****

# CS 305 Project One

**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** | **5/18/2022** | **Brandon Hobbs** | **First Release** |

## Client



## Developer

Brandon Hobbs

## 1. Interpreting Client Needs

*Determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Consider the following 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 about secure communications to consider?*
* *What external threats might be present now and in the immediate future?*
* *What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?*

Because Artemis is a financial services company, security will be a high priority as financial rewards can be a prime target for hackers. Moreover, customers will also value security, to not lose their assets, of course, but also privacy. It could be embarrassing for certain patrons to have their assets, or lack of, disclosed in a public forum.

As a financial firm transfers to and from foreign nations are possible. This could expose Artemis to foreign laws – even those surrounding Personally Identifiable Information (PII). Even further, differing countries may have varied rules concerning records retention, contract law, or even format rules, i.e., digital records vs. physical. This may also mean that there are applicable laws concerning secure communications.

For example, in the USA the Electronic Communications Privacy Act (ECPA) allows the government “…access [to] digital communications such as email, social media messages, information on public cloud databases, and more with a subpoena.” There are also the CFAA, SOX, and Gramm-Leach-Bliley Act that may come into play. Outside council should be used to verify full compliance and applicability of any legal requirement.

Artemis having an online presence brings risk of different cybercrimes. These might be as low-level and annoying but not as harmful as a denial-of-service (DOS) attack or as harmful and severe as a breach leading to loss.

A breach could lead to financial loss for consumers or the company which then could damage Artemis’ reputation leading to loss of business. Currently, it would take ~15 years to crack RSA-512 encryption – NIST suggests no less than 112 bits (RSA-2048) – but given the changing power of processing (GPU crackers) even RSA-2048 is thought to fail by 2030. High level encryption should be applied to all databases, even though considered as not confidential, as just having small amount of PII might make a phishing attack more fruitful or expose Artemis to regulatory scrutiny.

Artemis needs to be aware of any implication of open source code. For one, it may bring overhead with certain open-source licenses, such as any *copyleft* licenses (GNU GPL and others) that force reciprocity. This openness could expose Artemis to scrutiny and possibly vulnerability if a weakness was identified in either code. Moreover, any library, open source or purchased needs to be maintained and updated as frequently as possible. For example, the recent Log4J issues have been traced to over 93-million exploits already and Apache had to release 4 separate patches to fix all the Common Vulnerabilities and Exposures (CVEs) identified.

## 2. Areas of Security

*Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.*

This being a RESTful API I would focus on Input Validation, Secure API Interactions, Code Errors, and Encapsulation.

Input Validation is important as you need to “sanitize” user inputs, even if from a trusted user. A malformed input, on purpose or not, could cause errant behavior. Securing the API goes one step farther in ensuring the connection (and possibly the user) is trustworthy. There is some code checking built into Spring but this error handling needs to be generic and not provide any signature that a hacker could use to understand more about the code structure. And finally, because we are using methods, the methods should use encapsulation to protect the sensitive data from unwanted changes, i.e., set variables to private and use Get/Set functions.

Cryptography should also be considered as all data flows should be encrypted. As discussed before RSA-2048 strength should be considered for all databases and data transmission. Encryption-type and capabilities should be checked in each operating country as law vary on cryptography.

## 3. Manual Review

*Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.*

As part of the static testing it was noticed that the POM.xml needed to be updated to a more recent release of Maven (5.1.X vs 7.1.X). Using old static cases will cause some CVEs to be missed.

Within the code, some methods have Get/Set functions available and, for example, the Greeting class sets its parameters private. GreetingController, however, does not use the available get/set functions.

User input to the API is not sanitized. For example, within the number method the value from the array is passed via string splices: *String message = "Element in the given index is :: "+myArray[id]. id* is the raw variable obtained from the user. This is also true in the CRUDcontroller’s CRUD method. These could lead to injection

.

All methods taking user input, such as the greeting method, need to be sanitized and checked for length, i.e., to prevent buffer overruns.

The customer class has the account\_balance as a public member. This should be set to private and have Get/Set methods used to control manipulation.

The myDateTime class has a few poorly coded methods, e.g., setMyDateTime, as they are incomplete. The DocData class also has human readable username and password hardcoded! These should be obfuscated, even if only for testing, and much harder to guess than “root” “root”. Moreover, all passwords should be salted and hashed – none of that appears to be done here.

Also, error checking and messaging needs to be implemented. Only a few classes and methods have any user implemented error checking. Because, Artemis will not want to disclose more information than they mean to, e.g., “your password is incorrect” or “array out of bounds”, these messages need to be scrutinized and sanitized from “extra information” leaking to anyone trying to find cracks in the code. For example, if the connection to the SQL database, in DocData --> read\_document, fails an entire stack trace is returned! This could easily lead to exploits.

## 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 dependency check report.*

|  |  |  |  |
| --- | --- | --- | --- |
| Dependency | Vulnerability Code | Description | Mitigation |
| [bcprov-jdk15on-1.46.jar](file:///C:\Users\BTH\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7) | 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:legion-of-the-bouncy-castle-java-crytographyapi:1.46  cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46 | The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms | Upgrade Bouncy castle to version 1.60 or higher |
| tomcat-embed-websocket-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30 | Core Tomcat implementation | Update Tomcat to the newest version |
| tomcat-embed-core-9.0.30.jar | cpe:2.3:a:apache:tomcat:9.0.30  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30 | Core Tomcat implementation | Upgrade to newest Tomcat version |
| spring-core-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release  cpe:2.3:a:springsource:spring\_framework:5.2.3:release  cpe:2.3:a:vmware:spring\_framework:5.2.3:release  cpe:2.3:a:vmware:springsource\_spring\_framework:5.2.3:release | Spring Core | Upgrade to the latest version of Spring |
| spring-aop-5.2.3.RELEASE.jar | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release  cpe:2.3:a:springsource:spring\_framework:5.2.3:release  cpe:2.3:a:vmware:spring\_framework:5.2.3:release | Spring AOP | Upgrade to the latest version of Spring |
| spring-boot-2.2.4.RELEASE.jar | cpe:2.3:a:vmware:spring\_boot:2.2.4:release | Spring Boot | Only affects unsupported versions. Upgrade. |
| snakeyaml-1.25.jar | cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25 | YAML 1.1 parser and emitter for Java | Allows entity expansion. Upgrade. |
| jackson-databind-2.10.2.jar | cpe:2.3:a:fasterxml:jackson-databind:2.10.2  cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2 | General data-binding functionality for Jackson: works on core streaming API | Data integrity concerns. Upgrade to latest release. |
| logback-core-1.2.3.jar | cpe:2.3:a:qos:logback:1.2.3 | logback-core module | Arbitrary code execution. Upgrade to the latest version |
| hibernate-validator-6.0.18.Final.jar | cpe:2.3:a:redhat:hibernate\_validator:6.0.18 | Hibernate's Bean Validation (JSR-380) reference implementation | Input validation bypass issue. Upgrade to the latest version. |
| log4j-api-2.12.1.jar | cpe:2.3:a:apache:log4j:2.12.1 | The Apache Log4j API | Allows Man-in-the-middle attack on SMTP. Upgrade. |

## 5. Mitigation Plan

*After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial’s software application.*

Firstly, all of the dependent libraries need to be updated. This could lead to new code being written so this should be the first step.

Then, after the code recompiles and passes any internal tests the code needs to be refactored to add in the sanitized error messages, input validation/sanitization, and security improvements (salting and hashing).

Code review or peer programming should also be considered as there are some potentially large security issues within the code.