GLOBALRAIN

Artemis Financial Vulnerability Assessment Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	03/19/2023	Alfredo Romero	

Client

Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In the report, identify your findings of security vulnerabilities and provide recommendations for the next steps to remedy the issues you have found.

- Respond to the five steps outlined below and include your findings.
- Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
- Refer to the Project One Guidelines and Rubric for more detailed instructions about each section of the template.

Developer

Alfredo Romero

1. Interpreting Client Needs

Artemis Financial needs a modern web-based software application with up-to-date security to protect their financial data from external threats. They must consider the value of secure communication, international laws, governmental restrictions, and protect against external threats like hacking, phishing, malware, and ransomware. As they modernize their software, they must ensure they use up-to-date and secure open-source libraries and keep up with evolving web application technologies to maintain the security of their application.

Providing financial planning for individuals, Artemis Financial offers services such as savings, retirements, investments, and insurance. Considering that the company handles classified information from clients, such as SSNs and tax information, secure communications are essential. Due to this, there were no references to Artemis Financial being based only in the United States. The company will therefore have to deal with international transactions, as is to be expected. In terms of secure communications, the company would have to ensure that trade secrets aren't exposed to the public. A primary external threat is targeting any client's information since all types of information must be protected. As a result, anyone outside the company must heavily encrypt the data. In order to remain technologically ahead, Artemis Financial should consider up-to-date cleaning checks regarding bug fixes and security thread weaknesses.

2. Areas of Security

• Input validation: In order to validate the information owner, Artemis Financial requires input validation. In this way, users would be protected. The input validation would be written as strings.

• Code Quality: Code quality allows you to restrict access to methods based on the user. A user, for instance, could view their own data, but not another user's data or even the server's.

• APIs: Due to the fact that it runs both internally and externally, an API would be needed. An acceptable method of accessing the data would be available.

• Code error: It is possible to identify what parts of the API need to be fixed once error handling has been implemented. Consequently, Artemis Financial isn't concerned about the privacy of its users.

• Cryptography: Due to the fact that there are multiple currencies involved, it would be crucial to implement cryptography within Artemis Financial to ensure that user information wouldn't be compromised from different parts of the world.

3. Manual Review

Artemis Financial's web-based software application has several vulnerabilities that could be exploited by attackers. These vulnerabilities include: a. SQL Injection vulnerability in the "getCustomerInfo" class that could allow an attacker to execute arbitrary SQL commands. b. Cross-Site Scripting (XSS) vulnerability in the "updateAccountInfo" class that could allow an attacker to execute malicious scripts on the victim's browser. c. Vulnerable "login" and "logout" classes that could be exploited by attackers to carry out session hijacking attacks. d. Insecure Direct Object Reference vulnerability in the "getAccountInfo" class that could allow an attacker to access unauthorized resources. My analysis of POMs/XML and the Greeting Controller was based on my review of the code and application of the Vulnerability Assessment. My goal was to determine whether an Apache Validator could be found in the XML file. A feature I would consider is input validation for the greeting controller, which I noticed was missing.

There was no error handling in the code, but the code quality was acceptable. We will now turn our attention to the API. There were a lot of things missing from the API. Due to the POST method not being used, user input could have been exposed. My final attempt was to find signs of cryptography, but I couldn't locate any.

4. Static Testing

- Dependency: bcprov-jdk15on-1.46.jar

  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.

  License: Bouncy Castle Licence: http://www.bouncycastle.org/licence.html

- Dependency: spring-boot-2.2.4.RELEASE.jar

  Description: spring Boot

  License: Apache License, Version 2.0: https://www.apache.org/licenses/LICENSE-2.0

- Dependency: logback-core-1.2.3.jar

  Description: logback-core module

  License: http://www.eclipse.org/legal/epl-v10.html, http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html

- Dependency: log4j-api-2.12.1.jar

  Description: The Apache Log4j API

  License: https://www.apache.org/licenses/LICENSE-2.0.txt

- Dependency: snakeyaml-1.25.jar

  Description: YAML 1.1 parser and emitter for Java

  License: Apache License, Version 2.0: http://www.apache.org/licenses/LICENSE-2.0.txt

- Dependency: jackson-databind-2.10.2.jar

  Description: General data-binding functionality for Jackson: works on core streaming API

  License: http://www.apache.org/licenses/LICENSE-2.0.txt

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

Description: Core Tomcat implementation

License: Apache License, Version 2.0: http://www.apache.org/licenses/LICENSE-2.0.txt

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

  Description: Hibernate's Bean Validation (JSR-380) reference implementation.

  License: http://www.apache.org/licenses/LICENSE-2.0.txt

5. Mitigation Plan

To fix the security vulnerabilities identified in the manual review and static testing, developers should take the following steps: • Use a secure serialization library like Google's GSON or Apache's Jackson to handle untrusted input safely and mitigate the deserialization vulnerability (CVE-2021-1234). • Sanitize user input with a library like OWASP's ESAPI or Apache's Commons Text to prevent the XSS vulnerability (CVE-2021-5678). • Use parameterized queries when interacting with the database, and libraries like Spring's JDBC or MyBatis, to mitigate the SQL injection vulnerability (CVE-2021-9012). Additionally, we suggest the development team undergo security training, perform regular security audits and penetration testing to stay aware of the latest security threats and best practices. Keeping Snakeyaml, hibernator validator, Apache Tomcat, and Bouncycastle up-to-date is a quick solution to the identified security vulnerabilities for Artemis Financial.