****

# 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** | **5/21/2023** | **Nickolas Dober** |  |

## 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

Nickolas Dober

## Interpreting Client Needs

Figure out client’s vulnerability and needs determined by their security threats and application use. It is useful to consider the client’s field of use and suspected vulnerabilities to maximize efficiency and protection. Use the following to help determine client needs:

* What is the value of secure communications to the company?
* Does the company make any international transactions?
* Are there governmental restrictions about secure communications to consider?
* What external threats might be present now and in the immediate future?
* What are the modernization requirements that you must consider?

Artemis financial is a consulting company who develops individualized financial plans for customers. This would mean that the value of secure communications for their customers and themselves is of top importance. This is due to the private nature of financial transactions that would need to take place over a secure connection. As for Artemis Financial doing any international transactions, there is no indication that the company is solely focused on non-international transactions. It is safe to assume that with Artemis Financial being a financial consulting company, there is the possibility for international transactions. With Artemis dealing with financial business, there will be certain governmental restrictions tied to exposure of trade information. External threats are a strong possibility. Client data and information as well as that of the company and its employees is of high value and therefore poses a threat to data breaches. Data and transactions will need to be encrypted to prevent external threats. This elevated need for security will also mean a need for maintenance and updates on a regular basis to take advantage of any security updates.

## Areas of Security

Input Validation: In order to secure user information, input validation is needed to validate the owner of any data that is trying to be accessed. This provides protection for users against unauthorized access to their data.

Code Quality: It would be suitable to include access control to different users. This would designate certain amount or fields of access for different users so that clients don’t have access to other client data and so on.

APIs: Using APIs creates a set of procedures to access data from the system or service. This allows for more secure connections, preventing unwanted system access.

Code Error: Handling of code errors is vital to keeping user data from being exposed. Utilizing error handling will reduce the risk for data leaks.

Cryptography: Due to the possible use of foreign currencies, cryptography implementation would be useful in keeping user information from being compromised.

## Manual Review

Manual review provided the following observations:

* No test values.
* No test cases.
* No entry criteria.
* No exit criteria.
* Spring maven plugin can be updated.

## Static Testing

**Dependency:** 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.”

**Vulnerabilities:** CVE-2016-1000338. CVE-2016-1000342, CVE-2016-1000343, CVE-2016-1000344, CVE-2016-1000352, CVE-2016-1000341, CVE-2016-1000345, CVE-2017-13098, CVE-2020-15522, CVE-2020-0187 (OSSINDEX), CVE-2016-1000339, CVE-2020-26939 (OSSINDEX), CVE-2015-7940, CVE-2018-5382, CVE-2013-1624, CVE-2016-1000346, CVE-2015-6644 (OSSINDEX)

**Description & Solutions:**

1. Bouncy Castle JCE Provider version 1.55 and earlier the DSA does not fully validate ASN.1 encoding of signature on verification. Bouncy Castle JCE Provider version 1.55 and earlier ECDSA does not fully validate ASN.1 encoding of signature on verification.
   * Solution: Inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure.
2. Bouncy Castle JCE Provider version 1.55 and earlier the DSA key pair generator generates a weak private key if used with default values. If the JCA key pair generator is not explicitly initialized with DSA parameters, 1.55 and earlier generates a private value assuming a 1024-bit key size.
   * Solution: Can be dealt with by explicitly passing parameters to the key pair generator.
3. Bouncy Castle JCE Provider version 1.55 and earlier the DHIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.
   * Solution: Upgrade Bouncy Castle JCE Provider version to newer than 1.55.
4. Bouncy Castle JCE Provider version 1.55 and earlier DSA signature generation is vulnerable to timing attack. Where timings can be closely observed for the generation of signatures, the lack of blinding in 1.55, or earlier, may allow an attacker to gain information about the signature's k value and ultimately the private value as well.
   * Solution: Upgrade Bouncy Castle JCE Provider version to newer than 1.55
5. Bouncy Castle JCE Provider version 1.55 and earlier the DHIES/ECIES CBC mode vulnerable to padding oracle attack. For BC 1.55 and older, in an environment where timings can be easily observed, it is possible with enough observations to identify when the decryption is failing due to padding.
   * Solution: Upgrade Bouncy Castle JCE Provider version to newer than 1.55
6. BouncyCastle TLS prior to version 1.0.3, when configured to use the JCE (Java Cryptography Extension) for cryptographic functions, provides a weak Bleichenbacher oracle when any TLS cipher suite using RSA key exchange is negotiated. An attacker can recover the private key from a vulnerable application. This vulnerability is referred to as "ROBOT."
   * Solution: Update BouncyCastle TLS
7. Bouncy Castle BC Java before 1.66, BC C# .NET before 1.8.7, BC-FJA before 1.0.1.2, 1.0.2.1, and BC-FNA before 1.0.1.1 have a timing issue within the EC math library that can expose information about the private key when an attacker is able to observe timing information for the generation of multiple deterministic ECDSA signatures.
   * Solution: Update BouncyCastle BC Java to after 1.66.
8. In engineSetMode of BaseBlockCipher.java, there is a possible incorrect cryptographic algorithm chosen due to an incomplete comparison. This could lead to local information disclosure with no additional execution privileges needed. User interaction is not needed for exploitation.
   * Solution: Address incorrect cryptographic algorithm in use.
9. In Legion of the Bouncy Castle BC before 1.61 and BC-FJA before 1.0.1.2, attackers can obtain sensitive information about a private exponent because of Observable Differences in Behavior to Error Inputs. This occurs in org.bouncycastle.crypto.encodings.OAEPEncoding. Sending invalid ciphertext that decrypts to a short payload in the OAEP Decoder could result in the throwing of an early exception, potentially leaking some information about the private exponent of the RSA private key performing the encryption.
   * Solution: Update BouncyCastle BC to newer than 1.61.
10. The Bouncy Castle Java library before 1.51 does not validate a point is withing the elliptic curve, which makes it easier for remote attackers to obtain private keys via a series of crafted elliptic curve Diffie Hellman (ECDH) key exchanges, aka an "invalid curve attack."
    * Solution: Update BouncyCastle BC to newer than 1.51.
11. The default BKS keystore use an HMAC that is only 16 bits long, which can allow an attacker to compromise the integrity of a BKS keystore. Bouncy Castle release 1.47 changes the BKS format to a format which uses a 160 bit HMAC instead. This applies to any BKS keystore generated prior to BC 1.47. For situations where people need to create the files for legacy reasons a specific keystore type "BKS-V1" was introduced in 1.49.
    * Solution: Update BouncyCastle version to newer than 1.49.

**Dependency: hibernate-validator-6.0.18.Final.jar**

**Vulnerabilities:** CVE-2020-25649, CVE-2020-36518, CVE-2021-46877, CVE-2022-42003, CVE-2022-42004

**Description & Solutions:**

1. A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.
   * Solution: Secure entity expansion in FasterXML or update.
2. jackson-databind before 2.13.0 allows a Java StackOverflow exception and denial of service via a large depth of nested objects.
3. 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.
4. In FasterXML jackson-databind before 2.14.0-rc1, resource exhaustion can occur because of a lack of a check in primitive value deserializers to avoid deep wrapper array nesting, when the UNWRAP\_SINGLE\_VALUE\_ARRAYS feature is enabled. Additional fix version in 2.13.4.1 and 2.12.17.1
5. 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.
   * Solution: Update jackson-databind to newer than 2.14.0.

**Dependency:** log4j-api-2.12.1.jar

**Vulnerabilities:** CVE-2020-9488

**Description & Solutions:**

1. Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1
   * Solution: Update Apache Log4j to newer than 2.13.1.

**NOTE:** Many more vulnerabilities were displayed by the dependency check report. Solutions all related to updating plugin version due to security and maintenance updates. Dependencies in question are:

* **logback-core-1.2.3.jar**
* **snakeyaml-1.25.jar**
* **spring-boot-2.2.4.RELEASE.jar**
* **spring-boot-autoconfigure-2.2.4.RELEASE.jar**
* **spring-boot-starter-web-2.2.4.RELEASE.jar**
* **spring-core-5.2.3.RELEASE.jar**
* **spring-web-5.2.3.RELEASE.jar**
* **tomcat-embed-core-9.0.30.jar**
* **tomcat-embed-websocket-9.0.30.jar**

## Mitigation Plan

Most major issues with dependencies were due to outdated plugins. The vast majority of vulnerabilities displayed on the dependency check report were issues with plugins, with solutions stating newer versions that addressed certain errors and vulnerabilities. Security risks due to errors in these outdated plugins will be resolved with updated plugins. The static test revealed what vulnerabilities were and what the proper actions are to take after reviewing the suggested solutions.