

# 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](#_Toc32574614) 5

[5. Mitigation Plan](#_Toc32574615) 7

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **September 13, 2022** | **Adrian Sanchez** |  |

## 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

Adrian Sanchez

## Interpreting Client Needs

Global Rain’s mission statement is, “Security is everyone’s responsibility.” Secure communications is vital to Global Rain because the integrity of our customers data, to include entrepreneurs, businesses, and government agencies across the world, is on the line. The scope of Artemis Financial’s web-based application could jeopardize savings, retirement, investments, and insurance accounts. All of these attributes carry high threat. The goal is to use best practices across many scopes, to include legal/governmental/international compliance to secure communications. Although a full cyber security scope cannot be 100% secure, we will do our best to ensure external & internal security features are implemented and only the people/applications with the “need-to-know” are “in-the-loop”. Since we live in a dynamically changing technological environment, implementation of best practices for modernization/standardization is critical. We should include attributes such as scalability, reliability, recover-ability, and maintainability/manageability when incorporating safety/security features into the software to ensure everyone’s data is secure without compromise.

## Areas of Security

Security assessment for Artemis Financial’s web-based application should be focused around:

* Cryptography – Artemis Financial’s software will have access to vital client information that needs to be encrypted.
* APIs & client/server– In order for an API to be considered RESTful, it has to conform to certain security features. Artemis Financial has a RESTful API, but specific attention to the client-server architecture should be thoroughly inspected to ensure personally identifiable information cannot be accessed.
* Input Validation – Artemis Financial uses customer data, which would need to be validated to ensure protection to the customers, input as strings. Input from untrusted sources should be validated before being used.
* Code Quality – Finally, we need to ensure comments and code objects containing confidential information is purged after it’s no longer needed. The non-purged information could reveal the confidential information if accessed.

## Manual Review

Outlined below are the security findings after inspection of the applications code:

* pom.xml file – The presence of apache validator could not be found. Should focus on getting Sprint Security with Maven and incorporate it.
* Cryptography – Encryption of sensitive information could not be observed, such as account numbers & account balance (customer.java), or input validation in GreetingController.java when accessing Greeting.java.
* Code quality – Needs to be improved since purging of sensitive information wasn’t found.
* Client/Server & API – A default method is being used in the RestServiceApplication.java file. This is not safe and should have specific criteria. After running the application sensitive information was displayed that should be hidden (such as system information). No client-server problems were found at this time.

## Static Testing

The relevant vulnerability codes below were found after a dependency-check was run on the program:

* CVE-2016-1000338 – Improper verification of cryptographic signature which could lead to the validation of signatures injected with extra elements.
* CVE-2016-1000342 - Improper verification of cryptographic signature which could lead to the validation of signatures injected with extra elements.
* CVE-2016-1000343 – Cryptographic issue if the JCA key pair generator is not explicitly initialised with DSA parameters.
* CVE-2016-1000344 – Cryptographic issue. Program allows the use of ECB mode which is regarded as unsafe and support for it has been removed by the provider.
* CVE-2016-1000352 - Cryptographic issue. Program allows the use of ECB mode which is regarded as unsafe and support for it has been removed by the provider.
* CVE-2016-1000341 – Timing & state issue where the earliest DSA signature generation is vulnerable to timing attack and can be closely observed by attackers to gain information about the signature’s k value and ultimately the private value as well.
* CVE-2016-1000345 - Timing & state issue where the DHIES/ECIES CBC mode is vulnerable to a padding oracle attack where it’s possible with enough observation to identify when the decryption is failing due to padding.
* CVE-2017-13098 – Observable discrepency issue. When using JCE for cryptographic functions, provides a weak Bleichenbacher oracle when TLS cipher suite using RSA key exchange is negotiated. An attacker can recover the private key from a vulnerable application.
* CVE-2020-15522 - 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.
* CVE-2020-0187 (OSSINDEX) - 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.Product: AndroidVersions: Android-10Android ID: A-148517383
* CVE-2016-1000339 - The primary engine class used for AES was AESFastEngine. Due to the highly table driven approach used in the algorithm it turns out that if the data channel on the CPU can be monitored the lookup table accesses are sufficient to leak information on the AES key being used. There was also a leak in AESEngine although it was substantially less. AESEngine has been modified to remove any signs of leakage (testing carried out on Intel X86-64) and is now the primary AES class for the BC JCE provider from 1.56. Use of AESFastEngine is now only recommended where otherwise deemed appropriate.
* CVE-2020-26939 (OSSINDEX) - 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.
* CVE-2015-7940 - 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."
* CVE-2018-5382 - 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. It should be noted that the use of "BKS-V1" is discouraged by the library authors and should only be used where it is otherwise safe to do so, as in where the use of a 16 bit checksum for the file integrity check is not going to cause a security issue in itself.
* CVE-2013-1624 - The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169.
* CVE-2016-1000346 - In the Bouncy Castle JCE Provider version 1.55 and earlier the other party DH public key is not fully validated. This can cause issues as invalid keys can be used to reveal details about the other party's private key where static Diffie-Hellman is in use. As of release 1.56 the key parameters are checked on agreement calculation.
* CVE-2015-6644 (OSSINDEX) - Bouncy Castle in Android before 5.1.1 LMY49F and 6.0 before 2016-01-01 allows attackers to obtain sensitive information via a crafted application, aka internal bug 24106146.
* 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.
* CVE-2020-25649 - 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.
* CVE-2020-36518 - jackson-databind before 2.13.0 allows a Java StackOverflow exception and denial of service via a large depth of nested objects.
* CVE-2020-9488 - 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
* CVE-2021-42550 - 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.
* CVE-2017-18640 - The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.
* CVE-2022-25857 - The package org.yaml:snakeyaml from 0 and before 1.31 are vulnerable to Denial of Service (DoS) due missing to nested depth limitation for collections.
* CVE-2022-38749 - 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 stackoverflow.
* CVE-2022-38751 - 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 stackoverflow.
* CVE-2022-38752 (OSSINDEX) - 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.
* CVE-2022-38750 - 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 stackoverflow.
* **spring-core-5.2.3.RELEASE.jar**: CVE-2022-22965 – Improper control and Generation of code. CVE-2021-22118 – Improper privilege management. CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, & CVE-2021-22096 – Deals with malicious input, RFD & DoS attacks, and improper handling of case sensitivity. These vulnerabilities can be dealt with by upgrading to the current Spring Framework.
* **spring-web-5.2.3.RELEASE.jar & spring-boot-2.2.4.RELEASE.jarspring-boot-2.2.4.RELEASE.jar**: All CVE IDs associated by these dependencies can be dealt with by updating to a current release of the Spring Framework.
* **tomcat-embed-core-9.0.30.jar & tomcat-embed-websocket-9.0.30.jar**: Many CVEs are associated with these dependencies and can be fixed for free by going to https://security.snyk.io/package/maven/org.apache.tomcat.embed:tomcat-embed-core

## Mitigation Plan

Suggestions for improvement by area:

Overall – Most if not all of the security risk was found in running older version of Snakeyaml, Hybernate Validator, Apache Tomcat, BouncyCastle, and the Spring Framework. Artemis Financial should switch to the most recent installments where security features are in place for these dependencies. This would be a major improvement to security.