****

# 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** | **9/17/2023** | **Nick Vito** |  |

## 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

Nick Vito

## Interpreting Client Needs

Value of Secure Communication: Secure communications are very important to a business such as Artemis Financial. Because they handle sensitive financial data and personal information, ensuring the confidentiality and integrity of data transmissions is crucial. This includes protecting client data during interactions with their web application, as well as secure communication with external partners and other financial institutions. Making sure communications are safe not only helps maintain the company’s reputation but also allows the company to continue to gain the trust of their clients.

International Transactions: If Artemis Financial plans to conduct international transactions, additional security measures must be addressed. Transactions that span our globe will have to comply with international data and protection regulations such as GDPR or CCPA, which both add stringent security measures. International transactions also add another layer of cyber threats, as it opens the company up to cybercrime networks that operate strictly outside the US.

Governmental Restrictions: Depending on where Artemis Financial plans to operate or serve their clients, there will be governmental restrictions on secure communications to factor in. The company must follow these regulations to a T, as most of these are nonnegotiable. If there is a failure to comply with these regulations it could have legal ramifications and cause severe damage to the company’s reputation.

External Threats: Identifying external threats is a critical aspect of vulnerability assessment. Cyber criminals have notoriously targeted the financial sector specifically due to the potential financial gain. Threats that’s come from these external threats include DDoS attacks, phishing attempts, SQL Injection, and ransomware attacks. Having a deep comprehension of the evolving threat landscape and having the ability to anticipate emerging threats is essential to proactive security.

Modernization Requirements: Meeting the requirements Artemis Financial has for modernization, several things must be considered. First, the role of open-source libraries is important. They can easily introduce vulnerabilities if they are not properly managed or updated. Staying updated on the latest developments in web application technologies is essential to guarantee both compatibility and security. Embracing modern software development practices, such as DevSecOps, can help integrate security seamlessly throughout the development lifecycle, mitigating vulnerabilities early in the process.

## Areas of Security

The areas of security that I feel are relevant to Artemis Financial are as follows:

Secure API Interactions: Artemis Financials web application will likely communicate with external systems and other financial partners to exchange financial data. Securing these API interactions is vital to prevent data breaches and unauthorized access to the sensitive information being disseminated. It is paramount to secure data exchanged with external systems.

**Secure Coding Practices and Patterns:** Secure coding practices are essential for Artemis Financials web application. These practices will help prevent common vulnerabilities, for example SQL injection and cross site scripting. They will also help the code base be resilient to security threats.

**Cryptography and Encryption Use:** Due to the confidential nature of the data and personal information Artemis Financial handles Cryptography and Encryption are very important. By making sure data is encrypted it can be protected from unauthorized access and possible data breaches.

Code Review: Code Reviews are essential in pinpointing vulnerabilities in the application. This includes models, controllers, data access functions, services, plug-ins, and APIs. With good code reviews, Artemis Financial can ensure that secure coding practices are followed. This reduces the risk of exploitation.

## Manual Review

The only vulnerability that could possibly become an issue is that the username and password are set to root. This configuration raises security concerns since 'root' is not a recommended username and password combination. I believe that if they are easily guessable that this could leave the system open to unauthorized access attempts or brute force attacks.

## Static Testing

A screenshot of a computer

Description automatically generated

My dependency check found 13 and 75 vulnerabilities. Out of the 13 dependencies 4 were critical and 4 were high, the rest were medium. I will list the dependencies that need tending too the most.

bcprov-jdk15on-1.46.jar

Description

The Bouncy Castle Crypto package represents a Java-based implementation of cryptographic algorithms. Enclosed within this JAR file are both a Java Cryptography Extension (JCE) provider and a streamlined API that seamlessly aligns with the Bouncy Castle Cryptography APIs. This versatile package offers compatibility with Java Development Kit (JDK) versions spanning from 1.5 to 1.7.

Vulnerabilities:

CVE-2016-1000342: In Bouncy Castle JCE Provider versions 1.55 and earlier, there is a susceptibility in the DSA signature generation process, making it vulnerable to timing attacks. In situations where the generation of signatures allows for precise timing observation, the absence of blinding in these versions (1.55 and earlier) creates an opportunity for attackers to gather information regarding the signature's "k" value, and potentially, the private value, thereby compromising the security of the signature generation process.

Solution: Apply Update to Bouncy Castle.

CVE-2016-1000338: In Bouncy Castle JCE Provider versions 1.55 and earlier, there exists a vulnerability related to DSA signature verification. Specifically, these versions do not perform comprehensive validation of the ASN.1 encoding in signature verification. Consequently, it becomes feasible to inject supplementary elements into the signature's sequence and still attain a successful validation outcome. In certain scenarios, this security weakness might enable the introduction of 'hidden' data into a signed structure.

Solution: Apply Update to Bouncy Castle.

CVE-2016-1000352 & CVE-2016-1000344: Prior to version 1.55, the Bouncy Castle JCE Provider included support for ECB mode in its ECIES implementation. However, ECB mode is recognized as unsafe, and as a result, it has been deprecated and removed from the provider in later versions.

To address these vulnerabilities, I would suggest updating to 1.70 from what I have looked up. Also suggest upgrading bouncycastle packages.

spring-boot-2.2.4.RELEASE.jar

Description:

Spring Boot

Vulnerabilities:

CVE-2022-27772: This vulnerability, designated as CVE-2022-27772, affects spring-boot versions that were vulnerable to temporary directory hijacking. Specifically, versions prior to v2.2.11.RELEASE were susceptible to this issue. The vulnerability is associated with the method org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir. It is important to note that this vulnerability only impacts products and/or versions that are no longer supported by the maintainer. If you are using a supported version of Spring Boot, you are not affected by this vulnerability.

Solution: Update to v2.2.11.RELEASE

CVE-2023-20873: In Spring Boot versions 3.0.0 to 3.0.5, as well as 2.7.0 to 2.7.10, and any older unsupported versions, there exists a potential security bypass vulnerability when deploying applications on Cloud Foundry.

Solution: For 3.0.x users, it is recommended to upgrade to version 3.0.6 or higher. If you are using version 2.7.x, it is advisable to upgrade to version 2.7.11 or higher. Users of older, unsupported versions should consider upgrading to either 3.0.6 or 2.7.11 or higher to address the vulnerability.

CVE-2023-20883: Within the Spring Boot ecosystem, encompassing versions 3.0.0 through 3.0.6, 2.7.0 through 2.7.11, 2.6.0 through 2.6.14, 2.5.0 through 2.5.14, as well as any older unsupported iterations, a vulnerability has been identified. This vulnerability poses a potential risk for a denial-of-service (DoS) attack when Spring MVC is employed in conjunction with a reverse proxy cache.

Solution: If you are using version 3.0.x, It is highly advisable to perform an upgrade to the specified version 3.0.7 or a higher release to address the security concerns. For users on version 2.7.x, it is advisable to upgrade to version 2.7.12 or a newer version to ensure the vulnerabilities are resolved. If your application relies on version 2.6.x, it is essential to upgrade to version 2.6.15 or a later release to mitigate potential risks. Similarly, for those using version 2.5.x, upgrading to version 2.5.15 or higher is crucial to address any security issues. Users who are on older, unsupported versions should consider upgrading to either version 3.0.7 or 2.7.12 or a more recent release to apply the necessary security mitigations and protect their systems from potential vulnerabilities.

log4j-api-2.12.1.jar

Description:

The Apache Log4j API

Vulnerabilities:

CVE-2021-44228: Apache Log4j2 versions 2.0-beta9 through 2.15.0, excluding security releases 2.12.2, 2.12.3, and 2.3.1, have been identified as vulnerable due to inadequate protection against potential threats from attacker-controlled LDAP and other JNDI-related endpoints within configuration, log messages, and parameters. This security issue may allow attackers to execute arbitrary code from LDAP servers when message lookup substitution is enabled. Notably, this vulnerability pertains exclusively to log4j-core and does not affect other Apache Logging Services projects like log4net or log4cxx.

Solution: Starting from log4j version 2.15.0, this behavior has been disabled by default, and from version 2.16.0 (including 2.12.2, 2.12.3, and 2.3.1), this functionality has been entirely removed. So, suggest updating to 2.16.0

CVE-2021-45046: A critical issue was discovered where the fix for CVE-2021-44228 in Apache Log4j 2.15.0 proved to be incomplete in certain non-default configurations. In these scenarios, attackers with control over Thread Context Map (MDC) input data could exploit the vulnerability when the logging configuration utilizes a non-default Pattern Layout combined with either a Context Lookup (e.g., $${ctx:loginId}) or a Thread Context Map pattern (%X, %mdc, or %MDC). This manipulation of input data allowed malicious actors to craft specific JNDI Lookup patterns, leading to potential information disclosure and remote code execution in specific environments, and local code execution across all environments.

Solution: To address this issue, Log4j versions 2.16.0 (Java 8) and 2.12.2 (Java 7) have taken corrective actions. These actions involve the removal of support for message lookup patterns and the default disabling of JNDI functionality, effectively resolving the problem.

snakeyaml-1.25.jar

Description:

YAML 1.1 parser and emitter for Java

Vulnerabilities:

CVE-2017-18640: The Alias feature in SnakeYAML versions prior to 1.26 exhibits a vulnerability that permits entity expansion during a load operation, resembling a concern related to CVE-2003-1564.

Solution: Suggest using OWASP also suggest patching.

CVE-2022-1471: In SnakeYAML, the Constructor() class does not enforce restrictions on the types that can be instantiated during deserialization. Consequently, deserializing YAML content supplied by a potential attacker can potentially result in remote code execution.

Solution: To mitigate this risk, it is strongly advised to utilize SnakeYAML's SafeConstructor when parsing untrusted content, as it imposes constraints on deserialization. Additionally, upgrading to SnakeYAML version 2.0 or later is recommended to benefit from enhanced security measures.

CVE-2022-25857: The org.yaml:snakeyaml package, specifically versions prior to 1.31, is susceptible to a Denial of Service (DoS) vulnerability. This vulnerability arises from the absence of a nested depth limitation for collections.

Solution: Upgrade SnakeYAML packages.

CVE-2022-3064: Parsing YAML documents that are either maliciously crafted or exceptionally large may result in the consumption of excessive CPU or memory resources.

Solution: Recommend upgrading golang-yaml.v2 Packages.

jackson-databind-2.10.2.jar

Description:

General data-binding functionality for Jackson works on core streaming API.

Vulnerabilities:

CVE-2020-25649: A vulnerability was identified in FasterXML Jackson Databind where entity expansion was not adequately secured, leaving it susceptible to XML external entity (XXE) attacks. The primary concern associated with this vulnerability is the potential compromise of data integrity.

Solution: Update to version 2.10.5.1

CVE-2020-36518: Before version 2.13.0, jackson-databind is susceptible to a Java StackOverflow exception and a potential denial-of-service situation when confronted with an extensive nesting of objects.

Solution: Upgrade Jackson-databind packages.

CVE-2021-46877: In Jackson-databind versions 2.10.x through 2.12.x before 2.12.6, as well as 2.13.x before 2.13.1, a vulnerability exists that could be exploited by attackers to induce a denial-of-service condition. This occurs in uncommon scenarios related to JsonNode JDK serialization and can lead to a substantial transient heap usage of up to 2 GB per read operation.

Solution: Update to 2.13.1

CVE-2022-42004: In FasterXML jackson-databind prior to 2.13.4, a resource exhaustion risk emerges due to a missing check in BeanDeserializer's \_deserializeFromArray method. This omission allows the use of deeply nested arrays, potentially leading to resource depletion. It's important to note that this vulnerability affects applications with specific customizations for deserialization.

Solution: As with CVE-2020-36518 Upgrading Jackson-databind packages will fix this.

tomcat-embed-core-9.0.30.jar

Description:

Core Tomcat implementation

CVE-2020-11996: A carefully crafted sequence of HTTP/2 requests directed at Apache Tomcat versions 10.0.0-M1 to 10.0.0-M5, 9.0.0.M1 to 9.0.35, and 8.5.0 to 8.5.55 may result in a significant spike in CPU utilization for several seconds. In cases where a substantial volume of such requests is simultaneously made on concurrent HTTP/2 connections, this could lead to server unresponsiveness.

Solution: Upgrade Tomcat9 packages

CVE-2020-13934: When establishing an h2c direct connection to Apache Tomcat versions 10.0.0-M1 to 10.0.0-M6, 9.0.0.M5 to 9.0.36, and 8.5.1 to 8.5.56, a situation arises where the HTTP/1.1 processor is not properly released after transitioning to HTTP/2. In cases where a significant volume of these requests is generated, it could potentially result in an OutOfMemoryException, consequently causing a denial-of-service situation.

Solution: As with CVE-2020-11996 Upgrading tomcat9 packages will fix this.

CVE-2020-13935: In Apache Tomcat versions 10.0.0-M1 to 10.0.0-M6, 9.0.0.M1 to 9.0.36, 8.5.0 to 8.5.56, and 7.0.27 to 7.0.104, there was a lack of proper validation for payload length within WebSocket frames. This oversight meant that invalid payload lengths could potentially trigger an infinite loop. In the event of multiple requests featuring these invalid payload lengths, a denial-of-service situation could be initiated.

Solution: Again, upgrading tomcat9 packages for fix.

CVE-2020-1938: When working with the Apache JServ Protocol (AJP), Caution must be exercised when depending on incoming connections to Apache Tomcat. In various versions of Apache Tomcat, ranging from 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50, and 7.0.0 to 7.0.99, there was a default configuration where an AJP Connector was enabled, listening on all configured IP addresses. The security guide recommended disabling this Connector if not explicitly required. However, a vulnerability was discovered that allowed for the retrieval of arbitrary files within the web application and the processing of any file as a JavaServer Page (JSP). In cases where web applications permitted file uploads and stored these files within the application, along with the ability to process files as JSP, this vulnerability presented the risk of remote code execution.

Mitigation is necessary only if an AJP port is accessible to untrusted users. Those seeking a defense-in-depth approach can upgrade to Apache Tomcat versions 9.0.31, 8.5.51, or 7.0.100 or later, which introduce changes to the default AJP Connector configuration to enhance security. Users upgrading to these versions may need to make minor configuration adjustments to align with these enhancements.

Solution: Upgrade Tomcat9 packages

CVE-2020-8022: In various SUSE products, including SUSE Enterprise Storage 5, SUSE Linux Enterprise Server versions 12-SP2-BCL, 12-SP2-LTSS, 12-SP3-BCL, 12-SP3-LTSS, 12-SP4, 12-SP5, 15-LTSS, SUSE Linux Enterprise Server for SAP versions 12-SP2, 12-SP3, and 15, as well as SUSE OpenStack Cloud versions 7, 8, and SUSE OpenStack Cloud Crowbar 8, a security vulnerability related to incorrect default permissions has been identified. This vulnerability may allow local attackers to elevate their privileges from the "tomcat" group to "root." The affected versions of the Apache Tomcat software in these products include those prior to 8.0.53-29.32.1 for SUSE Enterprise Storage 5, and versions prior to 9.0.35-3.57.3 for the other mentioned products. It is essential to address this vulnerability promptly to prevent unauthorized privilege escalation.

Solution: Update Tomcat9 to latest version

spring-web-5.2.3.RELEASE.jar

Description:

Spring Web

CVE-2016-1000027 (OSSINDEX): Pivotal Spring Framework, up to version 5.3.16, has a potential remote code execution (RCE) vulnerability when handling Java deserialization of untrusted data. Whether this vulnerability is exploitable depends on how the library is implemented in a specific product, and sometimes, authentication might be necessary for an attack. It's important to note that the vendor's official stance is that using untrusted data for deserialization is not an intended use case. Consequently, the product's behavior will remain unchanged, as certain users rely on deserialization for trusted data.

Solution: No permanent solution but found a page that suggested to ensure there are no HTTP Invoker endpoints exposed to untrusted clients.

CVE-2021-22118 (OSSINDEX): Within Spring Framework, specifically versions 5.2.x prior to 5.2.15 and versions 5.3.x before 5.3.7, a security issue arises in WebFlux applications. This vulnerability enables a locally authenticated malicious user to elevate their privileges. The method involves manipulating the temporary storage directory, granting unauthorized access to read or alter files uploaded to the WebFlux application. Additionally, it facilitates the overwriting of arbitrary files by utilizing multipart request data.

Solution: To resolve this issue, users on version 5.3.x should upgrade to version 5.3.7, while those on version 5.2.x should upgrade to version 5.2.15. No additional actions are required.

spring-beans-5.2.3.RELEASE.jar

Description:

Spring Beans

CVE-2022-22965 (OSSINDEX): A Spring MVC or Spring WebFlux application operating on JDK 9 or later may potentially be susceptible to remote code execution (RCE) through data binding. The successful exploitation of this vulnerability necessitates the application's deployment on Tomcat as a WAR deployment. If, however, the application is deployed as a Spring Boot executable JAR, following the default configuration, it is not susceptible to this exploit. It's important to note that while this exploitation is described in a specific context, the underlying vulnerability has broader implications, and there may be alternative methods of exploitation.

Solution: Recommend upgrading to Spring Framework v5.3.18 to fix this vulnerability.

## Mitigation Plan

Looking through my dependency check, it seems as though most vulnerabilities are due to not keeping code, and application architecture up to date with the latest versions. To mitigate these vulnerabilities, it is very important to always check for the latest version of everything. Here is a list of mitigations that should be carried out to make sure we keep the application secure.

Bouncy Castle (bcprov-jdk15on-1.46.jar)

Vulnerabilities (CVE-2016-1000342, CVE-2016-1000338, CVE-2016-1000352, CVE-2016-1000344): Update to version 1.70 and consider upgrading Bouncy Castle packages.

Spring Boot (spring-boot-2.2.4.RELEASE.jar)

Vulnerabilities (CVE-2022-27772, CVE-2023-20873, CVE-2023-20883): Upgrade to version 2.2.11.RELEASE for CVE-2022-27772, and for others, upgrade to the recommended versions (3.0.6 or 2.7.11 or higher).

Apache Log4j (log4j-api-2.12.1.jar)

Vulnerabilities (CVE-2021-44228, CVE-2021-45046): Upgrade to log4j version 2.16.0. Also, could update to log4j2 which will also take care of these vulnerabilities.

SnakeYAML (snakeyaml-1.25.jar)

Vulnerabilities (CVE-2017-18640, CVE-2022-1471, CVE-2022-25857, CVE-2022-3064): Upgrade to SnakeYAML version 2.0 or later.

FasterXML Jackson Databind (jackson-databind-2.10.2.jar)

Vulnerabilities (CVE-2020-25649, CVE-2020-36518, CVE-2021-46877, CVE-2022-42004): Update to version 2.13.1.

Apache Tomcat (tomcat-embed-core-9.0.30.jar)

Vulnerabilities (CVE-2020-11996, CVE-2020-13934, CVE-2020-13935, CVE-2020-1938, CVE-2020-8022): Upgrade to the latest Tomcat9 version.

Spring Web (spring-web-5.2.3.RELEASE.jar)

Vulnerabilities (CVE-2016-1000027, CVE-2021-22118): Ensure there are no HTTP Invoker endpoints exposed to untrusted clients. And upgrade to version 5.3.7

Spring Beans (spring-beans-5.2.3.RELEASE.jar)

Vulnerability (CVE-2022-22965): Upgrade to Spring Framework version 5.3.18.

If we implement these changes to the code, I feel that it will be secure and safeguarded from any vulnerabilities discussed or found in the dependency check.