

**Artemis Financial Vulnerability Assessment Report**

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| **1.0** | **9/17/2023** | **Jacob Senior** | Conducted vulnerability assessment |

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

Jacob Senior

* Interpreting Client Needs

Secure communications should be highly valued in Artemis Financial's software application. The application involves communicating information regarding savings, retirement, investment, and insurance information between parties. Secure communications should be valued in order to prevent unauthorized access to user data. In managing users' sensitive personal data, we should consider how to be compliant with government regulations regarding data privacy. In addition to federal regulations, some states in the U.S. have their own data privacy regulations (*Which States*, 2023). Our consideration extends to international regulations as well, for example, compliance with the General Data Protection Regulation (GDPR) (Wolford, n.d.).

The described application has many external threats to be concerned about. For example, the application could be vulnerable to a variety of cyber attacks such as injection attacks, denial of services, data breaches, or even social engineering attacks. Another "threat" to the application in the future is changing laws on data privacy. Failure to comply with changing data privacy regulations could result in large fines (Wolford, n.d.). This is also an example of a modernization requirement we should consider in the application. As technology continues to change over time, we should consider how these changes could impact our software. For example, dependencies used in our application could release patches for existing vulnerabilities, or even be exposed to new vulnerabilities. Thus, it is important that security continues to be considered even after the deployment of the application.

* Areas of Security

The areas of security relevant to this software application are input validation, secure API interactions, cryptography, and code quality.

Since this application uses HTTP requests as well as databases, we should practice input validation to prevent injection attacks. Improper handling of input validation could lead to SQL injections, cross-site scripting, or remote code execution.

Secure API interactions are important since the application implements a RESTful web API. Any APIs used in this application could be vulnerable to their own set of exploits. We should make ourselves aware of exploits within these APIs to reduce their impact on our applications.

Since we need secure communication within this application, we should make use of encryption and be aware of related vulnerabilities. Cryptography should be used both when transmitting and storing data in our application.

Consequently, we should make use of secure coding practices/patterns to defend against vulnerabilities. Good code quality is essential for implementing secure code throughout this application. This includes using coding practices to validate input, interact with APIs, and implement cryptography.

* Manual Review

- CRUDController.java

The CRUD function does not properly validate the name parameter. This could make our code vulnerable to injection attacks. We should instead sanitize the name parameter to remove dangerous characters, validate the length, and check if the value is null.

- customer.java

The account\_balance variable is missing an access modifier. The default access modifier for a variable is package-protected, meaning it can be accessed by members within the same package. To make this more secure, it should be changed to private.

The deposit amount in the deposit function is not validated. We should check that the deposit amount is non-negative and should account for integer overflow.

- DocData.java

Although the functionality is not yet implemented, the read\_document function could be vulnerable to injection attacks. We should validate the key and value inputs, sanitizing the parameters to remove dangerous characters, validating their length, and checking if they are null. We should also use a parameterized query when querying the database.

Establishing a database connection in plain text in the read\_document function could expose the URL, username, and password for the database. Instead, we could store this information in an encrypted configuration file and use that to establish a connection.

The database connection is not closed properly in the read\_document function. After querying the database, we should close the connection we opened.

- GreetingController.java

The greeting function does not properly validate the name parameter. This could allow for injection attacks. We should validate the name input before creating a greeting, sanitizing the parameter to remove dangerous characters, validate the length, and check if the value is null.

* Static Testing

- 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.

- hibernate-validator-6.0.18.Final.jar

Hibernate's Bean Validation (JSR-380) reference implementation.

- jackson-databind-2.10.2.jar

General data-binding functionality for Jackson: works on core streaming API.

- log4j-api-2.12.1.jar

The Apache Log4j API.

- logback-core-1.2.3.jar

logback-core module.

- snakeyaml-1.25.jar

YAML 1.1 parser and emitter for Java.

- spring-boot-2.2.4.RELEASE.jar

Spring Boot.

- 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.

- spring-core-5.2.3.RELEASE.jar

Spring Core.

- spring-web-5.2.3.RELEASE.jar

Spring Web.

- spring-webmvc-5.2.3.RELEASE.jar

Spring Web MVC.

- tomcat-embed-core-9.0.30.jar

Core Tomcat implementation.

- tomcat-embed-websocket-9.0.30.jar

Core Tomcat implementation.

* Mitigation Plan

The code vulnerabilities identified in the manual review section have their solutions given after the descriptions of the vulnerabilities.

The dependency vulnerabilities can be solved by upgrading the identified dependencies to the most recent versions. If for any reason we are not able to update a given dependency, some vulnerabilities can be prevented by not using certain functions or implementing proper safeguards. The best solution, however, is going to be to update the dependencies listed above.

As highlighted by this vulnerability assessment report, we should be more cautious during the future development of this application. Some of the major areas we should be concerned with are input validation, secure API interactions, cryptography, and secure coding practices/patterns. Some strategies we can use to catch vulnerabilities in the future are manual code review and static testing. As technology evolves, this application could be exposed to more vulnerabilities in the future. We should frequently update software dependencies and search for vulnerabilities to protect this application from attacks in the future.

* Resources

*Which States Have Consumer Data Privacy Laws.* (2023, September 7). Bloomberg Law. Retrieved September 17, 2023, from <https://pro.bloomberglaw.com/brief/state-privacy-legislation-tracker/#:~:text=Currently%2C%20there%20are%2011%20states,data%20privacy%20laws%20in%20place>.

Wolford, B. (n.d.). *What is GDPR, the EU's new data protection law.* GDPR.eu. <https://gdpr.eu/what-is-gdpr/>