****

# Artemis Financial Vulnerability Assessment Report

**Table of Contents**

[Document Revision History 3](#_gjdgxs)

[Client 3](#_30j0zll)

[Instructions 3](#_1fob9te)

[Developer 4](#_3znysh7)

[1. Interpreting Client Needs 4](#_2et92p0)

[2. Areas of Security 4](#_tyjcwt)

[3. Manual Review 4](#_3dy6vkm)

[4. Static Testing 4](#_1t3h5sf)

[5. Mitigation Plan 4](#_4d34og8)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/14/2023** | **Clayton Smith** |  |

## 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

Clayton Smith

## Interpreting Client Needs

Artemis Financial is a consulting company that develops financial plans for customers. These financial plans can range from savings, retirement, investments, and insurance. Artemis Financial wants us to modernize their operations by using the most current and up to date software security.

With this information, we can assume that client information is going to need to be secure and protected, especially when financials are at play. This information could range anywhere from banking numbers, SSN, tax information, etc.

The information provided does not disclose if Artemis is an international company or domestic, so we should operate as if they are or potentially will have international ties. This will require the company to run maintenance checks regularly to ensure the protection of their client’s data.

## Areas of Security

Below I covered the areas of security by using the Vulnerability Assessment Process Flow Diagram:

* **Input Validation**
  + Requiring Input Validation will help Artemis protect their end-user’s data. Input Validation would provide that extra step of protection that should be required for the end-user to access any info about their account with Artemis.
* **APIs**
  + Since data will be transferred between end-users and internal employees, an API is required in this situation.
* **Cryptography**
  + Cryptography would either be a requirement or strongly encouraged as sensitive data will be transferred between Artemis and the end-user. Protecting this information and data should be crucial.
* **Code Error**
  + Handling errors in the code would be important with the amount of sensitive data Artemis would be handling. A data leak could be devastating for the end-user and Artemis as a company. To prevent this, secure error handling should be implemented.
* **Code Quality**
  + When dealing with sensitive data, code quality should also be the best it could be. Having neat and clean code is important not only in development, but also if a fix is needed and someone who didn’t participate in the development of the code has to understand what they are looking for.

## Manual Review

Upon opening the code, I initially checked the pom file to see if anything needed to be updated or was throwing any errors. Initially, everything looked good. The next file I checked was the “DocData.java” file. It was throwing an error due to the “con” variable not being used. Ultimately, I would look into this as it could be a sign of unfinished code. The final file I decided to check was the “Greeting.java” file. I wanted to see if the input validation was up to par with my initial recommendation. It's safe to say it lacked what I wanted to see. There needs to be a large overhaul on some type of input validation due to the current system being so vulnerable.

## Static Testing

After running the code through a dependency check, I’ve outlined the potential vulnerable dependencies below in a table:

| **Dependency** | **Vulnerability IDs** | **Description** | **Solution** |
| --- | --- | --- | --- |
| bcprov-jdk15on-1.46.jar | CVE-2016-1000352 | In the Bouncy Castle JCE Provider version 1.55 and earlier the ECIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider. | Upgrade Bouncy Castle past version 1.55 |
| spring-boot-2.2.4.RELEASE.jar | CVE-2022-27772 | \*\* UNSUPPORTED WHEN ASSIGNED \*\* spring-boot versions prior to version v2.2.11.RELEASE were vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer. | Upgrade spring-boot to v2.2.11 |
| logback-core-1.2.3.jar | CVE-2021-42550 | In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configuration files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers. | Upgrade logback to v1.3.5 |
| log4j-api-2.12.1.jar | CVE-2021-44832 | Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP data source URI when an attacker has control of the target LDAP server. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2. | Confirmed fixed in version 2.17.1 |
| snakeyaml-1.25.jar | CVE-2022-41854 | Those using Snakeyaml to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stack overflow. This effect may support a denial of service attack. | Update to v1.32 RELEASE for fix |
| jackson-databind-2.10.2.jar | CVE-2022-42004 | In FasterXML jackson-databind before 2.13.4, resource exhaustion can occur because of a lack of a check in BeanDeserializer.\_deserializeFromArray to prevent use of deeply nested arrays. An application is vulnerable only with certain customized choices for deserialization. | Fixed in version 2.12.1 |
| tomcat-embed-core-0.0.30.jar | CVE-2022-42252 | If Apache Tomcat 8.5.0 to 8.5.82, 9.0.0-M1 to 9.0.67, 10.0.0-M1 to 10.0.26 or 10.1.0-M1 to 10.1.0 was configured to ignore invalid HTTP headers via setting rejectIllegalHeader to false (the default for 8.5.x only), Tomcat did not reject a request containing an invalid Content-Length header making a request smuggling attack possible if Tomcat was located behind a reverse proxy that also failed to reject the request with the invalid header. | Upgrade to Apache Tomcat 10.1.1 or later |
| hibernate-validator-6.0.18.Final.jar | CVE-2020-10693 | 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. | Update to v6.0.20 Final |

## Mitigation Plan

Once everyone has had a chance to review the static testing and manual review, I suggest that we begin by upgrading all vulnerable libraries to the recommended versions above. If a newer version is available, we can run a check on those before making the move to that version.