****

# 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** | **7-16-2023** | **Aaron Adeyemi** |  |

## 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

Aaron Adeyemi

## Interpreting Client Needs

Artemis Financial deals with savings, investments, insurance, and retirement plans. These kinds of services deal with highly sensitive customer information (names, addresses, credit card information, employment, and so on). Artemis Financial needs their security to be as tight as possible to instill a consistent sense of security in their customers at every step of the way.

It isn’t specified in the briefing what parts of the world Artemis Financial provides their services, but it might be necessary to clarify this information before proceeding.

All of Artemis Financial’s software and communications should be PCI DSS and SOX compliant.

The largest persistent threat to Artemis Financial would be malicious hackers. They will be constantly looking for a means of accessing and likely stealing the information being held by Artemis Financial about their customers. They might look to hold this information ransom or simply to sell the information.

As technology improves, so do the means of illegally accessing those technologies. New software and APIs aren’t always fully secure on release, ensuring that all new implementations are properly tested and vetted before employing them across the company will be crucial in securing Artemis Financial’s modernization efforts.

## Areas of Security

These areas of security are most relevant to Artemis Financial’s software: Input Validation, APIs, Cryptography, Code Error, and Code Quality.

Input Validation: The software needs to be fully secure against attacks such as SQL injections and other attacks that rely on illegal inputs that predate on improperly handled exceptions.

APIs: Most software relies on external libraries and APIs to help build their own applications. Extensive testing and vetting of any externally applied APIs will be necessary to ensure that all additions are secure enough to employ.

Cryptography: Artemis Financial’s service demands that they handle very sensitive customer information. This necessity then calls for properly implemented cryptography to secure customer information when it is being input, transported, or stored.

Code Error: This alongside Input Validation, are safe guards to ensure that the code isn’t at any point being manipulated to handle errors incorrectly, thus producing unexpected, and unintended results that might be detrimental to the customers of Artemis Financial.

Code Quality: Implementing industry standards and taking every necessary precaution to make the most secure code possible is the bare minimum for software of this nature.

## Manual Review

* In the “pom.xml” file, the version for the Dependency Maven Check is out of date. Ensuring that all tools are up to date is key for securing code.
* The HTTPs protocol is not employed in any communication for security.
* There is no validation/exception of any request made.
* The business name should not be used as a request parameter in CRUDController.

## Static Testing

|  |  |
| --- | --- |
| **Dependency** | **Description** |
| bcprov-jdk15on-1.46.jar | The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7. |
| log4j-api-2.12.1.jar | The Apache Log4j API |
| logback-core-1.2.3.jar | logback-core module |
| snakeyaml-1.25.jar | YAML 1.1 parser and emitter for Java |
| spring-boot-2.2.4.RELEASE.jar | Spring Boot |
| spring-boot-starter-web-2.2.4.RELEASE.jar | Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container |
| spring-core-5.2.3.RELEASE.jar | Spring Core |
| spring-web-5.2.3.RELEASE.jar | Spring Web |
| spring-webmvc-5.2.3.RELEASE.jar | Spring Web MVC |
| tomcat-embed-core-9.0.30.jar | Core Tomcat implementation |
| jackson-databind-2.10.2.jar | General data-binding functionality for Jackson: works on core streaming API |
| hibernate-validator-6.0.18.Final.jar | Hibernate's Bean Validation (JSR-380) reference implementation. |

|  |  |
| --- | --- |
| **Dependency** | **Summaries** |
| bcprov-jdk15on-1.46.jar | 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. |
| log4j-api-2.12.1.jar | 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. |
| logback-core-1.2.3.jar | 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. |
| snakeyaml-1.25.jar | SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution. We recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization. We recommend upgrading to version 2.0 and beyond. |
| spring-boot-2.2.4.RELEASE.jar | There is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache. |
| spring-boot-starter-web-2.2.4.RELEASE.jar | There is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache. |
| spring-core-5.2.3.RELEASE.jar | It is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition. |
| spring-web-5.2.3.RELEASE.jar | It is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition. |
| spring-webmvc-5.2.3.RELEASE.jar | It is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition. |
| tomcat-embed-core-9.0.30.jar | When using the RemoteIpFilter with requests received from a reverse proxy via HTTP that include the X-Forwarded-Proto header set to https, session cookies created by Apache Tomcat 11.0.0-M1 to 11.0.0.-M2, 10.1.0-M1 to 10.1.5, 9.0.0-M1 to 9.0.71 and 8.5.0 to 8.5.85 did not include the secure attribute. This could result in the user agent transmitting the session cookie over an insecure channel. |
| jackson-databind-2.10.2.jar | jackson-databind 2.10.x through 2.12.x before 2.12.6 and 2.13.x before 2.13.1 allows attackers to cause a denial of service (2 GB transient heap usage per read) in uncommon situations involving JsonNode JDK serialization. |
| hibernate-validator-6.0.18.Final.jar | 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. |

## Mitigation Plan

* Update dependencies and test tools.
* Implement HTTPs protocol for all communications.
* Implement secure validation/authentication of all requests.