****

# 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** | **11/12/2021** | **Anthony See** | **Updated**  **Recommendations** |

## Client



## Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found.

Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Anthony See

## 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?

Secure communications are valued very highly to the Artemis. Artemis is a financial company that handles all sorts of financial transactions and money. Which means that security needs to be the best it can be. This is needed to make sure their clients’ accounts and transactions are kept safe.

As for international transactions, nothing was found to suggest that Artemis handles them. However, it is recommended that the software should be able to handle it. That is only if it doesn’t create a security issue for their current needs.

There are a large number of government restrictions to consider about secure communications, especially for financial data. Most of the laws require data to be correctly encrypted and reasonably secure against attackers. Another restriction is that the government requires the destruction of unreadable and unused information. This isn’t necessarily something to stress over though. All that is required is data to be secure and destroyed correctly.

External threats that are current as well as future, are attackers wanting sensitive information, rival companies, and other governments.

Modernizing the web application would include making the company more web-based than before. Being able to access accounts without having to drive the bank is convenient for a lot of people. This, also, means that data connections must be as secure as possible, which open source libraries can help with. Open source programs have the ability to add more layers of security and usability to the application.

## 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.

The areas that are applicable to Artemis Financial’s application are: API, Cryptography, client/server, and code quality.

The API is needed because the application uses APIs to run. The use of extra APIs may be needed as well to add extra security. These libraries need to be looked at and checked to make sure there’s no issues between them and the system, as well as, leaving no open doors for attackers.

Cryptography is used because the data that is being transferred will be send over the internet, which means that it needs to be encrypted. Without the encryption, the data can be attacked and modified during transmission, doing harm to Artemis and its clients. With correct encrypting and decrypting, these attacks can be nullified.

Client/Server is required as the customer of Artemis is communicating with a server to deliver financial information. If any of the data is compromised, then personal data can be stolen, resulting in a bad review for Artemis.

Code quality is another requirement as the quality of the code helps prevent attacks. Handling of errors or mistyped inputs is a large portion as this could easily lead to remote code execution, which is something that the system does not want.

## 3. Manual Review

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

DocData.java

In this file, there is a test for a database name, which uses a default username and password. This should be changed or removed, as this can be used to brute force the system and gain access to sensitive information.

Pom.xml

There are a few APIs that can be updated:

* BouncyCastle 1.46 to 1.69
* SpringFramework 2.2.4 to 5.3.8

Updating these will provide better security and more functionality to the application.

## 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. Include the following:

1. The names or vulnerability codes of the known vulnerabilities
2. A brief description and recommended solutions provided by the dependency check report
3. Attribution (if any) that documents how this vulnerability has been identified or documented previously

**Dependency**: bcprov-jdk15on-1.46.jar (Bouncy Castle)

Bouncy Castle provides cryptographic algorithms.

**Vulnerabilities:**

* CVE-2013-1624: Time issues
* CVE-2016-1000338: Inject elements into signature data
* CVE-2016-1000339: Is possibly to monitor CPU lookup table accesses
* CVE-2016-1000341: Timing attack
* CVE-2016-1000342: Can inject elements into signature data
* CVE-2016-1000343: DSA generator can generate a weak key
* CVE-2016-1000344: Use of ECB mode, which is unsafe
* CVE-2016-1000345: Observing timings can show failed decryptions
* CVE-2016-1000346: DH public key is not fully validated
* CVE-2016-1000352: Use of ECB mode, which is unsafe
* CVE-2017-13098: Provides a weak Bleichenbacher oracle
* CVE-2018-1000613: Allows remote code execution
* CVE-2018-5382: HMAC is 16 bits long, can be used to compromise BKS keystore

**Report’s recommendation:** Update to latest Bouncy Castle

**Dependency:** log4j-api-2.12.1.jar

The Apache Log4j API

**Vulnerabilities:**

* CVE-2020-9488: Improper certificate validation with host

**Dependency:** snakeyaml-1.25.jar

YAML 1.1 parser and emitter for Java

**Vulnerabilities:**

* CVE-2017-18640: Allow entity expansion during load

**Dependency:** jackson-databind-2.10.2.jar

General data-binding functionality for Jackson

**Vulnerabilities:**

* CVE-2020-25649: Did not secure data expansion properly

**Report’s recommended solution:** Update to latest Jackson API

**Dependency:** tomcat-embed-core-9.0.30.jar

Core Tomcat implementation

**Vulnerabilities:**

* CVE-2019-17569: Refactoring presented regression that could lead to HTTP Request Smuggling
* CVE-2020-11996: Specific sequence of HTTP requests could trigger high CPU usage
* CVE-2020-13934: H2C direct did not release HTTP processor, leading to DOS
* CVE-2020-13935: Invalid payload lengths could trigger infinite loops -> DOS
* CVE-2020-17527: Re-use of HTTP headers could possibly leak information
* CVE-2020-1935: Invalid HTTP headers could be mistaken as valid -> HTTP Smuggling
* CVE-2020-1938: Specific open IP addresses could allow for remote code execution
* CVE-2020-8022: Incorrect default perms could allow attackers more access
* CVE-2020-9484: Remote code execution if certain criteria are met
* CVE-2021-25122: Duplicate request headers. User A and B would see A’s information
* CVE-2021-25329: Fix of CVE-2020-9484 was incomplete using a specific configuration
* CVE-2021-30640: Vulnerability in JNDI Realm allows authenticity of attacks to appear valid
* CVE-2021-33037: Does not correctly parse HTTP transfer encoding header.
* CVE-2021-41079: Does not properly validate incoming TLS packets
* CVE-2021-42340: Fix for 63362 introduced a memory leak

**Report’s recommendation:** Update to latest tomcat

**Dependency:** hibernate-validator-6.0.18.Final.jar

Hibernate’s Bean Validation reference implementation

**Vulnerabilities:**

* CVE-2020-10693: Enables invalid EL expressions to appear valid

**Dependency:** spring-core-5.2.3.RELEASE.jar

Spring Framework’s core

**Vulnerabilities:**

* CVE-2020-5421: Allows protections against RFD attacks to be bypassed
* CVE-2021-22096: Allows malicious input to cause additional log entries
* CVE-2021-22118: WebFlux application is vulnerable to privilege escalation

**Report’s recommendation:** Update Spring to latest version

**Dependency:** spring-jcl-5.2.3.RELEASE.jar

Spring commons logging bridge

**Vulnerabilities:**

* CVE-2020-5421: Allows protections against RFD attacks to be bypassed

## 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.

The dependencies in section 4 had vulnerabilities mainly because of the program using an outdated version. It is highly recommended to update any API and library in use, especially Bouncy Castle and TomCat, as both of those have the most vulnerabilities.

An update to the Spring Framework would be ideal as well, as it may provide functionality that could be considered useful to the application. It may, also, fix unknown flaws and bugs that the in-use version has.

The pom.xml file should be examined after updating to make sure that the correct versions of the APIs are being used.

The data access username and password needs to be correctly created or removed. The username and password should not be related to each other in any way. Both should have a strong combination of letters, numbers, and symbols. Said username and password should, also, be changed frequently, to help prevent personnel from leaking the login.