****

# 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** | **1/28** | **Josh Cantu** | **Version 1.0** |

## 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

Josh Cantu

## Interpreting Client Needs

Artemis is a financial service company and security must be a priority. Being a company that deals with financial services makes it a target for people with malicious intentions. The increase in online security will apply to the customers as well. They will feel comfortable keeping their assets and belongings with Artemis. Privacy should be protected and included for the customers. Having any sort of leak could taint the reputation of Artemis, this could result in a lack of trust in their capabilities to the public.

A breach could result in a huge loss for both the customers and the company. Again, this could result in a lack of trust in the public. Making it harder for Artemis to gain new customers or maintain their current customers. The highest level of encryption should be maintained by Artemis. New and improved encryption should be applied. As encryption technologies age, they become vulnerable to more and more powerful CPU/GPU that can take the time to crack those encryptions. Phishing attempts are still possible, if successful, it won’t matter how powerful the encryption technology is.

Keeping an online presence does keep you exposed to phishing attempts. In that case, training for Artemis staff and customers is required. Other types of attacks include Dos attacks. Having a plan to prevent DOS attacks should be in the playbook.

The use of open-source software should be reviewed carefully. Certain types of open- source software contain vulnerabilities if there is some sort of exposure in the code. Careful review and approval must be done prior to any code implementation.

## Areas of Security

After reviewing the Vulnerability Assessment Process Flow Diagram, I would focus on the following:

* Input Validation
* Secure API Interaction
* Code Error
* Encapsulation

Input validation would clean the input from the user. It would prevent anything malicious from being entered into the database. This includes a trusted user mis inputting something and causes a negative event. Securing the API would ensure that the connection between the user and the service is trustworthy. This would apply to any connection when working with Artemis’ services. Code checking would allow the code structure to be secure and does not allow the opening of malicious users to insert their code. Finally, Encapsulation would secure the data traffic and prevent any changes from being applied as it transitions from service to service.

## Manual Review

In POM.xml, Maven needed to be updated to the most recent version available. Utilizing older software will cause some CVEs to be missed.

User inputs in the API should be cleaned. “String message = "Element in the given index is :: "+myArray[id].” Found in the number method, allows input from the user. This is a problem because the variable is raw and allows an injection to take place in that variable.

myDateTime class should be revised. The method has been created improperly and is incomplete. DocData class contains a username and password, which is readable to anyone viewing the code. Before the code is made public, this type of data should be cleaned and removed. Utilizing the username and password during testing is acceptable but it should be removed after testing is complete. Even the combo is in a bad state. Root/root as a username/password is bad practice since it is very easy to guess.

## Static Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Dependency | Vulnerability ID | Description | Resolution |
| bcprov-jdk15on-1.46.jar | cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46  cpe:2.3:a:bouncycastle:bouncy\_castle\_for\_java:1.46  cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46  cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46 | Bouncy Castle API is a lightweight cryptography for Java and C#. | Update from this version. |
| hibernate-validator-6.0.18.Final.jar | cpe:2.3:a:redhat:hibernate\_validator:6.0.18 | Allows to express and validate application constraints | Upgrade to the latest version |
| snakeyaml-1.25.jar | cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25 | Popular Java Library | Upgrade to the latest version |
| spring-boot-2.2.4.RELEASE.jar | cpe:2.3:a:vmware:spring\_boot:2.2.4:release | Creates easy stand-alone spring-based app that “just run” | Upgrade to latest version |
| spring-core-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release  cpe:2.3:a:springsource:spring\_framework:5.2.3:release  cpe:2.3:a:vmware:spring\_framework:5.2.3:release | Core Framework for Spring | Upgrade to latest version |
| spring-web-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release  cpe:2.3:a:springsource:spring\_framework:5.2.3:release  cpe:2.3:a:vmware:spring\_framework:5.2.3:release  cpe:2.3:a:web\_project:web:5.2.3:release | Focused on creating document-driven web services. | Upgrade to the latest version |
| tomcat-embed-core-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30 | Core Tomcat service | Upgrade to the latest version |
| tomcat-embed-websocket-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30 | Core Tomcat service | Upgrade to latest the version |
| jackson-databind-2.10.2.jar | cpe:2.3:a:fasterxml:jackson-databind:2.10.2  cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2 | Allows the reading of JSON content to Java Objects | Upgrade to the latest version |
| logback-core-1.2.3.jar | cpe:2.3:a:qos:logback:1.2.3 | Contains logging API and core service | Upgrade to the latest version |

## Mitigation Plan

The results of the review and static testing of the code are the following:

An update of all services and libraries should be updated. This will patch any of the vulnerabilities in the code. After recompiling, the code will need to be tested once again to ensure everything is running correctly. The recommendations above should be applied once the libraries and services have been updated. Test the code again to ensure it is running correctly. Once internal testing is complete, it should be either peer-reviewed or approved by management to push to live and active version for public use.