****

# 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** | **3/19/2023** | **Matthew Tyson** | **Initial Commit** |

## 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

Matthew Tyson

## Interpreting Client Needs

Artemis Financial is looking to modernize their operations and wants to use the most current and effective software security. As a consulting company that develops individualized financial plans for their customers, the confidentiality and integrity of their clients’ financial data are of utmost importance. The RESTful web application programming interface (API) used by Artemis Financial is the primary means of communication with their clients. Therefore, secure communications are essential to ensure the safety of their clients’ data.

Since they deal with financial information, there may be governmental restrictions about secure communications that need to be considered. This could include regulations related to data privacy, financial data protection, and any restrictions on the use of encryption or other secure communication protocols. By understanding and complying with these regulations, the company can help ensure that its software and communications are secure and that it is not in violation of any laws or regulations.

Modernization requirements that need to be considered include the role of open-source libraries and evolving web application technologies.

* Open-source libraries can provide significant benefits for secure communications by enabling developers to leverage existing code that has already been vetted and tested by a community of contributors. This can help reduce development time and costs, while also increasing the quality and security of the final product.
* Evolving web application technologies also need to be considered as part of modernization requirements for secure communications. New web technologies and protocols such as WebRTC, WebSocket, and HTTP/2 can provide significant benefits for real-time communication and data transfer, but they may also introduce new security risks that need to be carefully managed.

## Areas of Security

1. Architecture Review: Analyzing the application architecture is essential to ensure the overall security of the application.
2. Input Validation: Secure input and representation are necessary to prevent injection attacks.
3. APIs: Secure API interactions are necessary to protect against unauthorized access and data breaches.
4. Cryptography: Encryption use and vulnerabilities need to be considered to ensure the confidentiality and integrity of sensitive data.
5. Code Error: Secure code handling is essential to prevent errors that can result in security vulnerabilities.
6. Code Quality: Secure coding practices and patterns are necessary to ensure the overall security of the code.

## Manual Review

1. In the DocData class, there is a SQL injection vulnerability as the method read\_document takes key and value parameters as inputs, but they are not validated or sanitized before being used in the SQL query. An attacker could exploit this vulnerability by crafting a malicious SQL statement and injecting it into the key or value parameter to perform unintended database operations.
2. The DocData class also hardcodes the database credentials in the code, which could be a potential security issue if the code is shared or stored in a version control system.
3. The credentials for the sql server is “root”, “root”. These appear to be default credentials and need to be changed to something more secure.
4. In the CRUDController class, the @RequestParam annotation is used to retrieve the value of the business\_name parameter from the client. However, there is no validation or sanitization of the input, which could lead to injection attacks or other security issues.
5. The customer class has public fields, which could be modified or accessed by unauthorized entities, leading to security vulnerabilities.
6. There are no security measures implemented in the myDateTime class to prevent unauthorized access or modification of the date and time values.

## Static Testing

1. CVE-2013-1624
   1. 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.CWE-310 Cryptographic Issues.
      1. Update beyond 1.6.1
2. CVE-2022-27772
   1. \*\* 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.
      1. This vulnerability was inadvertently fixed as a part of this patch: spring-projects/spring-boot@667ccda
      2. This vulnerability is patched in versions v2.2.11.RELEASE or later.
      3. Setting the java.io.tmpdir system environment variable to a directory that is exclusively owned by the executing user will fix this vulnerability for all operating systems.
3. CVE-2021-42550
   1. 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.
      1. Update logback-classic beyond 1.2.7
4. CVE-2020-9488
   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.
      1. Fixed in Apache Log4j 2.12.3 and 2.13.1
5. CVE-2017-18640
   1. The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.
      1. Fixed In Version: snakeyaml-1.26-1.fc32 snakeyaml-1.26-1.fc31
6. CVE-2020-25649
   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.
      1. Fixed In Version: jackson-databind-2.11.0, jackson-databind-2.10.5.1
7. CVE-2019-17569
   1. 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.
      1. Upgrade tomcat9 packages
8. CVE-2020-10693
   1. 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.
      1. Fixed In Version: hibernate-validator 7.0.0.Alpha2, hibernate-validator 6.1.5.Final, hibernate-validator 6.0.20.Final
9. CVE-2016-1000027
   1. 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.
      1. Do not use Java serialization for external endpoints, in particular not for unauthorized ones. HTTP invoker is not a well-kept secret (or an "oversight") but rather the typical case of how a Spring application would expose serialization endpoints to begin with.
      2. Pivoltal enhanced their documentation for the 4.2.6 and 3.2.17 releases.
10. CVE-2022-22965
    1. A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.
       1. 5.3.x users should upgrade to 5.3.18+, 5.2.x users should upgrade to 5.2.20+. No other steps are necessary.
11. CVE-2021-22060
    1. In Spring Framework versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries. This is a follow-up to CVE-2021-22096 that protects against additional types of input and in more places of the Spring Framework codebase.
       1. 5.3.x users should upgrade to 5.3.14+. 5.2.x users should upgrade to 5.2.19+.
12. CVE-2022-22968
    1. In Spring Framework versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20, and older unsupported versions, the patterns for disallowedFields on a DataBinder are case sensitive which means a field is not effectively protected unless it is listed with both upper and lower case for the first character of the field, including upper and lower case for the first character of all nested fields within the property path.
       1. 5.3.x users should upgrade to 5.3.19+. 5.2.x users should upgrade to 5.2.21+
13. CVE-2022-22950
    1. Spring Framework versions 5.3.0 - 5.3.16 and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial of service condition.
       1. 5.3.x users should upgrade to 5.3.17+. 5.2.x users should upgrade to 5.2.20+

## Mitigation Plan

1. Update all dependencies
2. Change root password in MySQL server.
3. Validate and Sanitize read\_document parameters.
4. Set credentials of MySQL server as an environmental variable instead of hardcoding.
5. Validate and Sanatize the business\_name parameter.
6. Validate input values in myDateTime to prevent invalid data from being set or retrieved.
7. Refactor the customer class field account\_balance to private.