****

# Artemis Financial Vulnerability Assessment Report

Adam Benoit

Southern New Hampshire University

CS-305-T2811 Software Security 22EW2

11/13/2022

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** | **11/13/2022** | **Adam Benoit** |  |

## 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

Adam Benoit

## Interpreting Client Needs

Artemis Financial is a company that provides finical help and advice in areas of savings, retirements investing options and insurance. Their clientele ranges from individual entrepreneurs, businesses and even to government agencies around the world. With this wide range of clients one can expect they have many international transactions they have to monitor that can leave them open to cyber attacks abroad. Currently in the briefing provided there was no specific restrictions listed but one can expect with government agencies their will be some and one in particular would be encryptions of data and communication to make state and foreign transactions secure. Encryption should also extend to other customers as best practice working with companies there may be those looking to get inside information by accessing company finical records. To better modernize Artemis would be to make sure they have up to date maintenance checks enforced for bug fixes. Also, to upgrade their web application to https if they are still using http.

## Areas of Security

After reviewing Artemis Finical, I have identified the following areas of possible vulnerabilities for security:

* APIs: Since this application is running externally and internally as well as on end users web browsers the API will need to be well developed. The API will be what the user will use and interact with when using the program.
* Code Error: Correct implementation of error handling will allow for better understanding of what areas of API would need to be fixed. This will also help prevent any unwanted access to unauthorized access to data.
* Code quality: The quality of the code is of high importance in an application that involves users and validating their inputs. This will help narrow down the possibility of data exposure to threat actors that look to exploit lack of code quality.
* Cryptography: This is of high importance due to valuable information being stored by the company’s customers. Data and any communications regarding accounts whether it be our government or foreign and personal business accounts should follow these guidelines. All data should be stored in compliance of domestic and foreign policy to limit exposure to threat actors and negligence lawsuits against Artemis.
* Input validation: This is crucial to verify who has access to information when users sign in. This would be written as strings.

## Manual Review

After manually inspecting the code and applying the vulnerability assessment and reviewing the pom.xml and the classes in the program file. I didn’t notice any input validation in the greeting controller which would need to be implemented. I also noticed in the pom.xml file there seems to be many things outdated from just a manual inspection. For example, the java version on line 18 is version 1.8 and has been outdated for a few years now and could be a risk for security breaches. The dependency check was also outdated and was using version 5.3.0 o n line 60. I upgraded the code for this to reflect the new version of 7.3.0 to have a more accurate dependency check for the next section. Spring framework is also outdated on line 8 and should be updated to the new version. Overall, the code quality seemed good but the code seems to lack error handling. The API was lacking and will need to be improved or rewritten using POST method. Lastly, I wanted to find the cryptography but was unable to locate any signs of it and this would need to be added to the list of things to do and would need to be one of the top priorities to implement as soon as possible.

## Static Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Dependency | Vulnerability | Description | Solution |
| 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:legion-of  -the-bouncy-castle-java-crytography  -api:1.46:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:bouncycastle:the\_bouncy  \_castle\_crypto\_package\_for\_java:1.  46:\*:\*:\*:\*:\*:\*:\* | This version does not fully validate ASN. This can be exploited to allow elements to be injected into a sequence and still have a signature be validated. This could allow DSA key generators for cryptography to be manipulated. | Would be to update to the newest version of 1.72 according to the software’s website or at least to any version past version 1.55 |
| hibernate-validator-6.0.18.Final.jar | cpe:2.3:a:redhat:hibernate\_  validator:6.0.18:\*:\*:\*:\*:\*:\*:\* | This version is open to the possibility of hacker exploiting it to allow them to bypass input sanitation that was put in place to handle user controlled data in error messages. Allowing normally rejected EL expressions to show as valid. | Would be to upgrade to the newest version of 8.0.0final according to the software website or minimal to upgrade to version 6.0.20 where this was patched. |
| 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:\*:\*:\*:  \*:\*:\*:\* | This threaten data integrity and does not have the entity expansion secured properly. This is open to external threat attackers looking to exploit our data integrity. A DOS attack can be used to exploit this. | Updating this to the newest version available is of high importance. |
| log4j-api-2.12.1.jar | cpe:2.3:a:apache:log4j:  2.12.1:\*:\*:\*:\*:\*:\*:\* | A man in the middle attack could exploit invalid certificate validation to allow SMTPS connection to be intercepted and log messages to be viewed by the same threat actor. | Upgrading this to the newest version |
| logback-core-1.2.3.jar | cpe:2.3:a:qos:logback:  1.2.3:\*:\*:\*:\*:\*:\*:\* | Could allow threat attackers to edit configurations files and craft malicious configurations allowing of code to be executed on the servers threatening all data. | Upgrade to newest version |
| snakeyaml-1.25.jar | cpe:2.3:a:snakeyaml\_  project:snakeyaml:1.25:  \*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:yaml\_project:  yaml:1.25:\*:\*:\*:\*:\*:\*:\* | Vulnerable to DOS attacks and also allows entity expansion during load operations. | Either updating to the newest version so this is patched or migrating to snakeyaml engine. The engine has the option to restrict collections. |
| spring-boot-2.2.4.RELEASE.jar | cpe:2.3:a:vmware:s  pring\_boot:2.2.4:  release:\*:\*:\*:\*:\*:\* | This may allow attacks through remote code execution via data binding attempts. Needs to be run as a WAR deployment but may be exploited in other ways. | Upgrade to the newest version available |
| 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:\*:\*:\*:\*:\*:\* | Can cause a threat actor to exploit a vulnerability that allows privilege escalation. After said attacker can read and modify files that have been uploaded. | Upgrade to the newest version available |
| 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:  \*:\*:\*:\*:\*:\* | Same as spring-core-5.2.3.RELEASE.jar | Same as spring-core-5.2.3.RELEASE.jar |
| tomcat-embed-core-9.0.30.jar and 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:\*:\*:\*:\*:\*:\*:\* | Allows reverse proxy to request smuggling by allowing http to not correctly parse the HTTP. Does not ensure that the encoding is the final encoding at the end of transfer. | Upgrade to the newest version or to atleast to version 10.0.6 where this was patched |

## Mitigation Plan

For a lot of the security threats found in this report the easy fix would be to upgrade to the most recent version of what has been implemented inside the code. For example, upgrading to the current versions of javadk, snakeyaml, hibernate, spring, ad tomcat would eliminate a lot of threats to the security of this program. Another thing that needs to be immediately implemented would be creating an encryption process to be coded in for the communication and data the application needs to process to better secure user data. The API needs to be reworked or start from the beginning and readded for increase security along with better error handling to limit client data to exposure. Also, if this application is to be used over URL then https is a must have and http cannot be used.

**References**

Manico, J., & Detlefsen, A. (2015). *Iron-clad java: Building secure web applications ; best practices for secure Java Web Application Development*. Oracle Press.