

# 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** | **09/16/2023** | **Christian Mendoza** | **revised and new recommendations.** |

## Client



## Instructions

## Developer

Christian Mendoza

## Interpreting Client Needs:

  For this assignment we would determine the customers' desires and possible menaces or attacks that are related to the software security demands. Artemis Financial is a company that formulates monetary objectives for clients that go from retirements, investments, savings, and also insurance. Their focus would be on security to make sure all their clients' portfolios and transactions are kept secure and safe. This is extremely essential because Artemis Financial handles a lot of classified important information, that includes tax information and social security numbers. This company not only deals in the United States, but it is also an international company that must make sure it protects all sensitive data. They must be aware of any phishing malware and invasion of privacy. A big restriction set by the government regarding all secure communication should be to make sure that there is no leaked information in regard to any trade secrets. This is due to the necessary need to protect all information data. The main objective would be to target any outside threats to customers' information. Much of this data has to be encrypted from someone not from the financial institution. An efficient way to limit malware or phishing would be by restricting access to all critical servers. Implementing two-way authentication as an additional form to authenticate and also alert all customers from an outside threat.

## Areas of Security:

-(a) input validation= When we input we need to keep a mindful consecutive injection because this can manipulate the expected output and eventually harm the application. This would also validate the customer's information and protect their data. This authentication would be recorded as a string.

-(b) code quality= With this you would be able to take charge of the input methods that would be assigned to the customer. A particular customer would be granted their specific information and no one else would have access to or even access to the server.

-(c) API's= Verification and authentication are extremely important which also includes the specific customers access validation and two step authentications. If we limit access to critical components to only a certain number of customers, this will make sure the platforms are secure.

-(d) code error: If we implement error handling this would allow us to understand the areas of API that would need to be updated.

-(e) cryptography= As we implement cryptography into this institution this would be essential as this would make sure all customers' information is not going to be endangered from other parts around the globe. Taking this into consideration there are different types of currencies.

## Manual Review:

As we go through the code, run, and expect it nothing appear to be out the extraordinary. It mainly had REST and no test principles. It had test occasions; it had no arrival criteria and no departure measure. The code quality was decent and there was a small issue with the error handling ultimately because it was not there. As we moved from the API, we were able to find it was missing a lot of things. There would eventually be a breach that could expose all customers' input mainly because it was not written through a POST method. I was mainly looking for signs of cryptography

## Static Testing:

-Dependency: log4j-api-2.12.1.jar

-Vulnerability: cpe: 2.3:apache:log4j:2.12.

-Description: unseemly corraboration of verification upon the host inconsistent in Apache log4j SMTP appended. As stated it would allow the SMTPS into connecting and intercepting by someone and can potentially leak any messages that would send through appender.

-solution: We need to improve tot the 2.12.2 that reinforce its requirement. Older versions would lay the systems properly mail, smtp.ssl.check server identity into the worldwide enable host name authenthication of SMTPS connections.

-Dependency: tomcat-embed-websocket-9.0.31.jar

-Vulnerability: cpe:2.3:a:apache.tomcat:9.0.31

cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.31:

-Description: The apache tomcat 10.0.7-mi to 10.0.7,8.0.0mi, to 9.0.47 and 8.6.0 to 8.6.66 these did not accurately determine the HTTP move cryptograph petition heading that leads to the possibility to petition process when we use with a opposite proxy. Tomcat was able to identify the cryptograph and tomcat act to not make sure as a result if shown the cast cryptograph was the the closing cryptograph.

-Solution: The best solution would be to upgrade to apache tomcat 10.0.7 otherwise current option.

-Dependency:bcprov-jdk14on-1.46.jar

-Vulnerability: cpe:2.3:a:bouncyastle:bouncy-castle-crypto-package:1.46.

-Description: The legion of the Bouncy castle Legion of the Bouncy castle java cryptography API's 1.67 up to and does not include 1.50 and contains a CWE-471: This employs apparently managing input through choosing categories or code susceptibilities.

-Solution: The best solution would be to update to bouncycastle to the last category update to 1.60.

-Dependency: spring-aop-5.1.3. RELEASE.jar

-Vulnerability: cpe:2.3:a:pivotal\_software:spring\_framework:5.1.3:release

cpe:2.3:a:springsource:spring\_framework:5.1.3:release

cpe:2.3:a:vmware:spring\_framework:5.1.3:release

cpe:2.3:a:vmware:springsource\_spring\_framework:5.1.3:release

-Description:In the spring frameworks versions 5.1.0-5.1.8,5.1.0-5.1.18, 5.0.7, 4.3.0-4.3.28, these are old versions and unsupported , the protection opposed RFD invasions from CVE-2015-5211 these can exist and go around and depend on the browser that is used along a jsessionid trail framework.

-Solution: The best solution would be to upgrade to the uptodate category.

-Dependency: hibernate-validator-6.0.19.final.jar

-Vulnerability: cpe:2.3:a:redhat:hibernate\_validator:6.0.19:

-Description: An error occurs using the hibernate authenticate session 6.1.2.final. There was a bug in the communication insert mainframe that makes unsupported el expressions that would have to be validated if they were authenticated. This defect would allow attackers to go around the input sanitation (stripping, escaping) controls that the developers would need to put in place when executing users controlled information in all unwanted messages.

-Solution: The best solution would be to upgrade to the hibernate-validator-6.0.20.

## Mitigation Plan:

## After analyzing all tests, the vast majority of the dependencies had liabilities which were due to outdated versions. Most of the security risk targets all these plugins. Most of these dependencies are open source. It is the main responsibility of the developer to make sure these expectations don't impact the code in a negative way. I was able to specify in detail all dependencies and weaknesses and all necessary actions to improve all these concerns. A vast majority of these vulnerabilities needed an update as such dependency show in different times of the code, as we review the manual process, we examine the spring Frame boot and conclude that it was an outdated version. We also examined the tomcat server and concluded that there was an update available, which was later confirmed by the static testing which exposed all these weaknesses on xml. file. An alternate ​result into​ identify security accountabilities for this Artemis Financial institution can be to just run a current version of the hibernator validator, bouncy castle, snakeyaml, and Apache Tomcat.