****

# 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** | **3-14-2023** | **James Soto** |  |

## 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

James Soto

## Interpreting Client Needs

1. What is the value of secure communications to the company?
   1. The value of secure communications to Artemis Financial is that they want to have safeguards in place to protect the sensitive data and information within their organization. Artemis Financial is seeking our expertise to help them secure the data from unwanted access, data breaches, or brute force attacks. By having safeguards in place they increase the confidence of the organization and of the clients.
2. Does the company make any international transactions?
   1. Artemis Financial offers their services on a web-based platform, meaning that they could make transactions or offer services to clients/customers from any part of the planet.
3. Are there governmental restrictions about secure communications to consider?
   1. Depending on the country and state that Artemis Financials’ offices are located, the government has security measures in place to ensure that the data and systems are properly secured.
4. What external threats might be present now and in the immediate future?
   1. Secure communications face a variety of external threats both now and in the future. Some of the most significant threats are the following: Cyberattacks: With the increasing use of digital communication channels, cyberattacks have become a major threat to secure communications. These attacks can come in various forms, including malware, phishing scams, and distributed denial-of-service (DDoS) attacks. Hacking: Hackers can breach secure communication channels to steal sensitive information, such as login credentials, financial data, and personal information. This can lead to identity theft, financial loss, and other serious consequences. Interception and eavesdropping: Communication channels can also be intercepted or monitored by unauthorized parties, allowing them to access sensitive information. This can be particularly problematic for businesses and governments that handle confidential data. Physical theft or loss of devices: Devices that store or transmit sensitive data can be lost or stolen, creating a security risk. This can occur in a variety of settings, including offices, airports, and public transportation. Social engineering: Social engineering attacks involve tricking individuals into divulging sensitive information or performing actions that compromise security. These attacks can be difficult to detect and can target anyone, regardless of their technical expertise. As technology continues to evolve, new threats to secure communications will emerge.
5. What are the modernization requirements that you must consider? For example:
   1. The role of open-source libraries:

Open source libraries play a critical role in secure communications by providing developers with reliable and tested code that can be used to build secure communication applications. Open source libraries are community-developed software libraries that are publicly available and licensed in a way that allows users to freely use, modify, and distribute the code. Some examples of open source libraries used in secure communications include OpenSSL, which provides encryption and decryption functionality for secure communication protocols such as HTTPS and SSH, and libSodium, which provides cryptographic primitives for secure communication protocols such as TLS and VPNs. Some of the key benefits of using an open source library include:

* Security: Open source libraries are often more secure than proprietary libraries because they are developed by a community of developers who can review and test the code. This means that security vulnerabilities can be detected and fixed more quickly, reducing the risk of security breaches.
* Quality: Open source libraries are typically high-quality and reliable because they are developed and maintained by a community of developers who are passionate about the project. This means that the code is often well-documented, well-tested, and well-maintained.
* Cost: Open source libraries are often free or low-cost, making them accessible to developers and organizations of all sizes. This can significantly reduce the cost of developing secure communication applications.
* Customization: Open source libraries are often highly customizable, allowing developers to tailor the code to their specific needs. This can help ensure that the resulting application is optimized for security and performance.
* Collaboration: Open source libraries promote collaboration among developers, allowing them to share code, ideas, and best practices. This can lead to better security outcomes and more innovative solutions.
  1. Evolving web application technologies

Modernizing secure communications in the evolving web application technology sector requires a comprehensive approach that addresses a range of technical, organizational, and operational requirements. Some of the ways to accomplish this feat is by using the following tools.

• Encryption: Encryption is essential to protecting the confidentiality of communications. Modernizing secure communications requires implementing strong encryption algorithms and protocols to ensure that data remains secure in transit and at rest.

• Multi-factor authentication: Multi-factor authentication is an effective way to protect against unauthorized access to communications.

• Network security: Modernizing secure communications requires implementing advanced network security technologies, such as firewalls, intrusion detection and prevention systems (IDPS), and security information and event management (SIEM) systems. These technologies help protect against cyberattacks and other security threats.

• Security awareness training: Employees and users are often the weakest link in secure communications.

• Mobile device security: Mobile devices are increasingly being used for communication, and they are vulnerable to security threats.

• Compliance with regulations and standards: Many industries and organizations are subject to regulations and standards related to secure communications.

## Areas of Security

By utilizing the RESTful API our interface would securely communicate and exchange information between two software programs over the internet. The REST API would use a GET request to retrieve a record, a POST request to create the record, a PUT request to update the record and a DELETE request to delete one. Since we are utilizing an API, we would need strictly validate all the input that would interact with our program to ensure the security of our system, like multiple login attempts. After reviewing the code, we also need to ensure the code is free of errors, including any syntax errors, logic errors, compilation errors, and runtime errors. Code quality ensures we are writing code that is secure, free of any deficiencies, is maintainable, readable and testable. We would also need to implement proper cryptography, which the process of protecting sensitive information from unauthorized access when it is at rest or in transit by rendering it unreadable without a key. Leveraging encryption, cryptography helps users secure data transmission over networks, ensuring that only individuals with designated keys can access encrypted data.

## Manual Review

Manually inspecting the code I came across the following vulnerabilities:

* In CRUDController.java, business name is sent as a request parameter in the CRUD method, and is passed to DocData, exposing the data and information to an outside attack.
* In DocData.java , in the read document method, line 26 had a variable that was unused. To test, the test uses the database name, and root is username and password. This leaves the system extremely vulnerable since the root is using default logins, such as username and password.
* The requests are not validated, leaving the entire system vulnerable to an outside attack.
* There is no input validation in place to validate and authenticate any input that is interacting with the system.
* When sharing sensitive information, it is recommended to use HTTPS, which in this case is not utilized.
* Performing the JUnit test, the coverage for rest-service is at 14.6% overall, meaning the code needs to be improved drastically to ensure the security of the entire system, including employees and clients.

Graphical user interface, text

Description automatically generated

## Static Testing

1. The names or vulnerability codes of the known vulnerabilities.
2. A brief description and recommended solutions that are found in the dependency-check report.
3. Attribution (if any) that documents how this vulnerability has been identified or how it was documented in the past.

Text

Description automatically generated

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

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.

* + **CVE-2016-1000338**
  + **CVE-2016-1000342**
  + **CVE-2016-1000343**
  + **CVE-2016-1000344**
  + **CVE-2016-1000352**
  + **CVE-2016-1000341**
  + **CVE-2016-1000345**
  + **CVE-2017-13098**
  + **CVE-2020-15522**
  + **CVE-2020-0187**
  + **CVE-2016-1000339**
  + **CVE-2020-26939**
  + **CVE-2015-7940**
  + **CVE-2018-5382**
  + **CVE-2013-1624**
  + **CVE-2016-1000346**
  + **CVE-2016-6644**

17 vulnerabilities found on software version 1.46, recommended to update to latest software version.

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

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.

CWE-20 Improper Input Validation

* + **CVE-2020-10693**

1 vulnerability found on software version 6.0.18, You can pass user input as an expression variable by unwrapping the context to HibernateConstraintValidatorContext. Or by upgrading to the newest software version.

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

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.

CWE-611 Improper Restriction of XML External Entity Reference ('XXE')

* + **CVE-2020-25649**
  + **CVE-2020-36518**
  + **CVE-2022-42003**
  + **CVE-2022-42004**

4 vulnerabilities found on software version 2.10.2,update to the latest software per NetApp

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

The Apache Log4j API

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

* + **CVE-2020-9488**

1 vulnerability found on software version 2.12.1, update the software to the latest software per Apache.org. Previous versions can set the system property mail.smtp.ssl.checkserveridentity to true to globally enable hostname verification for SMTPS connections.

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

logback-core module

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.

CWE-502 Deserialization of Untrusted Data

* + CVE-2021-42550

1 vulnerability found on software version 1.2.3, update the software to the latest version, restrict the write access to the logback configuration file (logback.xml) to trusted personnel

* **snakeyaml-1.25.jar**

YAML 1.1 parser and emitter for Java

* + **CVE-2022-1471**
  + **CVE-2017-18640**
  + **CVE-2022-25857**
  + **CVE-2022-38749**
  + **CVE-2022-38751**
  + **CVE-2022-38752**
  + **CVE-2022-41854**
  + **CVE-2022-38750**

8 vulnerabilities found on software version 1.25,update the software to the latest version per Snakeyaml

* **spring-boot-2.2.4.RELEASE.jar**

Spring Boot

* + **CVE-2022-27772**

This vulnerability exists because File.mkdir returns false when it fails to create a directory, it does not throw an exception. As such, the following race condition exists:

1 vulnerability found on software version 2.2.4, update software to the newest software version or use the following workaround: 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.

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

Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container

* + **CVE-2022-27772**

1 vulnerability found on software version 2.2.4, update software to the newest software version or use the following workaround: 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.

* **spring-core-5.2.3.RELEASE.jar**

Spring Core

* + **CVE-2022-22965 – CISA KNOWN EXPLOITED VULNERABILITY**
  + **CVE-2021-22118**
  + **CVE-2020-5421**
  + **CVE-2022-22950**
  + **CVE-2022-22971**
  + **CVE-2022-22968**
  + **CVE-2022-22970**
  + **CVE-2021-22060**
  + **CVE-2021-22096**

9 vulnerabilities found in software version 5.2.3,update to the newest software version or use the following workaround: Block both incoming and outgoing connections between the system and the internet.

* **spring-web-5.2.3.RELEASE.jar**

Spring Web

* + **CVE-2016-1000027**
  + **CVE-2022-22965 – CISA KNOWN EXPLOITED VULNERABILITY**
  + **CVE-2021-22118**
  + **CVE-2020-5421**
  + **CVE-2022-22950**
  + **CVE-2022-22971**
  + **CVE-2022-22968**
  + **CVE-2022-22970**
  + **CVE-2021-22060**
  + **CVE-2021-22096**

10 vulnerabilities found in software version 5.2.3,update to the newest software version or use the following workaround: Block both incoming and outgoing connections between the system and the internet.

* **spring-webmvc-5.2.3.RELEASE.jar**

Spring Web MVC

* + **CVE-2022-22965 – CISA KNOWN EXPLOITED VULNERABILITY**
  + **CVE-2021-22118**
  + **CVE-2020-5421**
  + **CVE-2022-22950**
  + **CVE-2022-22971**
  + **CVE-2022-22968**
  + **CVE-2022-22970**
  + **CVE-2021-22060**
  + **CVE-2021-22096**

9 vulnerabilities found in software version 5.2.3,update to the newest software version or use the following workaround: Block both incoming and outgoing connections between the system and the internet.

* **tomcat-embed-core-9.0.30.jar**
* Core Tomcat implementation
  + **CVE-2020-1938 - CISA KNOWN EXPLOITED VULNERABILITY**
  + **CVE-2020-11996**
  + **CVE-2020-13934**
  + **CVE-2020-13935**
  + **CVE-2020-17527**
  + **CVE-2021-25122**
  + **CVE-2021-41079**
  + **CVE-2022-29885**
  + **CVE-2022-42252**
  + **CVE-2020-9484**
  + **CVE-2021-25329**
  + **CVE-2021-30640**
  + **CVE-2022-34305**
  + **CVE-2021-24122**
  + **CVE-2021-33037**
  + **CVE-2019-17569**
  + **CVE-2020-1935**
  + **CVE-2020-13943**
  + **CVE-2021-43980**

19 vulnerabilities found in software version 9.0.30, update to newest software version available.

* **tomcat-embed-websocket-9.0.30.jar**
* Core Tomcat implementation
  + **CVE-2020-1938 - CISA KNOWN EXPLOITED VULNERABILITY**
  + **CVE-2020-8022**
  + **CVE-2020-11996**
  + **CVE-2020-13934**
  + **CVE-2020-13935**
  + **CVE-2020-17527**
  + **CVE-2021-25122**
  + **CVE-2021-41079**
  + **CVE-2022-29885**
  + **CVE-2022-42252**
  + **CVE-2020-9484**
  + **CVE-2021-25329**
  + **CVE-2021-30640**
  + **CVE-2020-34305**
  + **CVE-2021-24122**
  + **CVE-2021-33037**
  + **CVE-2019-17569**
  + **CVE-200-1935**
  + **CVE-2020-13943**
  + **CVE-2021-43980**

20 vulnerabilities found in software version 9.0.30, update to newest software version available.

## Mitigation Plan

To help against brute force attacks the company should implement stronger password parameters with 2 factor authentication for accounts that hold extremely valuable information. The company needs to make improvements to the code inside of the program to help the quality of the code as well as implementing proper authentication and error catching protocols to ensure that all vulnerabilities are addressed. All the software and servers being utilized by the system need to be updated to the latest software versions, which include patches for any vulnerabilities found. The company needs to utilize HTTPS to ensure that all company and customer information/data is secure. It is also recommended that the company move the request parameters to the header. It is recommended that the code validate certificates both in the application and on the webserver.

Sources

Manico, J., & Detlefsen, A. (n.d.). Iron-clad Java. O'Reilly Online Learning. Retrieved March 15, 2023, from https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/?sso\_link=yes&sso\_link\_from=SNHU

Owasp secure coding practices-quick reference guide. OWASP Secure Coding Practices-Quick Reference Guide | OWASP Foundation. (n.d.). Retrieved March 15, 2023, from https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/

Secure Coding Guidelines for Java SE. Secure coding guidelines for java SE. (n.d.). Retrieved March 15, 2023, from https://www.oracle.com/java/technologies/javase/seccodeguide.html

NATIONAL VULNERABILITY DATABASE. NVD. (n.d.). Retrieved March 15, 2023, from https://nvd.nist.gov/vuln/search

Search CVE list. CVE. (n.d.). Retrieved March 15, 2023, from https://cve.mitre.org/cve/search\_cve\_list.html

Summary data security laws: State government. National Conference of State Legislatures. (n.d.). Retrieved March 15, 2023, from https://www.ncsl.org/technology-and-communication/data-security-laws-state-government