GLOBALRAIN

CS 305 Project One
Artemis Financial Vulnerability Assessment Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	**11/14/21**	**Hunter Webster**	

Client

ARTEMIS FINANCIAL

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
W. Hunter Webster

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?

Due to the highly sensitive nature of the customer data that Artemis financial handles (bank account numbers, financial information, etc.), the security of all communications across the new web-based platform is of high concern. Any breaches may result in severe financial loss and customer dissatisfaction that would negatively affect the company. Proper security measures should be in place such that these risks are minimized. At this point, there is not enough information to indicate whether the company handles international customers; however, it is likely that the individualized financial plans could contain international stocks or bonds. While regulatory bodies strongly encourage service providers to implement thorough security measures, there are no governmental mandates on secure communications. Despite the lack of regulation, any team of developers working on the new platform should be mindful that attacks on customer data are financially incentivized so attempts to attack the new platform will be inevitable. Potential attacks that Artemis Financial might face include phishing attacks, local malware injection via USB devices, or SQL injections. While there are many means of deterring these attacks, some may slip through the cracks due to human error. The company must also consider the security of any APIs implemented as the platform is modernized. Improper API integration may lead to unnecessary data vulnerabilities being introduced to the system. There are a couple of new technologies that I believe should be in place when modernizing the platform. First, two-factor authentication should be a high priority to deter any login attempts from non-authorized users. Second, all communication with the new system should require the use of HTTPS as a means of providing heightened security to all user information exchanged between users and the server.

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.

Code quality is extremely integral in creating a code base with a level of security appropriate for the type of information being handled by the application. Ensuring secure code will help to deter attempts to access sensitive data by unauthorized users and should be a high priority. Similarly, ensuring that all errors are properly handled is a key issue to be mindful of when attempting to create a secure platform. Errors like invalid login attempts or failed two-factor authentication should be carefully handled to protect users from malicious actors. To further protect the company from attacks, measures should be taken to validate all inputs to deter potential injection attacks. APIs are also relevant to this project as the platform employs RESTful API. Lastly, I believe that cryptography should be a core aspect of development as the information being handled will involve highly sensitive user information being transmitted and stored on the company's servers.

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

- Requests are not being validated by the system, exposing the system to malicious actors.
- The service is lacking HTTPS integration. This leaves potentially sensitive user information exposed to external entities.
- The CRUDController class is sending business names as request parameters which may expose such info to external entities
- There is currently no authentication system for verification.
- DocData.java includes a hard-coded password.
- customer.java allows for the customers account number to be displayed without verification.

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:

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

- bcprov-jdk15on-1.46.jar
    - Description: Many vulnerabilities on version 1.46
    - Solution: update to latest version.
    - CVE-2013-1624
        - Description: TLS implementation is vulnerable to side-channel attacks.
    - CVE-2015-6644
        - Description: Information disclosure vulnerability.
    - CVE-2015-7940
        - Description: Improper validation, potentially allowing outside access to private keys.
    - CVE-2016-1000338
        - Description: Improper validation, potentially allowing outside users to inject extra elements.
    - CVE-2016-1000339
        - Description: Flaw in dependency that potential enabled information to leak.
    - CVE-2016-1000341
        - Description: Timing attack vulnerability.
    - CVE-2016-1000342
        - Description: Improper validation, potentially allowing outside users to inject extra elements.
    - CVE-2016-1000343

        - Description: flaw in private key generation if generated with default parameters.
    - CVE-2016-1000344
        - Description: Allows use of ECB mode.
    - CVE-2016-1000345
        - Description: Vulnerability to padding oracle attack.
    - CVE-2016-1000346
        - Description: Other party public key is not fully validated.
    - CVE-2016-1000352
        - Description: Allows use of ECB mode.
    - CVE-2017-13098
        - Description: Cryptography vulnerability.
    - CVE-2018-1000613
        - Description: Cryptography vulnerability.
    - CVE-2018-5382
        - Description: I'm going to be honest, I don't understand this one.
- Log4j-api-2.12.1.jar
    - CVE-2020-9488
        - Description: Improper validation of certificate, potentially allowing man-in-the-middle-attacks
        - Solution: update to latest version.
- Snakeyalm-1.25.jar
    - CVE-2017-18640
        - Description: Allows entity expansion during load operations.
        - Solution: update to latest version.
- Jackson-databind-2.10.2.jar
    - CVE-2020-25649
        - Description: Entity expansion is not properly secured, allowing XXE attacks.
        - Solution: update to latest version.
- Tomcat-embed-core-9.0.30.jar
    - Description: many vulnerabilities on version 9.0.30
    - Solution: update to latest version.
    - CVE-2019-17569
        - Description: Invalid Transfer-Encoding headers.
    - CVE-2020-11996
        - Description: Potential to unintentionally trigger temporary high CPU usage. Server may become unresponsive with a high number of requests.
    - CVE-2020-13934
        - Description: A high number of requests might lead to an OutOfMemoryException. Denial of service risk.
    - CVE-2020-13935
        - Description: Invalid payload requests may result in an infinite loop. Denial of service risk.
    - CVE-2020-13943
        - Description: Potential for unexpected resources to be transmitted to users if there are too many concurrent streams for a connection.
    - CVE-2020-17527

        - Description: Potential to accidentally re-use a previous HTTP request header which would likely result in the closure of the connection.
    - CVE-2020-1935
        - Description: Flaw in HTTP header parsing which may allow invalid headers.
    - CVE-2020-1938
        - Description: Flaw in Apache that gives greater trust to Apache JServ Protocol connections. May allow for remote code execution.
    - CVE-2020-9484
        - Description: Potential for remote code execution.

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.

In order to ensure that customer information is kept as secure as possible, there a few solutions that should be implemented. First, an authentication system should be put in place to mitigated unauthorized data access. Second, greater measures should be added to validate user inputs. Third, HTTPS should be required as it is standard in modern web-development and will serve as another level of security against external actors. Fourth, all dependencies should be updated to the latest version and many of the identified issues will be resolved.