****

# 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** | **[Date]** | **[Your name]** |  |

## 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

Robert Hardial

## Interpreting Client Needs

* The value of secure communications in this company is high. Secure communications ensures that sensitive financial data, such as account information and transaction details, is protected from unauthorized access. Secure communications also increase the confidence of customers because they can trust the company. The use of secure communications can prevent financial fraud and protect the integrity of the company. The web application will allow global customers to invest with Artemis Financial so there will be international transactions occurring. The transactions will involve the purchasing and sale of stock. There will be government restrictions about secure communications to consider. The regulations will differ based on the different countries it operates in. There may be strict rules on the use of encryption or other security measures to protect sensitive financial information. There could also be regulations about the storage and sharing of financial data. An external threat the company can face now and, in the future includes being targeted by hackers looking to steal sensitive financial information. This can include phishing attacks, malware infections, and ransomware attacks. The modernization requirements that must be considered include the role of open-source libraries and evolving web application technologies. Open-source libraries can supply a cost-effective way to access a wide range of software development technologies. This can help companies to quickly build new applications and services. They also need to consider staying current on the latest web application technologies and trends. These technologies can help companies more effectively manage their data and operations.

## Areas of Security

* APIs - A secure API is an important aspect of this investment application. It ensures that sensitive financial information, such as transaction histories, is protected from unauthorized access and manipulation. It needs to be required that everything is secure by ensuring that the relationship is protected by limiting the APIs potential to compromise security between the system and the API. This helps to prevent fraud and misuse of the investment system.
* Cryptography- It is important for cryptography to be incorporated in this application. Cryptography can be used to encrypt sensitive information to protect it from unauthorized access. Encrypting the data also ensures that the data was not tampered with during transmission or storage. If encryption isn’t incorporated, it can result in the compromise of customer data.
* Client/Server: A client/server architecture is important in an investment application because it allows for a separation between the front-end and back-end of the system. The client handles the user interface, and it communicates with the back-end to retrieve and update financial data. The client-server architecture supplies ways to handle security issues. Clients are less at risk of being hacked than servers and by keeping sensitive data on servers, the system can be designed secure. The server can incorporate authentication and encryption to ensure that only authorized clients can access information.
* Code Quality: Code quality is important in developing the investment application. The code needs to be developed so that the system is reliable and scalable. The code also needs to be developed using an approach that protects against vulnerabilities that may be in the system. Secure code is more resilient to errors and bugs, which can reduce the risk of system failures. Code quality is important because system failures can have significant financial consequences.

## Manual Review

* DocData.java- It hardcodes the username and password in the code. This means that if an attacker gains access to the source code, they would also have access to the database.
* CRUDController.java : The method should check if the passed parameter is not null or empty before using it.
* No authentication system in place for verifying users

## Static Testing

* 1. bcprov-jdk15on-1.46.jar
     1. 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.
     2. CVE-2016-1000338: In Bouncy Castle JCE Provider version 1.55 and earlier the DSA does not fully validate ASN.1 encoding of signature on verification. It is possible to 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.
     3. CVE-2017-13098 : 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."
     4. 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.
  2. spring-boot-2.2.4.RELEASE.jar
     1. CVE-2022-27772: \*\* UNSUPPORTED WHEN ASSIGNED \*\* spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.
  3. logback-core-1.2.3.jar
     1. CVE-2021-42550 : n 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.
  4. log4j-api-2.12.1.jar
     1. 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 2.13.1CWE-295 Improper Certificate Validation
     2. CVE-2021-44228: Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled. From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.
     3. CVE-2021-44832 : Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are 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.
     4. CVE-2021-45105: Apache Log4j2 versions 2.0-alpha1 through 2.16.0 (excluding 2.12.3 and 2.3.1) did not protect from uncontrolled recursion from self-referential lookups. This allows an attacker with control over Thread Context Map data to cause a denial of service when a crafted string is interpreted. This issue was fixed in Log4j 2.17.0, 2.12.3, and 2.3.1.
  5. snakeyaml-1.25.jar
     1. 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.
     2. CVE-2022-3064:Parsing malicious or large YAML documents can consume excessive amounts of CPU or memory.
     3. 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.
     4. CVE-2022-41854: Those 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. This effect may support a denial of service attack
  6. jackson-databind-2.10.2.jar
     1. 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.
     2. CVE-2022-42004: 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.
  7. tomcat-embed-core-9.0.30.jar
     1. CVE-2019-17569: The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.
     2. CVE-2020-13935: The payload length in a WebSocket frame was not correctly validated in Apache Tomcat 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. Invalid payload lengths could trigger an infinite loop. Multiple requests with invalid payload lengths could lead to a denial of service.CWE-835 Loop with Unreachable Exit Condition ('Infinite Loop')
     3. CVE-2020-1938: When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.
  8. hibernate-validator-6.0.18.Final.jar
     1. 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.
  9. spring-web-5.2.3.RELEASE.jar
     1. CVE-2016-1000027: Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. Depending on how the library is implemented within a product, this issue may or not occur, and authentication may be required. NOTE: the vendor's position is that untrusted data is not an intended use case. The product's behavior will not be changed because some users rely on deserialization of trusted data.

## Mitigation Plan

The first course of action that can be taken in dealing with vulnerabilities and weaknesses in the application is updating dependencies in the report to their current release. Many of the issues in the report relate to using old versions of the dependencies. There also may be a need to consider alternatives that may be safer than the ones chosen. To solve the issue in DocData.java is that we need to remove the hardcoded database username and password from the code, and use variables to store the database credentials, and encrypt the sensitive information. For the issue in CRUDController.java we can add a check and if the parameter is null or empty it will return an error message. Next there needs to be code review to integrate secure coding practices. The review should lead to modifications that help the quality of the code have proper authentication and error catching when needed. Finally, we can implement two-factor authentication systems to help secure the application against unauthorized access.