# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/21/2024** | **Dorrian Robinson** |  |

## Client



## Developer

Dorrian Robinson

**1. Interpreting Client Needs**

Artemis Financial handles financial information and as such security is a top priority. It holds accounts, identifiable materials, plans for investment, and more. To keep the data secure it’s necessary to implement common methods like encryption and secure authentication while also following regulations. International transactions will be part of their business model. This means not only following local regulations but also international ones. This can apply to both financial regulations and data protection policies. For example, GDPR is a EU regulation while PCI-DSS is a global standard for card security.

External threats with financial platforms are endless, but many are targeted specifically at the organizations database(s) or their customers directly. Examples include SQL Injection, XSS, and commonly phishing customers and employees.

When it comes to open-source libraries, or any, it’s important to stay up to date and ensure that there’s no vulnerabilities and if there is to handle them. For technologies you should be using automated tools for security. An example of this is Vanta for regulations and compliance or Snyk for finding vulnerabilities in code.

**2. Areas of Security**

1. Input Validation
   1. Input validation can solve several security risks, but most importantly addresses can address issues such as SQL Injection and XSS. Preventing malicious queries to retrieve, modify, or delete from the database and injecting code are some of the most critical issues a company can have happen. A loss of data would require comparing backups to current data, validating and a host of other issues. Allowing changes in what customers are doing and seeing could be used to redirect, modify, and or trick customers.
2. Cryptography
   1. Cryptography is essential for protecting data both being sent and stored. It can prevent, or at least slow, damages done when data is accessed by unauthorized parties. This also falls into the category of the following regulations, which as stated above, have both global and local requirements. This also addresses secure communication which leads into Secure APIs.
3. Secure APIs
   1. Secure APIs are essential when dealing with multiple systems. When communicating with other systems like SendGrid, airbrake, or stripe for example it’s important that what’s being sent and received is both secure, and valid. Secure APIs help with data integrity and confidentiality, authentication, and can prevent tampering.
4. Code Error
   1. In conjunction with Input validation and Secure APIs exceptions should be handled and logged. Examples include unauthorized requests and values that have failed validation
5. Client Server
   1. The system is being used to communicate between a server and end users implying the above concerns.

**3. Manual Review**

1. Hardcoded Credentials: read\_document has a hardcoded jdbc connection
2. Insecure RNG: Java’s random util is predictable using an algorithm
3. Unvalidated Input: The CrudController is missing validation
4. HTTP: RestServiceApplication is using HTTP instead of HTTPS
5. DocData has no handling and is using the console instead of logging
6. Sensitive Data: When handling the exceptions there’s no validation or trimming of the message being stored. This could contain credentials or identifiable information
7. Authorization: Currently there is no verification of user permission in the CRUDController
8. Unvalidated Input: Greeting in the GreetingsController isn’t properly validated.

**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:

1. bcprov-jdk15on-1.46.jar

* Vulnerability ID:
  + cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:bouncy\_castle:crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:legion\_of\_the\_bouncy\_castle:java-cryptography-api:1.46:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:bouncy\_castle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: HIGH
* Solution: Update to the latest version of the Bouncy Castle library.

1. hibernate-validator-6.0.18.Final.jar

* Vulnerability ID:
  + cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: MEDIUM
* Solution: Update to the latest version of Hibernate Validator.

1. jackson-core-2.10.2.jar

* Vulnerability ID:
  + cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:json-java:project:json:2.10.2:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: HIGH
* Solution: Update to the latest version of Jackson Core.

1. jackson-databind-2.10.2.jar

* Vulnerability ID:
  + cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: HIGH
* Solution: Update to the latest version of Jackson Databind.

1. log4j-api-2.12.1.jar

* Vulnerability ID:
  + cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: LOW
* Solution: Update to the latest version of Log4j API.

1. logback-core-1.2.3.jar

* Vulnerability ID:
  + cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: HIGH
* Solution: Update to the latest version of Logback Core.

1. snakeyaml-1.25.jar

* Vulnerability ID:
  + cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: CRITICAL
* Solution: Update to the latest version of SnakeYAML.

1. spring-boot-2.2.4.RELEASE.jar

* Vulnerability ID:
  + cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: CRITICAL
* Solution: Update to the latest version of Spring Boot.

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

* Vulnerability ID:
  + cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:web\_project:web:2.2.4:release:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: CRITICAL
* Solution: Update to the latest version of Spring Boot Starter Web.

1. spring-core-5.2.3.RELEASE.jar

* Vulnerability ID:
  + cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: CRITICAL
* Solution: Update to the latest version of Spring Core.

1. spring-web-5.2.3.RELEASE.jar

* Vulnerability ID:
  + cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: CRITICAL
* Solution: Update to the latest version of Spring Web.

1. spring-webmvc-5.2.3.RELEASE.jar

* Vulnerability ID:
  + cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: CRITICAL
* Solution: Update to the latest version of Spring Web MVC.

1. tomcat-embed-core-9.0.30.jar

* Vulnerability ID:
  + cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: CRITICAL
* Solution: Update to the latest version of Tomcat Embed Core.

1. tomcat-embed-websocket-9.0.30.jar

* Vulnerability ID:
  + cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*:\*
* Severity: CRITICAL
* Solution: Update to the latest version of Tomcat Embed WebSocket.

**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.

To fix the issues from the static testing, we should update all packages to their latest version, or alternatively the suggested one from the report but those have some vulnerabilities as well fixed in newer versions.

* **Hardcoded Credentials:**
  + Replace with environment variables
  + Test the new implementation.
* **Insecure RNG:**
  + Replace with SecureRandom.
  + Code review and testing.
* **Unvalidated Input in CrudController:**
  + Sanitize inputs.
  + Implement and test input validation.
* **Unvalidated Input in GreetingsController:**
  + Sanitize inputs.
  + Implement and test input validation.
* **HTTP Usage in RestServiceApplication:**
  + Enable HTTPS.
  + SSL certificates.
  + Redirect HTTP to HTTPS.
  + Code review and testing
* **Lack of Proper Logging in DocData:**
  + Implement logging framework.
  + Ensure secure log management.
  + Implement proper error handling.
  + Code review and testing
* **Sensitive Data in Exception Handling:**
  + Sanitize exception messages.
  + Implement exception handling framework.
  + Implement data masking or trim values. Code review and testing
* **Lack of Authorization in CRUDController:**
  + Implement RBAC.
  + Use security frameworks.
  + Implement auditing and logging.
  + Code review and testing

A screenshot of a computer

Description automatically generatedStatic Testing Output