Thankfully, encryption and public and private keys allow for a safer means of data transfer. Since Global Rain’s motto is that security is everyone’s responsibility, it is also important for us to inform the users of the risks and teach them how to conduct themselves safely on the internet, or at least while they are using this application.
5. What are the modernization requirements that you must consider? For example:
   1. The role of open-source libraries
      1. Open source is a great feature of modern technology that has helped for rapid technological industrialization. However, these projects are free for anyone to construct including bad actors and therefor represent a potential threat to keep in mind. At the very least, this means that all open source libraries should be monitored regularly for potential vulnerabilities and teams should be able to respond urgently to any of these threats at any given moment.
   2. Evolving web application technologies
      1. New territory always heads new vulnerability. Treading cautiously into these new settings is always the best approach. It’s easier and safer to stick with the established technologies that have already withstood the brute force of global adaptation for long periods of time. It’s not impossible to step into the new technologies, but it would be wise to keep critical information and infostructure away from these new technologies until they have built a robust enough scene and minimum securities have been ensured.

## Areas of Security

* + - 1. Input Validation
         1. Input validation is a very important part of this application. Users will be able to input a lot of data into the application and it’s our job to make sure that these points are secure and don’t pose any security risk.
      2. APIs
         1. Artemis Financial has a RESTful web API that we will need to make sure is free of any vulnerabilities. This includes checking the latest version of each language that is being used, as well as testing the endpoints to make sure that they are secure.
      3. Cryptography
         1. Cryptography is the bread and butter of this application. There will necessarily be a sharing of sensitive information over the internet. To secure this data we can use cryptography. In a very broad generalization, we will be scrambling the data before we send it out and only the recipient with the correct key will be able to unscramble this data. We will also use cryptography to keep any sensitive that is saved to the DBs encrypted so that bad actors cannot do anything with the data if it is breached.
      4. Client/Server
         1. The security around our client and servers needs to be strong so that the data cannot be intercepted before or after it is sent.
      5. Code Error
         1. Any potential code errors that are not handled in a web application can lead to disastrous effects. Code errors always need to be handled in a web application so that bad actors cannot take advantage of any vulnerabilities hidden behind these errors.
      6. Code Quality
         1. I think code quality is always an important concept to keep in mind for security purposes, however I don’t think it is as important as some of the other security principles.
      7. Encapsulation
         1. Again, I think encapsulation is important, however for this application I don’t think it is very relevant.

## Manual Review

* One of the biggest things that sticks out to me right away is that there is no error/exceptions handling for the REST service application that is being imported and used.
* Within the CRUD.java file, there is a confusing duplication of the content, content1, and content2 variables. This is against the guidline 0-2 fundamentals: avoid duplication.
* In the CRUDController, there is another example of a lack of error/exception handling. Customer.java is full of confidential information that is labeled very obviously and not marked with any sort of sensitive information marker. This could easily call attention to bad actors and lead to an exploitation of this data.
* In DocData, there is a return of sensitive information made as well as a lack of robust error handling and connection checking.
* In CRUD, Greeting, and GreetingController, there is a good use of immutable value types which is preferred in the guideline however, this could have been continued with the variables in the customer.java as well as DocData.java.

## Static Testing

bcprov-jdk15on-1.46.jar

classmate-1.5.1.jar

hibernate-validator-6.0.18.Final.jar

jackson-core-2.10.2.jar

jackson-databind-2.10.2.jar

jakarta.annotation-api-1.3.5.jar

jakarta.validation-api-2.0.2.jar

jboss-logging-3.4.1.Final.jar

jul-to-slf4j-1.7.30.jar

log4j-api-2.12.1.jar

log4j-to-slf4j-2.12.1.jar

logback-core-1.2.3.jar

slf4j-api-1.7.30.jar

snakeyaml-1.25.jar

spring-boot-2.2.4.RELEASE.jar

spring-core-5.2.3.RELEASE.jar

spring-web-5.2.3.RELEASE.jar

tomcat-embed-core-9.0.30.jar

tomcat-embed-el-9.0.30.jar

tomcat-embed-websocket-9.0.30.jarjboss-logging-3.4.1.Final.jar

jul-to-slf4j-1.7.30.jar

log4j-api-2.12.1.jar

log4j-to-slf4j-2.12.1.jar

logback-core-1.2.3.jar

slf4j-api-1.7.30.jar

snakeyaml-1.25.jar

spring-boot-2.2.4.RELEASE.jar

spring-core-5.2.3.RELEASE.jar

spring-web-5.2.3.RELEASE.jar

tomcat-embed-core-9.0.30.jar

tomcat-embed-el-9.0.30.jar

tomcat-embed-websocket-9.0.30.jar

https://www.dropbox.com/preview/dependency-check-report.html?context=standalone\_preview&role=personal

## Mitigation Plan

Bouncy Castle JCE Provider version 1.55 and earlier the DSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure. It is important to make sure that we are using the latest version of this dependency.

Hibernate Validator 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. This can be handled with proper error handling

A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity. Check for the corresponding nested objects that create XML vulnerabilities.

Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1 These dependencies have since been patched.

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.

Using the most up-to-date dependency should do the trick.

The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

It seems like for the rest of these vulnerabilities, maintaining the most up to date dependencies is the best choice.