****

# 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** | **09/20/2023** | **Kaitlyn Lush** | **Reviewed clients needs, security, testing, and mitigation plan** |

## 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

Kaitlyn Lush

## Interpreting Client Needs

Artemis Financial is a company that provides financial services to their customers. These services include savings, retirement, investments, and insurance. Due to this, the company is handling personal and sensitive information about their customers. This means that this sensitive information could become the target of a security attack. This means that important customer information could be leaked such as identity theft as well as the loss of funds. By making sure the communications are secure we can ensure that the consumer as well as any other parties involved remain safe from any attacks. It is possible that Artemis Financial would be making international transactions. Whether the consumer lives abroad or a citizen traveling in a different country, the consumer will need access to their money regardless of where they are. Since the company is dealing with important and sensitive information, it is extremely likely that there will be government policies put in place to regulate it. This means that the company will need to follow any policies the government has. When dealing with such personal information there are many different ways this company can be at risk of an attack. If a consumer’s username and password get leaked this would give the attacker direct access to the customer’s information. These attacks include threats such as malware attacks, phishing, ransomware, and DoS attacks.

## Areas of Security

When looking at the application there are several areas of security that would apply. The areas listed below are the security applications that I think directly apply:

* The first area would be Input Validation. In order to keep this software safe, it is important that any information input into the system is validated to prevent any vulnerabilities.
* The next area that applies is API. Since this company uses a RESTful API, it is important that this API is tested for any vulnerabilities. This will prevent any issues with this API from occurring.
* Another area that applies is Cryptography. Since this company deals directly with money and other sensitive information pertaining to their customers it is essential that encryption is applied to the system. This will prevent any attackers from gaining access to these important files.
* Another important area involving this product is Code Error. It is important that the code is error free to ensure that there are no vulnerabilities within the code itself.
* Client/Server would also be extremely important for this product. Client servers allow for security features that are able to secure data.

## Manual Review

By reviewing the code, I was able to identify a couple of vulnerabilities that could cause a potential security threat.

In the DocData.java file and within the public DocData Class the read document method allows for a root username and root password. This use of root is not recommended as it could allow the password to be easily guessed. It could also allow for unauthorized users to perform a forceful attack to gain access.

Within the CRUDController.java file there was a vulnerability that I noticed. Within its public class it allows for the value “business\_name” to pass through the CRUD method which could allow for the DocData object to be exposed to potential vulnerabilities. This allows for the internal objects to be exposed to potential data breaches.

In the GreetingController.java file GET is used instead of POST to request a parameter. This is less secure as it will put sent data as part of the URL.

## Static Testing

1. **bcprov-jdk15on-1.46.jar**

* **CVE-2016-1000338 and CVE-2016-1000342:** These vulnerabilities allow for extra elements to be injected with may allow for the introduction of invisible data.
  + **CWE-347** Improper Verification of Cryptographic Signature
* **CVE-2016-1000343:** This vulnerabilityallows for the key pair generator to generate a weak private key if used with default values.
  + **CWE-310** Cryptographic Issues
* **CVE-2016-1000344:** The DHIES implementation allowed for the use of ECB mode. It’s been designated unsafe, and support for it has been removed by the provider.
  + **CWE-310** Cryptographic Issues
* **CVE-2016-1000352:** The ECIES implementation allowed for the use of ECB mode. It’s been designated unsafe, and support has been removed by the provider.
  + **CWE-310** Cryptographic Issues
* **CVE-2016-1000341:** The signature generation is vulnerable to a timing attack. It could allow for an attacker to gain information about the signature’s k and private value.
  + **CWE-361 7PK** Time and State
* **CVE-2016-1000345:** DHIES/ECIES CBC mode made the program vulnerable to padding oracle attack and timings were also easily observed.
  + **CWE-361 7PK** Time and State
* **CVE-2017-13098** Allows an attacker to recover the private key from a vulnerable application.
  + **CWE-203** Observable Discrepancy.
* **CVE-2020-15522** Has a timing issue within the EC math library that can expose information about the private key when an attacker observes the timing information.
  + **CWE-362** Concurrent Execution using Shared Resource with Improper Synchronization.
* **CVE-2020-0187 (OSSINDEX)** There is a potentially incorrect algorithm chosen due to an incomplete comparison which could allow exploitation without user interaction.
  + **CWE\_310** Cryptographic Issues
* **CVE-2016-1000339** The primary engine class used for AES was allowing for leaked information through the data channel if it was able to be monitored.
  + **CWE\_310** Cryptographic Issues
* **CVE-2020-26939 (OSSINDEX)** allows attackers to obtain sensitive information about a private exponent due to observable differences in behavior to Error Inputs.
  + **CWE\_203** Observable Discrepancy
* **CVE-2023-33201 (OSSINDEX)** affected by a LDAP injection vulnerability. It only affects applications that use LDAP CertStore from BouncyCastle to validate X.509 certificates.
  + **CWE-295** Improper Certificate Validation.
* **CVE-2015-7940** allows for remote hackers to obtain private keys through a series of crafted elliptic curve Diffie Hellman key exchanges.
  + **CWE-310** Cryptographic Issues and **CWE-200** Exposure of Sensitive Information to an Unauthorized Actor
* **CVE-2018-5382** The default BKS keystore uses a HMAC that is only 16 bits long, which can allow an attacker to compromise the integrity of a BKS keystore.
  + **CWE-354** Improper Validation of Integrity Check Value
* **CVE-2013-1624** The TLS implementation does not properly consider timing side-channel attacks on a noncompliant MAC check operation during malformed CBC padding processing. This allows for attackers to conduct attacks based on the timing data.
  + **CWE-310** Cryptographic Issues
* **CVE-2016-1000346** Does not fully validate the DH public key which allows details to be revealed about the other party’s private key.
  + **CWE-320** Key Management Errors
* **CVE-2015-6644 (OSSINDEX)** Allows attackers to obtain sensitive information through a crafted application or an internal bug.
  + **CWE-200** Exposure of Sensitive Information to an Unauthorized Actor

1. **hibernate-validator-6.0.18.Final.jar**

* **CVE-2020-10693** A flaw was found in the message interpolation processor that enables invalid EL expressions to be evaluated as valid. This allows attackers to bypass input sanitation.
  + **CWE-20** Improper Input Validation
    - **It is suggested that the package be updated to the most recent version** **hibernate-validator-6.0.20.**

1. **jackson-databind-2.10.2.jar**

* **CVE-2020-25649** A flaw was found that allows vulnerability to XML extern entity attacks.
  + **CWE-611** Improper Restriction of XML External Entity Reference
* **CVE-2020-36518** Allows a Java StackOverflow exception and denial of service through a large depth of nested objects.
  + **CWE-787** Out of Bounds Write
* **CVE-2021-46877** Allows attackers to cause a denial of service involving JDK serialization.
  + **CWE-770** Allocation of Resources Without Limits or Throttling
* **CVE-2022-42003** Resource exhaustion can occur due to a lack of a check in primitive value deserializers to avoid deep array nesting when the UNWRAP\_SINGLE\_VALUE\_ARRAYS feature is active.
  + **CWE-502** Deserialization of Untrusted Data
* **CVE-2022-42004** Resource exhaustion can occur due to a lack of a check in BeanDeserializer.\_deserializeFromArray to prevent the use of deeply nested arrays
  + **CWE-502** Deserialization of Untrusted Data
* **CVE-2023-35116** Allows an attacker to cause a denial of service through a crafted object that uses cyclic dependencies. (The vendor states that it’s not a valid vulnerability because it cannot be achieved by an external attacker.)
  + **CWE-770** Allocation of Resources Without Limits or Throttling
    - **It is suggested that the package be upgraded to the patched version 2.13.1.**

1. **log4j-api-2.12.1.jar**

* **CVE-2020-9488** Allows improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow for a man-in-the-middle attack that could leak any log messages sent through that appender.
  + **CWE-295** Improper Certificate Validation
    - **It is suggested that the package be upgraded to a 2.16.0 version as soon as possible to resolve this issue.**

1. **logback-core-1.2.3.jar**

* **CVE-2021-42550** Allows for an attacker with the required privileges to edit configuration files could allow arbitrary code to be loaded from LDAP servers.
  + **CWE-502** Deserialization of Untrusted Data
    - **It is suggested that the package be upgraded to the newest patch 1.3.0-alpha11, 1.2.9**

1. **snakeyaml-1.25.jar**

* **CVE-2022-1471** Does not restrict types that can be instantiated during deserialization. This can cause an attacker to provide remote code execution through deserializing yaml content.
* **CWE-502** Deserialization of Untrusted Data
* **CVE-2017-18640** A vulnerability issue related to CVE 2003-1564. This vulnerability allows the alias feature in SnakeYAML before 1.26 to have an entity expansion during a load operation.
* **CWE-776** Improper Restriction of Recursive Entity References in DTDs
* **CVE-2022-25857** A vulnerability that leads to Denial of Service. The package org.yaml: from 0 and before 1.31 are vulnerable due to missing nested depth limitation for collections.
* **CWE-776** Improper Restriction of Recursive Entity References in DTDs
* **CVE-2022-38749****, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854, CVE-2022-38750** Vulnerability of Denial-of-Service attacks. Uses snakeYML to parse untrusted YAML files. If the parser is running on user supplied input an attacker could supply content that causes the parser to crash by StackOverflow.
* **CWE-787** Out-of-Bounds Write
  + - **It is suggested that the package be upgraded to version 2.0 or higher as soon as possible**

1. **spring-boot-2.2.4.RELEASE.jar and spring-boot-starter-web-2.2.4.RELEASE.jar**

* **CVE-2023-20873** This vulnerability is susceptible to a security bypass. Spring Boot versions 3.0.0-3.0.5, 2.7.0-2.7.10 and older are affected by this vulnerability.
* **CVE-2022-27772** A vulnerability to temporary directory hijacking. Spring Boot versions prior to version v 2.2.11 release was affected. The vulnerability only affects products/ versions that are no longer supported by the maintainer.
* **CWE-668** Exposure of Resource to Wrong Sphere
* **CVE-2023-20883** The vulnerability causes a potential for a denial-of-service attack. This vulnerability is created by using Spring MVC and a reverse proxy cache together in Spring Boot versions 3.0.0-3.0.6, 2.7.0-2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older.
* **CWE-400** Uncontrolled Resource Consumption.
  + - **It is highly suggested that these packages be updated to a newer version such as 3.0.6+ as soon as possible.**

1. **spring-core-5.2.3.RELEASE.jar, spring-web-5.2.3.RELEASE.jar, and spring-webmvc-5.2.3.RELEASE.jar**

* **CVE-2022-22965** CISA Known Exploited Vulnerability. A Spring MVC or Spring WebFlux app running on JDK 9+ is vulnerable to remote code execution via data binding. This vulnerability requires the application to run on Tomcat as a WAR deployment.
* **CWE-94** Improper Control of Generation of Code
* **CVE-2021-22118** Vulnerable to privilege escalation. It does this by recreating the temporary storage directory in Spring Framework versions 5.2.x prior to 5.2.15 and versions 5.3.x prior to 5.3.7
* **CWE-668** Exposure of Resource to Wrong Sphere.
* **CVE-2020-5421** Protection against RFD attacks from CVE-2015-5211 may be bypassed depending on browser. Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older are susceptible through the use of a jsessionid path parameter.
* **CVE-2022-22950** Allows for a potential denial-of-service through a user providing a specially crafted SpEL expression.
* **CWE-770** Allocation of Resources Without Limits or Throttling.
* **CVE-2022-22971** Vulnerable to a denial-of-service attack by an authenticates user. Spring Framework versions prior to 5.3.20+ , 5.2.22+ and older are susceptible to using STOMP over Websocket endpoints.
* **CWE-770** Allocation of Resources Without Limits or Throttling.
* **CVE-2023-20861** Allows for a potential denial-of-service through a user providing a specially crafted SpEL expression. Affects Spring Framework versions 6.0.0 - 6.0.6, 5.3.0 - 5.3.25, 5.2.0.RELEASE - 5.2.22.RELEASE, and older.
* **CVE-2023-20863** Allows for a potential denial-of-service through a user providing a specially crafted SpEL expression. Affects Spring Framework versions prior to 5.2.24 release+ ,5.3.27+ and 6.0.8+.
* **CVE-2022-22968** Allows for a field to be susceptible through case sensitivity. Spring Framework versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20, and older
* **CWE-178** Improper Handling of Case Sensitivity
* **CVE-2022-22970** Vulnerable to DoS attacks. In spring framework versions prior to 5.3.20+, 5.2.22+ and older if the application relies on data binding to set a MultipartFile or javax.servlet.Part to a field in a model object.
* **CWE-770** Allocation of Resources Without Limits or Throttling
* **CVE-2021-22060, CVE-2021-22096** Allows user to provide malicious inputs. Spring Framework versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18, and older are affected.
* **CVE-2016-1000027 (spring-web-5.2.3.RELEASE.jar)** Allows for a potential remote code execution. Applies when used for Java deserialization of untrusted data.
* **CWE-502** Deserialization of Untrusted Data
  + - **It is advised that the package should be updated to the latest patch.**

1. **tomcat-embed-core-9.0.30.jar and tomcat-embed-websocket-9.0.30.jar**

* **CVE-2020-1938** CISA Known Exploited Vulnerability. Apache Tomcat versions 9.0.0M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99 are susceptible to improper privilege management issues that are related to the Apache JServ Protocol connections.
* **CVE-2020-11996** Allows for the server to become unresponsive. A crafted sequence of HTTP/2 requests sent to Apache Tomcat 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 can trigger high CPU usage for several seconds.
* **CVE-2020-13934** Can lead to a denial of service. A h2c direct connection to Apache Tomcat 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 did not release the HTTP/1.1 processor after the upgrade.
* **CWE-401** Missing Release of Memory after Effective Lifetime
* **CVE-2020-13935** Invalid loops can lead to a denial of service. This was due to a payload length in a websocket frame that was not correctly validated.
* **CWE-835** Loop with Unreachable Exit Condition
* **CVE-2020-17527** Could allow for information to be leaked between requests. While investigating a bug was discovered that allowed Apache Tomcat to re-use a HTTP request header value from the previous stream
* **CWE-200** Exposure of Sensitive Information to an Unauthorized Actor
* **CVE-2021-25122** Allows for results to be shared among other users. This happens by allowing duplicate request headers and a limited amount of request body from one request to another.
* **CWE-200** Exposure of Sensitive Information to an Unauthorized Actor
* **CVE-2021-41079** An infinite loop that could lead to a denial of service. Apache Tomcat did not properly validate incoming TLS packets allowing it to trigger an infinite loop.
* **CWE-835** Loop with Unreachable Exit Condition
* **CVE-2022-29885** does not protect against risks from running over any untrusted network. Apache Tomcat versions 10.1.0-M1 to 10.1.0-M14, 10.0.0-M1 to 10.0.20, 9.0.13 to 9.0.62 and 8.5.38 to 8.5.78 allow for DoS attacks.
* **CWE-400** Uncontrolled Resource Consumption
* **CVE-2022-42252** Does not reject a request containing an invalid Content-Length header. This allows for a request smuggling attack if tomcat was located behind a reverse proxy that failed to reject the request.
* **CWE-444** Inconsistent Interpretation of HTTP Requests
* **CVE-2020-9484** Allows for an attacker to trigger remote code execution through deserialization. The following statements must be true in order for the attack to succeed. If the attacker using tomcat versions 10.0.0-M1 to 10.0.0-M4, 9.0.0.M1 to 9.0.34, 8.5.0 to 8.5.54 and 7.0.0 to 7.0.103 is able to control the contents and name of a file on the server, if the server is configured to use PersistenceManager with a FileStore, if the PersistenceManager is configured with sessionAttributeValueClassNameFilter = “null”, and if the attacker knows the relative file path from the storage location used by FileStore to the file the attacker has control over, then the attack will succeed.
* **CWE-502** Deserialization of Untrusted Data
* **CVE-2021-25329** The fix for CVE-2020-9484 was incomplete and this allowed for vulnerabilities.
* **CVE-2021-30640** Allowed an attacker to authenticate using variations of a valid username and/or bypass some of the protection provided by the LockOutRealm.
* **CWE-116** Improper Encoding or Escaping of Output
* **CVE-2022-34305** A XXS vulnerability was found.
* **CWE-79** Improper Neutralization of Input During Web Page Generation
* **CVE-2023-41080** URL redirection to Untrusted site vulnerability.
* **CWE-601**
* **CVE-2021-24122** Allowed for vulnerability to JSP source code disclosure in some configurations.
* **CWE-706** Use of Incorrectly Resolved Name or Reference
* **CVE-2021-33037** Did not correctly parse the HTTP transfer-encoding request header leading to possible request smuggling when used with a reverse proxy.
* **CWE-444** Inconsistent Interpretation of HTTP Requests
* **CVE-2019-17569** Invalid transfer-encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if located behind a reverse proxy.
* **CWE-444** Inconsistent Interpretation of HTTP Requests
* **CVE-2020-1935** The HTTP header parsing code used an approach to end of line parsing that allowed invalid HTTP headers to be parsed as valid. Apache Tomcat versions 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99 led to a possibility of HTTP Request smuggling if it was located behind a reverse proxy that incorrectly handled the invalid Transfer Encoding header.
* **CWE-444** Inconsistent Interpretation of HTTP Requests
* **CVE-2020-13943** Allowed for users to see responses for unexpected resources. In Apache Tomcat versions 10.0.0-M1 to 10.0.0-M7, 9.0.0.M1 to 9.0.37 or 8.5.0 to 8.5.57 exceeded the agreed maximum number of concurrent streams for a connection it made it possible that a request made on that connection could contain HTTP headers from a previous request.
* **CVE-2023-28708** Allowed for the user to transmit the session cookie over an insecure channel. In Apache Tomcat versions 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 when using the RemotelpFilter with requests from a reverse proxy via HTTP that include the X-Fowarded-Proto set to https, it did not include the secure attribute.
* **CWE-523** Unprotected Transport of Credentials.
* **CVE-2021-43980** Allowed for responses or part responses to be received by the wrong client. This was caused by the simplified implementation of blocking reads and writes introduced in Tomcat 10 and backported to Tomcat 9.0.47 onwards. It exposed a long-standing bug that could cause client connections to share a HTTP 11 processor.
* **CWE-362** Concurrent Execution using Shared Resource
  + - **These packages need to be updated to a later version such as 11.0.0-M11 or later as soon as possible.**

## Mitigation Plan

In order to solve the issues within this application the first thing I would suggest doing is updating the packages used within the program. By ensuring that these packages are updated it will ensure that many of the vulnerabilities that were identified by the scan are taken care of. Listed below are the vulnerabilities that were identified as well as ways to fix the issue:

1. **bcprov-jdk15on-1.46.jar** This package has 18 vulnerabilities with a high severity level. To solve this, it is highly recommended that the package be updated to the newest version.
2. **hibernate-validator-6.0.18.Final.jar** This package has one vulnerability that is of medium severity level. In order to resolve this problem, it is suggested that the package be upgraded to hibernate-validator-6.0.20.
3. **jackson-databind-2.10.2.jar** This package has 6 vulnerabilities with a high severity level. To resolve this problem, it is suggested that the package be upgraded to the patched version 2.13.1.
4. **log4j-api-2.12.1.jar** This package has one vulnerability with a low severity level. It is suggested that the package be upgraded to a 2.16.0 version as soon as possible to resolve this issue.
5. **logback-core-1.2.3.jar** This package has one vulnerability with a medium severity level. A patched version of this package is available to use. (1.3.0-alpha11, 1.2.9)
6. **snakeyaml-1.25.jar** This package has 8 vulnerabilities with a critical severity. It is highly suggested that the package be upgraded to version 2.0 or higher as soon as possible to resolve this issue.
7. **spring-boot-2.2.4.RELEASE.jar and spring-boot-starter-web-2.2.4.RELEASE.jar** These packages have 3 vulnerabilities with a critical severity. It is highly suggested that they be updated to a newer version such as 3.0.6+ as soon as possible to resolve this issue.
8. **spring-core-5.2.3.RELEASE.jar and spring-webmvc-5.2.3.RELEASE.jar** This package has 11 vulnerabilities with a critical severity. It is advised that the package should be updated to the latest patch. It is important to make sure that the user is inputting the correct version as several of the versions have known bugs associated with them.
9. **spring-web-5.2.3.RELEASE.jar** This package has 12 vulnerabilities with critical severity. It is highly advised that the package be updated to the latest patch.
10. **tomcat-embed-core-9.0.30.jar and tomcat-embed-websocket-9.0.30.jar** This package has more than 20 vulnerabilities with a critical severity. It is important that this package gets updated to a later version such as 11.0.0-M11 or later as soon as possible.

After updating all of the packages being used the next step would be looking at the data access username and password. In order to ensure that the application is safe and secure all passwords should be required to be a strong combination of numbers, letters, and other characters. The passwords also should not contain any part of the username in them as it could be easily hacked. In order to access the database within the application the user should be created separate from the root.

Another important step would be to review the code written. The code should be looked over to ensure that any code that needs to be modified gets reviewed. This will ensure that secure code practices are being used within the program. Authentication needs to be implemented as well as error catching in order to prevent any potential attacks.

There needs to be a check implemented on both sides of the server, server/client through certificates to prevent the bouncy castle vulnerabilities. This also means that the certificates put in place need to be validated. The program needs to be cleaned to allow for the proper validation of these certificates.

The final step involves updating the Apache server to the most recent version. Performing regular updates on the packages that are being used within your software can ensure that any vulnerabilities are taken care of. This will ensure that your program remains secure from any security attacks.

**References**

<file:///C:/Users/klush/eclipse-workspace/rest-service/target/dependency-check-report.html>