# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11-16-2024** | **Sean Jetté** | **First Release** |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In this report, identify your security vulnerability findings and recommend 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 include images or supporting materials. If you include them, make certain to insert them in 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

Sean Jetté

**1. Interpreting Client Needs**

Determine your client’s needs and potential threats and attacks associated with the company’s application and software security requirements. Consider the following questions regarding how companies protect against external threats based on the scenario information:

* What is the value of secure communications to the company?
* Are there any international transactions that the company produces?
* Are there governmental restrictions on secure communications to consider?
* What external threats might be present now and in the immediate future?
* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

For a financial consulting organization such as Artemis Financial, secure communications would be held to the highest standards. Looking at the scenario, their mission statement is to provide individualized financial plans for customers that include savings, retirement, investments, and insurance. In the realm of cybersecurity, this would invoke the importance of confidentiality, integrity, and availability and as such, a lapse in their security would spell severe consequences for the survival of the organization and their ability to maintain a secure platform.

According to the given scenario, there is no specific information provided that indicates Artemis Financial conducts any international transactions. With that, financial institutions typically would have the means for international transactions through methods such as wire transfers or utilizing other forms of digital transfer methodologies. This indicates that within their security framework, a robust cybersecurity posture embedded in their software is paramount; especially with added complexities through data privacy laws in Europe or even the California Consumer Privacy Act (CCPA). Moreover, there are other regulatory requirements that might impact their international security stance specifically concerning countries that are currently facing economic sanctions being imposed and the implementation of regulatory requirements to ensure that certain banks are barred (Davidson, 2022).

There is an ongoing effort concerning economic sanctions imposed on countries like Russia, in which their ability to utilize certain communications or exchange messaging through SWIFT is being blocked. According to Davidson (2022), “The Society for Worldwide Interbank Financial Telecommunication, or SWIFT, is used by large financial institutions and governments to exchange information about rapid money transfers such as those traversing the Federal Reserve's Fedwire service.” Additionally, there may be international regulatory requirements concerning encryption standards for sensitive data handling; especially when dealing with financial transfers or even disclosing customer data to third parties. As such, Artemis Financial would be subjected to all international and federal regulatory requirements and would need to implement those requirements for secure communications within their banking software.

With respect to the software security aspects, there are a number of immediate external threats that Artemis Financial could face. Starting specifically with their software not being properly maintained and updated in accordance with updated guidance or newly disclosed CVEs. Even if their software currently meets the standards for cybersecurity practices, newly discovered CVEs can pose a risk to the integrity of the organization’s software and thus affect their customers and the organization’s reputation. Furthermore, following secure coding guidelines for their source code “can help to protect against hostile, misbehaving, or unsafe code” (Oracle, 2023). While this is ultimately an internal issue, with unsafe coding practices or lack of security in mind, Artemis Financial could be subjected to hacking attempts that try to exploit vulnerabilities within the code base or databases through the use of XSS, injection attacks, API vulnerabilities that do not impose proper encryption standards, or best standard practices within Java applications like query parameterization, which “is a computer programming technique to build a *dynamic SQL statement* safely by “binding” untrusted data into placeholders within a query” (Manico & Detlefsen, 2015). Within the immediate future, threat vectors towards the organization could manifest in more sophisticated attacks such as an insider threat, social engineering attempts, and zero-day vulnerabilities.

For Artemis Financial’s software’s modernization requirements, there are many aspects to consider that are essential to remaining competitive within the financial consulting world. These modernization requirements are not just for security purposes, but also to improve customer experiences and meet the needs of regulatory requirements. A particular challenge within the financial world is legacy banking systems, which “most financial experts also cited it as the main barrier to unlocking the full potential of Big Data analytics” which have several drawbacks to include data integrity and data quality issues (Vahromovs, 2021). In addition, within the web application development world, open-source adoption of programming web application languages, cloud technologies, and API-based service architectures are some driving factors in modernization (Lebed, 2024).

A function that is important to businesses is efficiency and cost; especially if it can speed up time-to-market with pre-built solutions as open-source libraries can provide, but this comes with a cost of potential security vulnerabilities and not fully meeting internal and regulatory standards. According to the latest Open-Source Financial Services report (2023), respondents within the financial industry have overwhelmingly agreed that open-source is valuable to the future of financial services, getting more value year-after-year, and that the adoption rate of open-source libraries are greater than 50% of respondents (Rosenbaum, 2024).

Within the trends of evolving web application technologies, modernizing API integration is foundational to success, often with third-party services, such as Zelle. Additionally, because of the need for API integration, choosing RESTful API technologies due to flexibility in data retrieval and management would prove to be essential for the complexity of a financial institution. These all come with the cost of maintaining up-to-date source code and implementing continuous monitoring for additional CVEs, zero-days, and other threat vectors that will be crucial for continual security requirements, even if they add some additional costs.

**2. Areas of Security**

Refer to the vulnerability assessment process flow diagram. Identify which areas of security apply to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

Utilizing the vulnerability assessment process flow diagram through the architectural review and considering Artemis Financial is utilizing a RESTful web API, the most applicable vulnerabilities are input validation, APIs, code errors, code quality, and encapsulation.

Starting with input validation as one of the first lines of defense within web applications, ensuring that all inputs are sanitized accordingly to prevent adversaries from gaining a foothold within your program is paramount for Artemis Financial. This defense should incorporate best practices suggested by OWASP (2024) “due to data from all potentially sources should be subjected to input validation, including not only internet-facing web clients but also backend feeds.” These guidelines, while not fully encompassing all requirements to prevent XSS, SQL injection or other attacks, help for applications to implement best practices for preventing malformed inputs that could lead to errant behavior and or introduce injection vulnerabilities.

API security, as the essential function for communication through the web application between the customers and the organization’s servers, would be another critical area for Artemis Financial to focus on. Through the usage of secure transmission protocols, such as HTTPS, this will help to prevent man-in-the-middle attacks through the interception of data as the contents of the packets in transmission will be encrypted by default. While adversaries have become more adept at modifying HTTP methods through the use of programs like *Burp Suite*, applying restrictions to such HTTP methods, such as GET, POST, PUT, while also rejecting requests that do not match the allowlist with HTTP response code 405 *Method not allowed* will help (OWASP, 2024). These methodologies can aid in the web application’s ability to prevent adversaries from tampering with the applications and will also protect endpoints with authentication.

With respect to code errors, the Java web application utilizes the Spring Framework and has some built-in error handling functionality but there would still be some proper implementation required to prevent adversaries from taking advantage. Essentially, “improper handling of errors can introduce a variety of security problems for a web site” (OWASP, 2024) and that while generation of error conditions within web applications is not outside the norm, these messages often reveal to adversaries some potential details through enumeration that could lead to flaws that may be exploited. Good error handling mechanisms should be configured by the organization that detail the necessary information required for the end user without revealing unnecessary internal details (OWASP, 2024).

Coding quality, while inherent in its name, helps ensure that the application is ultimately following best coding practices and standards. When developing the web application, any inherent flaws within the source code could be utilized for exploitation. These types of practices are important for defining clear security requirements for any code changes that comply with industry standards, create testing for potential SQL injection vulnerabilities, XSS flaws, or any other methodology for penetration, and implementing manual code reviews through changes in CVEs, code logic, and ensuring best security practices are always followed (Verdi, 2024).

Last but not least, encapsulation is an important function within the development process for data integrity. As an organization that deals with customer’s finances, implementing encapsulation throughout the codebase will help to protect valuable customer information. By making class variables private, implementing getter and setter methods, the program’s design will help mitigate any unintended consequences of data manipulation or exploitation.

**3. Manual Review**

Continue working through the vulnerability assessment process flow diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

In the CRUDController class, the @RequestParam annotation receives the business\_name parameter from the request. However, the code does not validate or sanitize this input, which could potentially allow injection attacks or malicious input.

The CRUDController uses the DocData class and calls its toString() method to generate a response. Since the toString() implementation is not provided, there’s a risk that it could expose sensitive or unintended data (e.g., object fields or raw data). If toString() includes sensitive information, it might inadvertently leak data through the API response.

The Customer class includes a field named account\_balance, which is not declared as private. This allows the field to be accessed and modified directly from outside the class. Sensitive fields like account\_balance should be declared as private and accessed through getter and setter methods to maintain data integrity and control access.

The showInfo() method in the Customer class is public but should likely have restricted access to prevent unintended external use. Making this method private or protected could be more appropriate. Also, the account\_number field should also remain private to ensure it cannot be modified directly from outside the class. This prevents unauthorized changes and maintains data integrity.

The read\_document method in the DocData class contains hardcoded database credentials:

* *Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");*

Using "root" for both the username and password is a poor cybersecurity practice, akin to using default credentials like "admin" and "password123". This approach makes the application vulnerable to credential leaks and unauthorized access. Credentials should be managed securely using environment variables or a credential management system. If detailed error messages are exposed from the read\_document method, sensitive information about the database or application structure might be leaked to attackers.

The GreetingController class contains the greeting() method, where the @RequestParam annotation receives the name parameter from user input. However, the input is not validated or sanitized before being used in the String.format method to generate a response. This lack of validation can lead to injection attacks (e.g., SQL injection, command injection) or Cross-Site Scripting (XSS) attacks. This oversight allows unauthorized users to invoke the API with malicious input, potentially disrupting the application's functionality.

The myDateTime class contains the setMyDateTime() method, which accepts parameters for seconds, minutes, and hours. However, the method does not validate these inputs to ensure they fall within acceptable ranges. For example, seconds and minutes should be between 0–59, and hours should be between 0–23. Without validation, the method could accept negative values or out-of-range inputs, potentially leading to undefined behavior or errors in the application.

The application configuration does not appear to enable *HTTPS*. The application.properties file specifies the server's port (8081) but lacks configurations for *SSL/TLS* encryption. Without *HTTPS*, sensitive data transmitted between the client and server, such as credentials and API requests, is vulnerable to interception by attackers. *HTTPS* should be configured in *application.properties* using SSL/TLS certificates.

**4. Static Testing**

Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from the dependency-check report. Include the following items:

* The names or vulnerability codes of the known vulnerabilities
* A brief description and recommended solutions provided by the dependency-check report
* Any attribution that documents how this vulnerability has been identified or documented previously

|  |
| --- |
| **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.  **Identifiers:**   * cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:\*:\*:\*:\*:\*:\*:\* * cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\* * [cpe:2.3:a:bouncycastle:bouncy\_castle\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*](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%3Abouncy_castle_for_java&cpe_version=cpe%3A%2F%3Abouncycastle%3Abouncy_castle_for_java%3A1.46) * [cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*](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) * cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*   **Package:** [pkg:maven/org.bouncycastle/bcprov-jdk15on@1.46](https://ossindex.sonatype.org/component/pkg:maven/org.bouncycastle/bcprov-jdk15on@1.46?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** HIGH  **CVE Count:** 22 |
| **hibernate-validator-6.0.18.Final.jar**  **Description:** Hibernate's Bean Validation (JSR-380) reference implementation.  **Identifiers:**   * [cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aredhat&cpe_product=cpe%3A%2F%3Aredhat%3Ahibernate_validator&cpe_version=cpe%3A%2F%3Aredhat%3Ahibernate_validator%3A6.0.18)   **Package:** [pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final](https://ossindex.sonatype.org/component/pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** MEDIUM  **CVE Count:** 2 |
| **jackson-databind-2.10.2.jar**  **Description:** General data-binding functionality for Jackson: works on core streaming API  **Identifiers:**   * [cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Afasterxml&cpe_product=cpe%3A%2F%3Afasterxml%3Ajackson-databind&cpe_version=cpe%3A%2F%3Afasterxml%3Ajackson-databind%3A2.10.2) * cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\*   **Package**: [pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2](https://ossindex.sonatype.org/component/pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** HIGH  **CVE Count:** 6 |
| **log4j-api-2.12.1.jar**  **Description:** The Apache Log4j API  **Identifiers:**   * [cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Alog4j&cpe_version=cpe%3A%2F%3Aapache%3Alog4j%3A2.12.1)  (*Confidence*:Highest)   **Package:** [pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1](https://ossindex.sonatype.org/component/pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** LOW  **CVE Count:** 1 |
| **logback-core-1.2.3.jar**  **Description:** logback-core module  **Identifiers:**   * [cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aqos&cpe_product=cpe%3A%2F%3Aqos%3Alogback&cpe_version=cpe%3A%2F%3Aqos%3Alogback%3A1.2.3)   **Package:** [pkg:maven/ch.qos.logback/logback-core@1.2.3](https://ossindex.sonatype.org/component/pkg:maven/ch.qos.logback/logback-core@1.2.3?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** HIGH  **CVE Count:** 2 |
| **snakeyaml-1.25.jar**  **Description:** YAML 1.1 parser and emitter for Java  **Identifiers:**   * [cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Asnakeyaml_project&cpe_product=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml&cpe_version=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml%3A1.25)   **Package:** [pkg:maven/org.yaml/snakeyaml@1.25](https://ossindex.sonatype.org/component/pkg:maven/org.yaml/snakeyaml@1.25?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** CRITICAL  **CVE Count:** 8 |
| **spring-boot-2.2.4.RELEASE.jar**  **Description:** Spring Boot  **Identifiers:**   * [cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_boot&cpe_version=cpe%3A%2F%3Avmware%3Aspring_boot%3A2.2.4)   **Package:** [pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE](https://ossindex.sonatype.org/component/pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** CRITICAL  **CVE Count:** 3 |
| **spring-boot-starter-web-2.2.4.RELEASE.jar**  **Description:** Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container  **Identifiers:**   * [cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_boot&cpe_version=cpe%3A%2F%3Avmware%3Aspring_boot%3A2.2.4) * [cpe:2.3:a:web\_project:web:2.2.4:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aweb_project&cpe_product=cpe%3A%2F%3Aweb_project%3Aweb&cpe_version=cpe%3A%2F%3Aweb_project%3Aweb%3A2.2.4)   **Package:** [pkg:maven/org.springframework.boot/spring-boot-starter-web@2.2.4.RELEASE](https://ossindex.sonatype.org/component/pkg:maven/org.springframework.boot/spring-boot-starter-web@2.2.4.RELEASE?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** CRITICAL  **CVE Count:** 3 |
| **spring-core-5.2.3.RELEASE.jar**  **Description:** Spring Core  **Identifiers:**   * [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3) * [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) * [cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_framework&cpe_version=cpe%3A%2F%3Avmware%3Aspring_framework%3A5.2.3)   **Package:** [pkg:maven/org.springframework/spring-core@5.2.3.RELEASE](https://ossindex.sonatype.org/component/pkg:maven/org.springframework/spring-core@5.2.3.RELEASE?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** CRITICAL\*  **CVE Count:** 11 |
| **spring-expression-5.2.3.RELEASE.jar**  **Description:** Spring Expression Language (SpEL)  **Identifiers:**   * [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3) * [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) * [cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_framework&cpe_version=cpe%3A%2F%3Avmware%3Aspring_framework%3A5.2.3)   **Package:** [pkg:maven/org.springframework/spring-expression@5.2.3.RELEASE](https://ossindex.sonatype.org/component/pkg:maven/org.springframework/spring-expression@5.2.3.RELEASE?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** CRITICAL\*  **CVE Count:** 12 |
| **spring-web-5.2.3.RELEASE.jar**  **Description:** Spring Web  **Identifiers:**   * [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3) * [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) * [cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_framework&cpe_version=cpe%3A%2F%3Avmware%3Aspring_framework%3A5.2.3) * [cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aweb_project&cpe_product=cpe%3A%2F%3Aweb_project%3Aweb&cpe_version=cpe%3A%2F%3Aweb_project%3Aweb%3A5.2.3)   **Package:** [pkg:maven/org.springframework/spring-web@5.2.3.RELEASE](https://ossindex.sonatype.org/component/pkg:maven/org.springframework/spring-web@5.2.3.RELEASE?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** CRITICAL\*  **CVE Count:** 15 |
| **spring-webmvc-5.2.3.RELEASE.jar**  **Description:** Spring Web MVC  **Identifiers:**   * [cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3) * [cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aspringsource&cpe_product=cpe%3A%2F%3Aspringsource%3Aspring_framework&cpe_version=cpe%3A%2F%3Aspringsource%3Aspring_framework%3A5.2.3) * [cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_framework&cpe_version=cpe%3A%2F%3Avmware%3Aspring_framework%3A5.2.3) * [cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aweb_project&cpe_product=cpe%3A%2F%3Aweb_project%3Aweb&cpe_version=cpe%3A%2F%3Aweb_project%3Aweb%3A5.2.3)   **Package:** [pkg:maven/org.springframework/spring-webmvc@5.2.3.RELEASE](https://ossindex.sonatype.org/component/pkg:maven/org.springframework/spring-webmvc@5.2.3.RELEASE?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** CRITICAL\*  **CVE Count:** 12 |
| **tomcat-embed-core-9.0.30.jar**  **Description:** Core Tomcat implementation  **Identifiers:**   * [cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Atomcat&cpe_version=cpe%3A%2F%3Aapache%3Atomcat%3A9.0.30) * [cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_tomcat&cpe_product=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat&cpe_version=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat%3A9.0.30)   **Package:** [pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30](https://ossindex.sonatype.org/component/pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** CRITICAL\*  **CVE Count:** 26 |
| **tomcat-embed-websocket-9.0.30.jar**  **Description:** Core Tomcat implementation  **Identifiers:**   * [cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Atomcat&cpe_version=cpe%3A%2F%3Aapache%3Atomcat%3A9.0.30) * [cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache_tomcat&cpe_product=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat&cpe_version=cpe%3A%2F%3Aapache_tomcat%3Aapache_tomcat%3A9.0.30)   **Package:** [pkg:maven/org.apache.tomcat.embed/tomcat-embed-websocket@9.0.30](https://ossindex.sonatype.org/component/pkg:maven/org.apache.tomcat.embed/tomcat-embed-websocket@9.0.30?utm_source=dependency-check&utm_medium=integration&utm_content=11.1.0)  **Highest Severity:** CRITICAL\*  **CVE Count:** 27 |

**5. Mitigation Plan**

Interpret the results from the manual review and static testing report. Then identify the steps to mitigate the identified security vulnerabilities for Artemis Financial’s software application.

Based on the identified security vulnerabilities and findings from both the manual code review and static dependency analysis, the following mitigation methodologies are recommended for Artemis Financial to improve the overall security of the software application. Implementing input validation or sanitization in methods that accept user input is vital for mitigating SQL injection and cross site scripting (XSS) threats. The adopting of an allowlist approach, where only explicitly defined inputs are accepted, can significantly reduce the potential of attacks from hackers/threat vectors. Additionally, leveraging input sanitization libraries to prevent SQL and XSS injection attacks would be highly recommended overall. Another important security improvement is implementing code encapsulation by declaring sensitive fields as private and using getter/setter methods to control access, thus restricting unauthorized access to sensitive information.

In terms of credential management, hardcoding credentials within the source code is a poor practice that should be avoided. Simplistic default credentials can pose a significant risk, as they are vulnerable to brute-force attacks. Instead, Artemis Financial should use environment variables, encrypted configuration files, or credential management services to securely store and retrieve sensitive information. To prevent sensitive information leakage through error messages, secure error handling should be implemented to ensure messages remain generic and do not reveal critical implementation details. Detailed error logs should be kept internally for debugging purposes and should never be exposed to end-users.

The absence of SSL/TLS configuration in the application.properties file is a critical issue that must be addressed before deployment. To mitigate the risk of man-in-the-middle (MITM) attacks, HTTPS should be enforced for all client-server communications, and any HTTP traffic should be redirected to HTTPS. Another major issue identified is the outdated dependencies, such as spring-core, bcprov-jdk150n, and snakeyaml, which must be updated to their latest secure versions. Consistent monitoring of dependency updates, CVEs, and zero-day vulnerabilities is vital, and tools like Maven’s dependency-check plugin should be utilized for this purpose.

Additional vulnerabilities, such as the lack of range validation in the myDateTime class, must also be addressed. Validating date and time inputs against acceptable ranges using Java's built-in date/time libraries is an effective solution. Finally, maintaining up-to-date versions of the Spring framework and Java ensures that the application remains secure and benefits from the latest advancements in web application development. Staying current with technology and fostering a culture of continuous training and proactive monitoring are essential to the long-term success of Artemis Financial’s software security.

References:

Davidson, C. (2022, March 3). *SWIFT’s role in global banking*. Federal Reserve Bank of Atlanta. https://www.atlantafed.org/economy-matters/banking-and-finance/2022/03/03/swift-role-in-global-banking

Lebed, D. (2024, February 23). *Web Application Architecture: The Latest Guide 2024*. Peiko. https://peiko.space/blog/article/web-application-architecture

Manico, J., & Detlefsen, A. (2015). *Iron-Clad Java: Building secure web Applications (Oracle Press)* (Version 1.0) [Webpage]. McGraw-Hill Education. https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/?sso\_link=yes&sso\_link\_from=SNHU

Oracle. (2023, May). *Secure Coding Guidelines for Java SE*. https://www.oracle.com/java/technologies/javase/seccodeguide.html

OWASP. (2024). *Input Validation - OWASP Cheat Sheet Series*. OWASP Foundation. https://cheatsheetseries.owasp.org/cheatsheets/Input\_Validation\_Cheat\_Sheet.html

Rosenbaum, A. (2024, February 12). The 2023 State of Open Source in Financial Services. *Red Hat Blog*. https://www.redhat.com/en/blog/state-open-source-financial-services

Vahromovs, V. (2021, November 22). Legacy systems in banking: the major barrier for digital transformation. *FinTech Futures*. https://www.fintechfutures.com/2021/11/legacy-systems-in-banking-the-major-barrier-for-digital-transformation/

Verdi, S. (2024, October). *Best practices and tools for secure code reviews*. Graphite.dev. https://graphite.dev/guides/code-reviews-for-security-best-practices