****

# 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** | **7/18/21** | **Jeremia Faust** |  |

## 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

Jeremia Faust

## 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?

The company Global Rain is a financial consulting company that deals with very sensitive personally data so secure communication is very important for the company. The data needs to be secured by using the most current communication protocols. This is more important as the company grows to work with international customers. Working internationally, you have to take in account the local regulations alongside the company’s home government. Europe has some very strict privacy laws along with some hefty fines if your found at fault. In the United States the New York department of financial services laws, “According to Section 500.15 (a):As part of its cybersecurity program, based on its Risk Assessment, each Covered Entity shall implement controls, including encryption, to protect Nonpublic Information held or transmitted by the Covered Entity both in transit over external networks and at rest.” This is just one of the many laws that needs to be followed so it is very important that you are familiar with all of them to protect the data and yourself of lawsuits. This is something that should be considered before any work is done as it will affect how we protect the company. As a financial company and as you grow, hackers are going to attack looking for personal data and for ways to steal money from the company. So, it is important to stay up to date to protect against new threats and other threats. Today’s “modernization” in security is the use of open-source libraries and web application technologies to cobble together top-level security. It is like a Frankenstein of different libraries and technologies that work together to try to make the most secure system possible. You should never think that maintaining security is done when the program goes live. It will be a constantly changing so naturally the programs security will continue to be updated and changed.

## 2. Areas of Security

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

* Input Validation: Any data received is considered unsafe until it has been thoroughly checked and validated. Never trust anything coming in, you never know when a hacker adds some smelly code to get into the system.
* API: They are a set of rules to allow programs to talk with each other securely. This is used to prevent unauthorized access to the system.
* Cryptography: any sensitive data needed to be encrypted to protect the data before it leaves the system and the client need to have the proper certificates to access the data. As data travels between server and client, it is a prime time for data to be intercepted. An example would be hashing passwords.
* Client/server: All communication between the server and client needs to be protected. In this case we are using RestAPI to facilitate that. We just need to make sure that the data going out is protected and the data coming in is checked.
* Code error: All code needs to be checked for errors. Errors can end up becoming major security problem.
* Code Quality: Since this program deals with a lot of sensitive data it is extra important to practice secure coding practices such as default deny or even something as heeding compiler warnings.
* Encapsulation: Everything should be encapsulated and only allowed to provide data without interacting with the data structures of the program.

## 3. Manual Review

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

* + HTTPS is not used
  + In the CRUDcontroller.java: line 12 and 13, the business name is sent as a request
  + DocData.java: no user authentication of the username or the password, Also the access method also tells the location for the database. The test username is root, and the password is also root which leaves the system open to unauthorized users. Also, there is an unused variable according to the compiler

“Connection con=DriverManager.getConnection ("jdbc:mysql://localhost:3306/test","root","root");

## 4. Static Testing

Run a dependency check on Artemis Financials’ 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-check version: 6.2.2
* Report Generated On: Sat, 17 Jul 2021 06:45:54 -0400
* Dependencies Scanned: 38 (18 unique)
* Vulnerable Dependencies: 8
* Vulnerabilities Found: 49
* Vulnerabilities Suppressed: 0
* bcprov-jdk15on-1.46.jar
* Severity: Unknown
* Code: CVE-2013-1624
* Description: 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.
* The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169.
* hibernate-validator-6.0.18.Final.jar
* Severity: Medium
* Code: CVE-2020-10693
* Description: Hibernate's Bean Validation (JSR-380) reference implementation.
* A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages.
* jackson-databind-2.10.2.jar
* Severity: High
* Code: CVE-2020-25649
* Description: General data-binding functionality for Jackson works on core streaming API
* A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.
* log4j-api-2.12.1.jar
* Severity: Low
* Code: CVE-2020-9488
* Description: The Apache Log4j API
* Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender.
* snakeyaml-1.25.jar
* Severity: High
* Code: CVE-2017-18640
* Description: YAML 1.1 parser and emitter for Java
* The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.
* spring-core-5.2.3.RELEASE.jar
* Severity: High
* Code: CVE-2020-5421
* Description: Spring Core
* In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter.
* tomcat-embed-core-9.0.30.jar
* Severity: Critical
* Code: CVE-2019-17569
* Description: Core Tomcat implementation
* The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.
* tomcat-embed-websocket-9.0.30.jar
* Severity: Critical
* Code: CVE-2019-17569
* Description: Core Tomcat implementation
* The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.

## 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 Financials’ software application.

**Widely accepted solutions:**

* tomcat-embed-websocket-9.0.30.jar: It is recommended that you update to the most current version of 9.0.31.
* tomcat-embed-core-9.0.30.jar: It is recommended that you update to the most current version.
* spring-core-5.2.3.RELEASE.jar: It is recommended that you update to the most current version of 5.2.9.
* snakeyaml-1.25.jar: this could be a false positive due to a configuration file. Otherwise, migrate to snakeYAML engine
* jackson-databind-2.10.2.jar: It is recommended that this is patched to the most current patch
* hibernate-validator-6.0.18.Final.jar: : It is recommended that the intern fix containing APAR PH29942 is applied
* log4j-api-2.12.1.jar: It is recommended that you update to the most current version
* bcprov-jdk15on-1.46.jar: It is recommended that you update to the most current version

While reviewing the results of the manual review of the code and the dependency check report there were some problems that needed to be addressed to protect this system. The dependency check report indicated that the dependencies need to be update to the most current versions to repair any holes in the security of the system. It is recommended that Hypertext Transfer Protocol or HTTP to be incorporated into the system. This will protect the communication between the client and the server in addition to the restAPI. The Data access of the user and password should never be hard coded in this system. There also should be an authentication to protect the system from SQL attacks. A good example would be to have the system reject passwords that are the same as the username. The code should be modified to prevent information to be leaked out such as the businesses name of the location of the database. This should always stay encrypted to protect the data.

says:, L. W. (2021, March 11). *10 Data Privacy and Encryption Laws Every Business Needs to Know*. Hashed Out by The SSL Store™. https://www.thesslstore.com/blog/10-data-privacy-and-encryption-laws-every-business-needs-to-know/.