****

# 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** | **Mathew Newton** |  |

## 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

Mathew Newton

## 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?
  + Since Artemis Financial is a financial organization, information must come from their servers to locations, customers, and venders in an orderly and secured manner. If the data was to be captured by an external threat, it will potentially lead to customers’ identities being stolen, having funds transferred to spoof accounts, and a host of other seriously offensive breaches that could affect a customer’s information and money stolen.
* Are there any international transactions that the company produces?
  + It is possible that Artemis Financial may have some international clients. Despite that they may only offer their services to US based clients, those clients may sometimes be in another country and want access to their account information or to complete financial transactions while abroad in the world, so international transactions are possible.
* Are there governmental restrictions about secure communications to consider?
  + Yes, there are several governmental restrictions about secure communications for financial institutions as outlined below:
    - The Sarbanes-Oxley Act requires audits be completed to make sure data is stored and managed in a secure manner. This includes monitoring, logging, and auditing activity that could potentially be at risk, maintaining access controls, and completing regular data backups on the regular.
    - The Gramm-Leach-Bliley Act requires private financial information be collected, stored, and used in a manner that protects user data. Financial institutions are required to follow the Safeguard’s Rule, which, required a written plan to address the secure management of financial user data.
    - The Payment Card Industry Data Security Standard details the requirements any organization that maintain cardholder data must follow to protect their users and organization. The PCI Security Standards Council provides documentation on all things that are needed for specifications and updates to help a financial institution stay compliant.
* What external threats might be present now and in the immediate future?
  + Malware
  + Phishing
  + Ransomware
  + DDoS attacks
  + Hacking into ATMs or database systems (Know the types of cyber threats)
* What are the “modernization” requirements that must be considered, such as the role of open-source libraries and evolving web application technologies?
  + Modernizing is important for protecting data and bringing old technologies up-to-date. Even newer versions of the same libraries may include security patches that could help protect data. In addition, using modern techniques such as two-factor authentication can help to protect private user data from potential threats.

## 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.

* Input Validation – users will need to enter usernames, passwords, and potentially other credentials. This information should be validated to prevent potential vulnerabilities.
* APIs – Since the web application is a RESTful application, the REST API should be checked for potential vulnerabilities and threats.
* Cryptography – Encryption is required for financial data stored by financial institutions, so the methods used for encryption should be reviewed (Probasco).
* Code Error – The code should be checked for errors that could impact security.
* Code Quality – The code should be checked to see if secure practices and patterns are being utilized.

## 3. Manual Review

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

* Input Validation – When the parameter is requested at /greeting, it isn’t validated prior to being formatted and added to the string “template”. This creates a vulnerability where additional, potentially dangerous information could be included with the string, or the string could be formatted so that it would cause an error that could stop the program from continuing.
* APIs – GET is used instead of POST when requesting a parameter at /greeting. This is less secure because the parameter will be passed as part of the url, making it more vulnerable.

## 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

Bcprov-jdk15on-1.46-jar

Update the version to the latest on available.

* CVE-2013-1624
* CVE-2016-6644
* CVE-2015-7940
* CVE-2016-100338
* CVE-2016-100339
* CVE-2016-100341
* CVE-2016-100342
* CVE-2016-100343
* CVE-2016-100344
* CVE-2016-100345
* CVE-2016-100346
* CVE-2016-100352
* CVE-2017-13098
* CVE-2018-1000613
* CVE-2018-5382

Log4j-api-2.12.1.jar

Update the version to the latest one available.

* CVE-2020-9488 Snakeyalm-1.25.jar

Update to the latest version available.

* CVE-2017-08640 Jackson-databind-2.10.2.jar

Update to the latest version available.

* CVE-2020-25649 Tomcat-embeded-core-9.0.30.jar

Update to the latest version available.

* CVE-2019-17569 CVE-2020-11996
* CVE-2020-13934 CVE-2020-13935
* CVE-2020-17527 CVE-2020-1935
* CVE-2020-1938 CVE-2020-8022
* CVE-2021-24122 Hibernate-validator-6.0.18.final.jar

Update to the latest version available.

* CVE-2020-10693 Spring-core-5.2.3release.jar

Update to the latest version available.

* CVE-2020-5421

## 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 following are the recommendations to remedy the above discussed security vulnerabilities: 1. By validating user input, prior to formatting it to be entered into the string by whitelisting acceptable characters and confirming the length of the input does not exceed a specified length (for example, 25 characters). This will resolve a vulnerability found in both manual review and static testing, which found a vulnerability in SnakeYAML 1.18 resolved with these steps..

2. Use POST instead of GET when requesting input parameters at /greeting to protect the input. 3. Upgrade bcprov-jdk15on-1.46.jar to version 1.61 (the highest required by the vulnerabilities discovered).

4. Pass user input as an expression variable, unwrapping the context and sending it through HibernateConstraintValidatorContext to avoid the vulnerability in the Hibernate Validator.

5. The vulnerability in FasterXML Jackson Databind, should be corrected adding the following two settings:

* factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true); factory.setFeature("http://apache.org/xml/features/nonvalidating/load-external-dtd", false);

6. Upgrade log4j-api-2.12.1.jar to version 2.13.2

7. Upgrade spring-core-5.2.3 to version 5.2.9

8. Upgrade tomcat-embed-core-9.0.30.jar to version 9.0.43 (the highest required by the vulnerabilities discovered).