****

# 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** | **11/10/23** | **Anthony Aleman** |  |

## 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

[Insert your name here.]

## Interpreting Client Needs

1. What is the value of secure communications to the company?

Secure communications will be pertinent to the functionality of their RESTful web application. These standards for secure communication can ensure data is not intercepted or tampered with during transmission, maintaining confidentiality and integrity. Without them, the company could leave the door open for financial fraud. Having the edge in such a competitive industry like financial services cannot be achieved when business resources are focused on managing the company’s reputation because of an incident that could have been prevented by having secure and robust communication pipelines. On top of this considerations for legal compliance must also be made as for example in the United States the Gramm-Leach-Bliley Act mandates that financial entities, which provide customers with financial products or services such as loans, investment counsel, or insurance, must disclose their methods of information sharing to their clientele and protect confidential information, (Newman et al, 2023).

1. Does the company make any international transactions?

Based on the brief description of the project, the company’s intention for serving an international customer base is not clear but adding that layer of complexity for security considerations also means following international regulations, restrictions, customs, and protocols.

1. Are there governmental restrictions about secure communications to consider?

There are two other considerations not including the Gramm-Leach-Bliley Act, those being Data sovereignty laws and surveillance laws. Data sovereignty restricts where data is allowed to travel outside of the borders of where it originated, while surveillance laws will affect the monitoring and interception of data, and in the case of the company this includes extremely sensitive financial data.

1. What external threats might be present now and in the immediate future?

Cyberattacks will be an external threat that will be present as they provide a service through the interface of a web application, giving hackers access to tools that can automate and scale their attacks.

1. What are the modernization requirements that you must consider? For example:
2. The role of open-source libraries

While they can accelerate development, they also require careful management to ensure they do not introduce vulnerabilities into the system. Choosing to use open-source libraries attaches to it the possibility of having to help manage the security of these libraries as well as having contact with the maintainers of the project.

1. Evolving web application technologies

Keeping up with the latest frameworks and platforms can improve performance and security but requires continuous learning and updating of systems.

## Areas of Security

Based off the Vulnerability Assessment Process Flow Diagram I identified four areas of security that apply to this project. These include:

Input Validation/Access Control – Users access resources that the institution will provide if requested with the proper validation and authentication

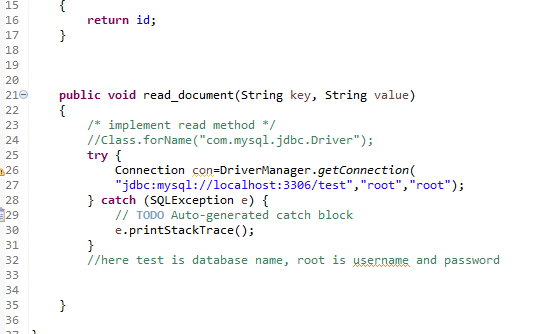
APIs – APIs in general give access to a load of functionality for a myriad of services. In the case of this company API endpoints are the sensitive exposure points for personal information, they must be secure and follow industry standards

Cryptographic – cryptography is essential for modern day secure communications. In the case of the company secure communications cryptographic keys can be a vehicle for authentication of users.

Code Quality – In the following manual review we will focus on the problem that low code quality can cause. Without proper configuration and well throught out solutions, other problems can arise.

## Manual Review

Taking a preliminary look at the code base I would like to focus on the CRUDController class and its functionality. It returns a Doc object which lead me to check that class file. The file shows multiple red flags that would be categorized as security risks. Specifically the file contains hardcoded credentials that expose sensitive information in the source code of the application. Further on, the SQLException object is not handled gracefully.



The other problems include not validating the function input, and not following secure best practices by foregoing SQL query parameterization

## Static Testing

* bcprov-jdk15on-1.46.jar
  + Vulnerability ID: CVE-2016-1000338
  + Description: 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.
  + Attribution: 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. Upgrade version
* hibernate-validator-6.0.18.Final.jar
  + Vulnerability ID: CVE-2020-10693
  + Description: Hibernate's Bean Validation (JSR-380) reference implementation.
  + Attribution: 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. Upgrade version
* jackson-databind-2.10.2.jar
  + Vulnerability ID: CVE-2020-25649
  + Description: General data-binding functionality for Jackson: works on core streaming API
  + Attribution: 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.
* log4j-api-2.12.1.jar
  + Vulnerability ID: CVE-2020-9488
  + Description: The Apache Log4j API
  + Attribution: 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. Upgrade version.
* logback-core-1.2.3.jar
  + Vulnerability ID: CVE-2021-42550
  + Description: logback-core module
  + Attribution: 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. Upgrade version
* snakeyaml-1.25.jar
  + Vulnerability ID: CVE-2022-1471
  + Description: YAML 1.1 parser and emitter for Java
  + Attribution: SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution. We recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization. We recommend upgrading to version 2.0 and beyond.
* spring-boot-2.2.4.RELEASE.jar
  + Vulnerability ID: CVE-2023-20873
  + Description: Spring Boot
  + Attribution: In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application that is deployed to Cloud Foundry could be susceptible to a security bypass. Users of affected versions should apply the following mitigation: 3.0.x users should upgrade to 3.0.6+. 2.7.x users should upgrade to 2.7.11+. Users of older, unsupported versions should upgrade to 3.0.6+ or 2.7.11+.
* spring-boot-starter-web-2.2.4.RELEASE.jar
  + Vulnerability ID: CVE-2023-20873
  + Description: Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container
  + Attribution: In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application that is deployed to Cloud Foundry could be susceptible to a security bypass. Users of affected versions should apply the following mitigation: 3.0.x users should upgrade to 3.0.6+. 2.7.x users should upgrade to 2.7.11+. Users of older, unsupported versions should upgrade to 3.0.6+ or 2.7.11+.
* spring-core-5.2.3.RELEASE.jar
  + Vulnerability ID: CVE-2022-22965
  + Description: Spring Core
  + Attribution: 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.
* spring-web-5.2.3.RELEASE.jar
  + Vulnerability ID: CVE-2016-1000027
  + Description: Spring Web
  + Attribution: 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.
* spring-webmvc-5.2.3.RELEASE.jar
  + Vulnerability ID: CVE-2022-22965
  + Description: Spring Web MVC
  + Attribution: 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.
* tomcat-embed-core-9.0.30.jar
  + Vulnerability ID: CVE-2020-1938
  + Description: Core Tomcat implementation
  + Attribution: 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.
* tomcat-embed-websocket-9.0.30.jar
  + Vulnerability ID: CVE-2020-1938
  + Description: Core Tomcat implementation
  + Attribution: 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.

## Mitigation Plan

* **bcprov-jdk15on-1.46.jar (CVE-2016-1000338)**
  + **Action**: Upgrade to a version later than 1.55 where this vulnerability is resolved.
* **hibernate-validator-6.0.18.Final.jar (CVE-2020-10693)**
  + **Action**: Upgrade to the latest version of Hibernate Validator that has fixed this vulnerability.
* **jackson-databind-2.10.2.jar (CVE-2020-25649)**
  + **Action**: Upgrade Jackson Databind to a version where the XXE vulnerability has been patched.
* **log4j-api-2.12.1.jar (CVE-2020-9488)**
  + **Action**: Upgrade to at least version 2.12.3 or 2.13.1, where the certificate validation issue is fixed.
* **logback-core-1.2.3.jar (CVE-2021-42550)**
  + **Action**: Upgrade to version 1.2.7 or later which addresses the arbitrary code execution via malicious configuration files.
* **snakeyaml-1.25.jar (CVE-2022-1471)**
  + **Action**: Upgrade to version 2.0 or later and use **SafeConstructor** for parsing untrusted YAML content.
* **spring-boot-2.2.4.RELEASE.jar (CVE-2023-20873)**
  + **Action**: Upgrade Spring Boot to version 3.0.6+ for 3.0.x users or 2.7.11+ for 2.7.x users as per the recommendation.
* **spring-boot-starter-web-2.2.4.RELEASE.jar (CVE-2023-20873)**
  + **Action**: Same as above, upgrade Spring Boot to a secure version that is not susceptible to the security bypass issue.
* **spring-core-5.2.3.RELEASE.jar (CVE-2022-22965)**
  + **Action**: If the application is running as a WAR in Tomcat on JDK 9+, upgrade to a non-vulnerable version of Spring Core.
* **spring-web-5.2.3.RELEASE.jar (CVE-2016-1000027)**
  + **Action**: Ensure that the application does not deserialize untrusted data or upgrade to a version of Spring Framework where this vulnerability has been addressed.
* **spring-webmvc-5.2.3.RELEASE.jar (CVE-2022-22965)**
  + **Action**: Similar to the **spring-core** jar action, upgrade to a secure version if running in the vulnerable configuration.
* **tomcat-embed-core-9.0.30.jar (CVE-2020-1938)**
  + **Action**: Disable the AJP Connector if not required or upgrade to Tomcat version 9.0.31 or later where defaults have been hardened.
* **tomcat-embed-websocket-9.0.30.jar (CVE-2020-1938)**
  + **Action**: Same as above, upgrade to a secure version and review the AJP Connector configuration.
* DocData Class Code Review
  + Action: Refactor the DocData class to be more secure and use parameterized SQL queries

References

Newman , J., Fondrie-Teitler, S., Jayanti, A., & Ritchie, A. (2023, June 16). *Gramm-Leach-Bliley Act*. Federal Trade Commission. <https://www.ftc.gov/business-guidance/privacy-security/gramm-leach-bliley-act>