****

# 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** | **07/16/2023** | **Philip Shaw** |  |

## 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

Philip Shaw

## Interpreting Client Needs

As an organization based in the finance industry, Artemis Financial is operating a system which is at a greater risk for security threats. Having greater consequences for cyber attacks combined with an increased likelihood of attempted attacks, the company understands that it’s imperative to employ the strongest available security measures in order to ensure their system’s integrity for their organization and for their clients. There are numerous potential threats from a wide range of potential antagonists, including rival firms, government offices, independent intruders and spontaneous software malfunctions. Additionally, Artemis also understands that improved security can’t come at the cost of convenience to their clientele—due to the risk of clients going elsewhere for similar services.

Because of globalization within the financial sector, Artemis is also required to understand and comply with governmental communication regulations and international conventions for financial firms. The Federal Trade Commission’s *Gramm-Leach-Bliley Act*—GLBA—of 1999 is one that directly affects the operations of Artemis Financial, as it pertains to the mergers, resource sharing and general transparency of financial institutions (FTC, 2023). Additionally, it’s important to adhere to conventions such as the International Accounting Standards Board’s *International Financial Reporting Standards* (IFRS, 2023). These compliance protocols should be built into Artemis’ system rather than having to be manually implemented.

Artemis’ decision to modernize their system will allow their organization to capitalize on current security measures able to assist and protect the people and processes in their network and allow their operations to remain compliant to regulations and standard conventions. Some of these measures include comprehensive input validation, multi-factor authentication and advanced encryption methods. Also, the adoption of strong open-source security and encryption libraries are proving to be incredibly beneficial for preventing intrusion. These include “Bouncy Castle,” “Google Keyczar” and the “OWASP AppSensor” (Manico & Detlefsen, 2015) (pp. 266-268). Ultimately, financial technology—Fintech—has disrupted banking and finance, creating pressure on organizations to provide “as a service” options that require continuous integration and delivery. Artemis is looking to leverage the available tools to ensure their own organization remains competitive.

## Areas of Security

**Input Validation:** this is a vital first point of contact—and potential trouble—for Artemis’ operations and needs to be thoroughly considered throughout the SDLC process. This is especially a concern regarding authentication. It’s common practice for multi-factor authentication to be used within financial applications.

**API:** Artemis Financial uses a restful API to facilitate their services to clients. Ensuring that this API remains secure at each interface is very important due to the exposure of risk in their industry.

**Cryptography:** Artemis Financial operates in an incredibly sensitive domain, where the use of robust encryption is not only necessary for the security of their operations, but a legal requirement by the Federal Trade Commission (FTC, 2023).

**Client/Server:** A system is only as strong as its weakest link. Artemis’ entire system needs to follow the same security measures and permission restraints within its global architecture as for each of its modules. This needs to remain congruent for both server-side and client-side functionality.

**Code Error Handling:** unhandled exceptions could lead to potential vulnerabilities, which is why special attention should be paid to discovering edge cases that could lead to these exceptions. Integration of continuous automated testing could prove useful in this regard.

**Code Quality Practices**: the practice of writing consistent, reusable code will not only make it easier to diagnose potential issues, but also ensure that new developers brought on board will be able to become familiarized with Artemis’ coding conventions without creating errors that could compromise the integrity of the system.

**Encapsulation:** Since Artemis’ RESTful API interfaces with users and other applications, it’s important to make sure that inputs and outputs follow the principle of least privilege, abstracting all data that isn’t relevant for each expected use case.

## Manual Review

For the sake of efficiency, this review summarizes the security concerns of each Java class within the application.

**CRUD:** this class appears to be a standard CRUD class for instantiating content in the rest API. The attributes present are private and there doesn’t seem to be any security issues within the code here. However, it’s difficult to tell if the if the constructor parameters are directly associated with table data—in which case query parameterization would be needed.

**CRUDController:** REST API for handling the application’s CRUD operations. This short class seems to be mostly secure and displays the use of query parameterization.

**customer:** this class is used for creating and updating object’s representative of real-world customer entities. Regarding the vulnerabilities here:

* It may be a potential vulnerability to declare the “account\_balance” integer as a global/public value.
* It’s more common for account balances to represented as decimal types. While this doesn’t seem to be a security issue now, it could become an issue if it’s refactored later.
* Allowing a public method to directly set the “account\_balance” attribute is potentially risky.

**DocData:** allows a connection between the application and its SQL database. This class has the most concerning security vulnerabilities, including:

* Within the try block, the login information is hardcoded into the DriverManager.getConnection() method. This is unsafe. There should also be some form of input validation.
* Additionally, the username and password are displayed within the comments.
* “root” and “admin” are common superuser login names and passwords, often left over from pre-deployment development and testing. Changing this to something more secure would also increase the database’s security integrity.
* It’s a more conventional practice to use a logging framework rather than printing the stack trace—line 30.

**Greeting:** This is another basic class used for constructing, setting and getting greeting objects within the application. Major security issues don’t appear to be present here.

**GreetingController:** This REST API is used to retrieve a Greeting message mapped to a corresponding suffix. The use of query parameterization within the GET request method makes this process more secure.

**myDateTime:** This class is used to determine a user’s current local date and time within the application. Some potential concerns could arise from the lack of abstraction of the three attributes here.

**RestServiceApplication:** this is a main class for executing the application. There doesn't seem to be any major security concerns within this part of the code.

## Static Testing

The OWASP Foundation’s dependency check for Maven was used to discover any known vulnerabilities in Artemis’ application. This is a screenshot of the check that was performed:



The table below displays the dependency vulnerabilities that were found in the static check. Displayed are: names of each dependent library, a brief description of that library, discovered vulnerability codes, a brief description of the most prevalent vulnerability, and the given threat level of vulnerabilities found in each library.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dependency** | **Description** | **Vulnerability Codes** | **Highest Ranked Vulnerabilities (CVSS V3)** | **Threat Level** |
| bcprov-jdk15on-1.46.jar | Library of cryptography algorithms tailored to Java. | [**CVE-2016-1000338**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338), | [**CVE-2016-1000338**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338), Encoding vulnerability (affects versions 1.55 and earlier). | High |
| hibernate-validator-6.0.18.Final.jar | Used to facilitate application constraint validation. | [**CVE-2020-10693**](https://nvd.nist.gov/vuln/detail/CVE-2020-10693) | XML vulnerability. | Medium/Moderate |
| jackson-databind-2.10.2.jar | Facilitates data-binding functionality for the Jackson library. | [**CVE-2020-25649**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649), [**CVE-2020-36518**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-36518), [**CVE-2021-46877**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-46877), [**CVE-2022-42003**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42003), [**CVE-2022-42004**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42004), | [**CVE-2020-25649**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649), Mail protocol vulnerability. | High |
| log4j-api-2.12.1.jar | Used for generating log messages. | [**CVE-2020-9488**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488) | Mail protocol connection vulnerability. | Low |
| logback-core-1.2.3.jar | Also used for generating log messages/files. | [**CVE-2021-42550**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42550) | Configuration file(s) vulnerability. | Medium/Moderate |
| snakeyaml-1.25.jar | Used for parsing files and creating associative “emissions.” | [**CVE-2022-1471**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-1471), [**CVE-2017-18640**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640), [**CVE-2022-25857**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-25857), [**CVE-2022-38749**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38749), [**CVE-2022-38751**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38751), [**CVE-2022-38752**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38752), [**CVE-2022-41854**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-41854), [**CVE-2022-38750**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-38750) | [**CVE-2022-1471**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-1471), Constructor class vulnerability. | Critical |
| spring-boot-2.2.4.RELEASE.jar | Facilitates project creation for the Spring Framework by curating third-party libraries. | [**CVE-2022-27772**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772), [**CVE-2023-20883**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20883) | [**CVE-2022-27772**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772), Temporary directory vulnerability (only affects versions older than 2.2.11). | High |
| spring-boot-starter-web-2.2.4.RELEASE.jar | Used for creating web applications with Spring Boot. | [**CVE-2022-27772**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772), [**CVE-2023-20883**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20883) | [**CVE-2022-27772**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-27772), Temporary directory vulnerability (only affects versions older than 2.2.11). | High |
| spring-core-5.2.3.RELEASE.jar | Provides fundamental components for the Spring Framework. | [**CVE-2022-22965**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965), [**CVE-2021-22118**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118), [**CVE-2020-5421**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421), [**CVE-2022-22950**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950), [**CVE-2022-22971**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971), [**CVE-2023-20861**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20861), [**CVE-2023-20863**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20863), [**CVE-2022-22968**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968), [**CVE-2022-22970**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970), [**CVE-2021-22060**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060), [**CVE-2021-22096**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096) | [**CVE-2022-22965**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965), Vulnerable to a remote code execution exploit. | Critical |
| spring-web-5.2.3.RELEASE.jar | Provides components for creating web applications on the Spring Framework. | [**CVE-2016-1000027**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000027), [**CVE-2022-22965**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965), [**CVE-2021-22118**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118), [**CVE-2020-5421**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421), [**CVE-2022-22950**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950), [**CVE-2022-22971**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971), [**CVE-2023-20861**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20861), [**CVE-2023-20863**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20863), [**CVE-2022-22968**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968),[**CVE-2022-22970**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970), [**CVE-2021-22060**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060), [**CVE-2021-22096**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096) | [**CVE-2016-1000027**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000027), Vulnerable to a remote code execution exploit. | Critical |
| spring-webmvc-5.2.3.RELEASE.jar | Default “model-view-controller” application functionality for the Spring Framework. | [**CVE-2022-22965**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965), [**CVE-2021-22118**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118), [**CVE-2020-5421**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421), [**CVE-2022-22950**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22950), [**CVE-2022-22971**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22971), [**CVE-2023-20861**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20861), [**CVE-2023-20863**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-20863), [**CVE-2022-22968**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22968), [**CVE-2022-22970**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22970), [**CVE-2021-22060**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22060), [**CVE-2021-22096**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096) | [**CVE-2022-22965**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965), Vulnerable to a remote code execution exploit. | Critical |
| tomcat-embed-core-9.0.30.jar | Provides core functionality for Apache Tomcat. | [**CVE-2020-1938**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938), [**CVE-2020-11996**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996), [**CVE-2020-13934**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934), [**CVE-2020-13935**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935), [**CVE-2020-17527**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527), [**CVE-2021-25122**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122), [**CVE-2021-41079**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079), [**CVE-2022-29885**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885), [**CVE-2022-42252**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42252), [**CVE-2020-9484**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484), [**CVE-2021-25329**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329), [**CVE-2021-30640**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640), [**CVE-2022-34305**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305), [**CVE-2021-24122**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122), [**CVE-2021-33037**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037), [**CVE-2019-17569**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569), [**CVE-2020-1935**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935), [**CVE-2020-13943**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943), [**CVE-2023-28708**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-28708), [**CVE-2021-43980**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-43980) | [**CVE-2020-1938**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938), Apache Jserv Protocol (AJP) vulnerability. | Critical |
| tomcat-embed-websocket-9.0.30.jar | Provides more core functionality for Apache Tomcat and WebSocket implementation. | [**CVE-2020-1938**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938), [**CVE-2020-8022**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-8022), [**CVE-2020-11996**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996), [**CVE-2020-13934**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934), [**CVE-2020-13935**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935), [**CVE-2020-17527**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527), [**CVE-2021-25122**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122), [**CVE-2021-41079**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079), [**CVE-2022-29885**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-29885), [**CVE-2022-42252**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-42252), [**CVE-2020-9484**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484), [**CVE-2021-25329**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329), [**CVE-2021-30640**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640), [**CVE-2022-34305**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-34305), [**CVE-2021-24122**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122), [**CVE-2021-33037**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037), [**CVE-2019-17569**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569), [**CVE-2020-1935**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935), [**CVE-2020-13943**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943), [**CVE-2023-28708**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-28708), [**CVE-2021-43980**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-43980) | [**CVE-2020-1938**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938), Apache Jserv Protocol (AJP) vulnerability. | Critical |

## Mitigation Plan

Within Artemis’ application, the most urgent course of action is to update the deprecated dependencies and to refactor code with potential vulnerabilities. Implementing the suggestions listed in the manual review—part 3—will be beneficial for mitigating risks due to the discovered vulnerabilities. Additionally, adhering to the fundamentals of a methodology such as DevSecOps will ensure that the application capitalizes on the benefits of current trends within its own development pipeline—such as automated testing, continuous integration of new security features and an improved ability to monitor processes within the application.

References:

FTC (2023). Gramm-Leach-Bliley Act. Federal Trade Commission. <https://www.ftc.gov/legal-library/browse/statutes/gramm-leach-bliley-act>

IFRS (2023). Who we are. International Financial Reporting Standards Foundation. <https://www.ifrs.org/about-us/who-we-are/>

Manico, Jim. & Detlefsen, August. (2015). Iron-Clad Java: Building Secure Web Applications. McGraw Hill.