****

# 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** | **03/23/24** | **Joshua Lewis** |  |

## 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

Joshua Lewis

## Interpreting Client Needs

Secure communications for Artemis are essential to daily operations. Not only will they be handling financial information, but they will be handling personal information such as a client’s social security number, address, and date of birth. Given the sensitivity of the information they will be dealing with, it is imperative that all communications, both within and outside of the company, domestic or international, be secure.

Alongside the ethical reasons of secure communications, there are government regulations regarding the safeguarding of customer information. The FTC states that all financial institutions over which the Federal Trade Commission (“FTC” or “Commission”) has jurisdiction shall:

“Base your information security program on a risk assessment that identifies reasonably foreseeable internal and external risks to the security, confidentiality, and integrity of customer information that could result in the unauthorized disclosure, misuse, alteration, destruction, or other compromise of such information, and assesses the sufficiency of any safeguards in place to control these risks.” (U.S. Federal Trade Commission, 2002.)

Because of the sensitivity of the information, Artemis needs to have modern security in place. A designated IT team should be implemented to monitor data flow throughout the system and to / from servers. This team should also routinely check for outdated software / packages / libraries that could potentially pose a risk. This is not only a suggestion but is also required by law by the FTC (U.S. Federal Trade Commission, 2002) All communication should be encrypted, specifically regarding consumer information. All users should also have Muli-Factor authentication in place to help reduce fraudulent login attempts. In addition to the above recommendations, I would also recommend that the company routinely hire a qualified “White Hat Hacker” these cybersecurity specialists use their skills in hacking to help proactively identify vulnerabilities and weaknesses in computer systems (HackerOne, n.d.)

## Areas of Security.

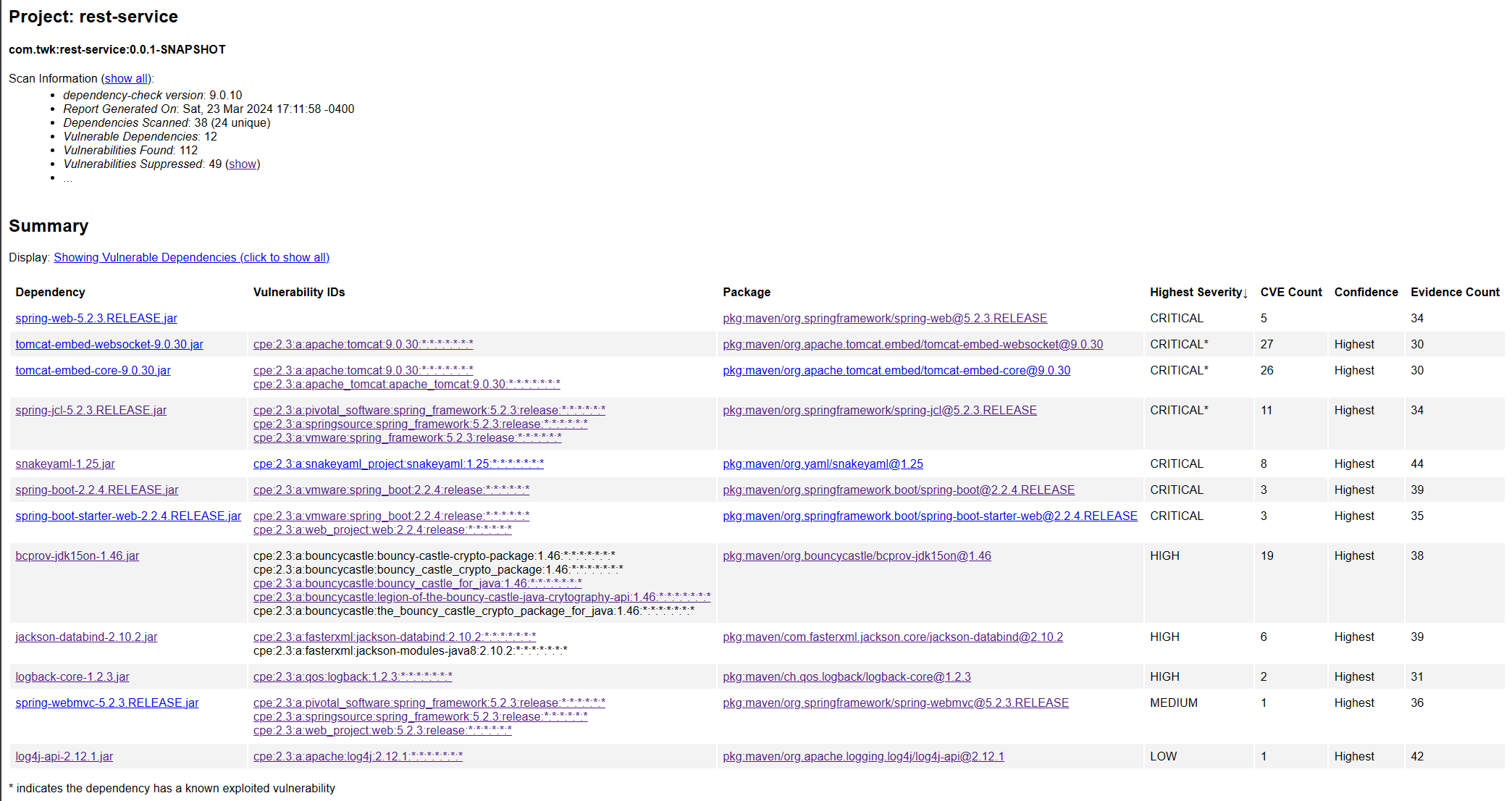
* **Input Validation:** This is important in not only creating/verifying secure login credentials but when used in conjunction with HTTP or query headers (example being @RequestParam used in the CRUDController class) can help prevent security vulnerabilities such as code injection.
* **APIs:** A secure API plays a pivotal role in software / system security. The API is responsible for many aspects regarding security involving authentication, encryption, rate limits and ensures the encryption of data in transit. This part of the software does make calls to the API requesting sensitive information (the company name / client name) and then uses that information to retrieve account information from the SQL database.
* **Cryptography:** As previously stated, all transactions that could potentially reveal secure information about consumers must be encrypted. When transmitting data to/from the API proper protocols for securing and protecting that information must be followed.
* **Client/Server:** Because this is a Web-Based application, clients / users will be connecting to the server to make requests, which means the security of the system lies with not only the API but also on the practices of Secure Coding, Code Quality and Encapsulation. This relates to the previous mention of Input Validation as well.
* **Code error:** Catching errors is important, relating back to Input Validation, an intrusion attempt can be detected by catching and logging failed login attempts, and correctly handling an error can prevent total system failure.
* **Code Quality:** Professionally written, secure code will play a significant role in the application's development. Keeping in mind information that may be available to outside sources (ex. @get methods that are visible in web browsers and could potentially leak sensitive information (Manico & Detlefsen, 2014.)) Having quality code can help prevent the accidental leak of consumer information or details about the system that may pose a threat. Additionally, having easily readable, and modifiable code is essential during a threat response, as the code can be easily read, understood, and modified to resolve the situation quickly.
* **Encapsulation:** The provided code does a decent job of correctly encapsulating most information, however, the field “account\_balance” has the default (package-private) visibility, this means that it can be accessed by other classes within the same package. Another concern is the showInfo() method that directly returns the value of “account\_number” it would be better to use a getter method for controlled access.

## Manual Review

In the review of the code provided, there are a few notable issues present:

* **Hardcoded Credentials:** The DocData class contains database credentials that are hardcoded. This is a risk as anyone with access to the codebase can see these credentials.
* **Potential SQL Injection:** The “read\_document” method in the DocData class could potentially be vulnerable to SQL injection due to the “key” and “value” parameters.
* **Insecure Deserialization of Objects:** The CRUDController class, contains the CRUD method. This method creates a new object from the toString representation of a DocData object. If the created object includes sensitive data, or the input is not validated, this opens the possibility for delivery of a malicious payload.
* **Information Exposure:** The “account\_balance” field in the customer class has (package-private) visibility. This could potentially expose sensitive information.

## Static Testing



* **Vulnerable Dependencies**

## tomcat-embed-core-9.0.30.jar

## **Description:** Core Tomcat implementation.

* + Bcprov-jdk15on-1.46.jar
    - **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.
  + snakeyaml-1.25.jar

## **Description:** YAML 1.1 parser and emitter for Java

* + jackson-databind-2.10.2.jar
    - **Description:** General data-binding functionality for Jackson: works on core streaming API.
  + Logback-core-1.2.3.jar
    - **Description:** logback-core module.
  + Log4j-api-2.12.1.jar
    - **Description:** Apache Log4j API.

## Mitigation Plan

* **tomcat-embed-core-9.0.30.jar:** CVE-2020-1938, this version of Apache Tomcat treats AJP connections with higher trust than similar HTTP connections, this allows the possibility of remote code execution (National Vulnerability Database, 2023a.) This dependency also contains a resource consumption vulnerability due to the HTTP/2 protocol, CVE-2023-44487, (Mitre Corporation, 2023a.) Upgrading to version 9.0.81 is the advised mitigation.
* **Bcprov-jdk15on-1.46.jar:** This library has many issues, notably CVE-2023-33202 and CVE-2016-1000338**.** Regarding CVE-2023-33202, the openssl.PEMParser class contains an issue in which parsing a file that has crafted ASN.1 data causes an OutOfMemoryError, this enables the vulnerability of a denial-of-service attack (National Vulnerability Database, 2023b.) Another security issue lies with CVE-2016-1000338, versions prior to 1.55 do not fully validate ASN.1 encoding of signatures. This allows the possibility of injecting elements into the sequence and the signature still validating. When successfully exploited this can lead to addition or modification of data (Mitre Corporation, 2016.) This version also contains other vulnerabilities (National Vulnerability Database, n.d) and it is advisable to upgrade to the latest version or a minimum of version 1.55 to resolve all security concerns.
* **snakeyaml-1.25.jar: T**his library, due to a flaw in its Constructor class, is vulnerable to arbitrary code execution as described in CVE-2022-1471 (National Vulnerability Database, 2023b.) This version is also vulnerable to Denial-of-Service attacks (DOS) if the parser is running on user-supplied input (Mitre Corporation, 2023b.) The advised solution is to upgrade to version 2.0.
* **jackson-databind-2.10.2.jar:** A vulnerability that could result in denial of service is described in CVE-2020-25649 due to entity expansion (National Vulnerability Database (2020a.) Additionally, Jackson-databind before 2.13.0 allows for a Java StackOverflow exception via a depth of nested objects which can also result in a denial of service (National Vulnerability Database (2020b.) Alongside the previously mentioned vulnerabilities, versions prior to 2.13.4.1 have a known exploit involving resource exhaustion because of a ack of a chick in primitive value deserializers (National Vulnerability Database, 2022.) Upgrading to version 2.13.4.1 will resolve all listed issues.
* **spring-boot-starter-web-2.2.4.RELEASE.jar / spring-boot-2.2.4.RELEASE.jar / spring-web-5.2.3.RELEASE.jar / spring-jcl-5.2.3.RELEASE.jar:** These packages do not directly have vulnerabilities, the main security vulnerabilities regarding these packages lies with Spring Framework. CVE-2022-22965 references that the application may be vulnerable to remote code execution if the application is run on Tomcat as a WAR deployment. (Spring, 2022.) Upgrading to Spring Framework 5.3.18 or 5.2.20 is advised. Versions of Spring Boot older than 2.5.14 and 2.6.14 have a vulnerability in which an application deployed to Cloud Foundry could be susceptible to a security bypass as described in CVE-2023-20873. The suggested measure is to upgrade to Spring Boot 2.5.15+ or Spring Boot 2.6.16+ (Spring, 2023.)
* **Logback-core-1.2.3.jar:** Versions of logback prior to 1.2.7 allow for an attacker to edit configuration files allowing for arbitrary code execution loaded from LDAP servers as described in CVE-2021-42550 (National Vulnerability Database, 2021.) An upgrade to 1.2.7 is advised.
* **Log4j-api-2.12.1.jar:** The current version does not properly validate certificates with SMTP appender. Designated as CVE-2020-9488 This could allow for a man-in-the-middle attack and result in the leaking of log messages sent through the appender (Mitre Corporation, 2020) alongside the logback-core-1.2.3.jar package, this should be upgraded accordingly to resolve security concerns.

The full mitigation plan involves updating all libraries and packages in use to ensure no vulnerabilities are present. Fixing issues related to input validation, encapsulation, and ensuring issues related to accidental data leakage are resolved.

# References:

National Vulnerability Database. (2023a). CVE-2020-1938. National Vulnerability Database. [NVD - CVE-2020-1938 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2020-1938)

Mitre Corporation. (2023a). CVE-2023-44487. Mitre Corporation. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-44487>

National Vulnerability Database. (2023b). CVE-2023-33202. National Vulnerability Database. [NVD - CVE-2023-33202 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2023-33202)

Mitre Corporation. (2016). CVE-2016-1000338. Mitre Corporation. [CVE Record | CVE](https://www.cve.org/CVERecord?id=CVE-2016-1000338)

National Vulnerability Database. (n.d.). Vulnerability Summary for Legion of the Bouncy Castle Java Cryptography API 1.46 [NVD - Results (nist.gov)](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Abouncycastle&cpe_product=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api&cpe_version=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api%3A1.46)

National Vulnerability Database. (2023c). CVE-2022-1471. National Vulnerability Database. [NVD - CVE-2022-1471 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2022-1471)

Mitre Corporation. (2022). CVE-2022-38752. Mitre Corporation. <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-38752>

National Vulnerability Database (2020a). CVE-2020-25649. National Vulnerability Database. [NVD - CVE-2020-25649 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2020-25649)

National Vulnerability Database. (2020b). CVE-2020-36518. National Vulnerability Database. [NVD - CVE-2020-36518 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2020-36518#range-8624378)

National Vulnerability Database. (2022). CVE-2022-42003. National Vulnerability Database. [NVD - CVE-2022-42003 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2022-42003)

Spring. (2022, March 31). CVE-2022-22965: Spring Framework RCE via Data Binding on JDK 9+. Spring Security Advisories. [CVE-2022-22965: Spring Framework RCE via Data Binding on JDK 9+](https://spring.io/security/cve-2022-22965)

Spring. (2023, May 18). CVE-2023-20873: Security Bypass With Wildcard Pattern Matching on Cloud Foundry. Spring Security Advisories. [CVE-2023-20873: Security Bypass With Wildcard Pattern Matching on Cloud Foundry (spring.io)](https://spring.io/security/cve-2023-20873)

National Vulnerability Database. (2021). CVE-2021-42550. National Vulnerability Database. [NVD - CVE-2021-42550 (nist.gov)](https://nvd.nist.gov/vuln/detail/CVE-2021-42550)

Mitre Corporation. (2020). CVE-2020-9488. Mitre Corporation. [CVE Record | CVE](https://www.cve.org/CVERecord?id=CVE-2020-9488)

U.S. Federal Trade Commission. [(2002, May 23)*Standards for Safeguarding Customer Information.* In *Electronic Code of Federal Regulations Title 16*](https://www.ecfr.gov/current/title-16/chapter-I/subchapter-C/part-314)

HackerOne**.** (n.d.) *What Is a White Hat Hacker? The Ethical Side of Hacking.* [White Hat Hackers: Techniques, Tools, and How to Become One (hackerone.com)](https://www.hackerone.com/knowledge-center/white-hat-hacker)

Manico, J., & Detlefsen, A. (2014, September). *Iron-clad Java*. O’Reilly Online Learning. <https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/ch01.html#ch01lev1sec15>

<suppress>

<notes><![CDATA[file name: spring-web-5.2.3.RELEASE.jar]]></notes>

<packageUrl regex="true">^pkg:maven/org\.springframework/spring\-web@.\*$</packageUrl>

<cve>CVE-2016-1000027</cve>

</suppress>

<suppress>

<notes><![CDATA[file name: spring-web-5.2.3.RELEASE.jar]]></notes>

<packageUrl regex="true">^pkg:maven/org\.springframework/spring\-web@.\*$</packageUrl>

<vulnerabilityName>CVE-2024-22243</vulnerabilityName>

</suppress>

<suppress>

<notes><![CDATA[file name: spring-web-5.2.3.RELEASE.jar]]></notes>

<packageUrl regex="true">^pkg:maven/org\.springframework/spring\-web@.\*$</packageUrl>

<vulnerabilityName>CVE-2024-22262</vulnerabilityName>

</suppress>

<suppress>

<notes><![CDATA[file name: spring-web-5.2.3.RELEASE.jar]]></notes>

<packageUrl regex="true">^pkg:maven/org\.springframework/spring\-web@.\*$</packageUrl>

<cve>CVE-2020-5421</cve>

</suppress>

<suppress>

<notes><![CDATA[file name: spring-web-5.2.3.RELEASE.jar]]></notes>

<packageUrl regex="true">^pkg:maven/org\.springframework/spring\-web@.\*$</packageUrl>

<cve>CVE-2022-22950</cve>

</suppress>

<suppress>

<notes><![CDATA[file name: spring-web-5.2.3.RELEASE.jar]]></notes>

<packageUrl regex="true">^pkg:maven/org\.springframework/spring\-web@.\*$</packageUrl>

<cve>CVE-2023-20861</cve>

</suppress>

<suppress>

<notes><![CDATA[file name: spring-web-5.2.3.RELEASE.jar]]></notes>

<packageUrl regex="true">^pkg:maven/org\.springframework/spring\-web@.\*$</packageUrl>

<cve>CVE-2023-20863</cve>

</suppress>

<suppress>

<notes><![CDATA[file name: spring-web-5.2.3.RELEASE.jar]]></notes>

<packageUrl regex="true">^pkg:maven/org\.springframework/spring\-web@.\*$</packageUrl>

<cve>CVE-2022-22968</cve>

</suppress>

<suppress>

<notes><![CDATA[file name: spring-web-5.2.3.RELEASE.jar]]></notes>

<packageUrl regex="true">^pkg:maven/org\.springframework/spring\-web@.\*$</packageUrl>

<cve>CVE-2022-22970</cve>

</suppress>