****

# 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** | **07/23/23** | **Zack Silerzio** |  |

## 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

Zack Silerzio

## Interpreting Client Needs

Artemis Financial's Requirements:

Artemis Financial, a consulting company, wants to modernize their operations by using a web-based software application to develop individualized financial plans for their customers. The application includes features related to savings, retirement, investments, and insurance. As part of their modernization, they want to ensure the highest level of software security to protect their customers' sensitive financial information.

Potential Threats and Attacks: Given the nature of financial planning and the sensitive data involved, Artemis Financial's application may be vulnerable to various external threats and attacks. Some of these threats include:

1. Unauthorized Access: Hackers and malicious actors may attempt to gain unauthorized access to the application to steal sensitive customer data or manipulate financial plans.
2. Data Breach: Inadequate security measures could lead to a data breach, compromising customer information and damaging Artemis Financial's reputation.
3. Injection Attacks: SQL injection and other code injection attacks could exploit vulnerabilities in the application, allowing attackers to execute malicious code.
4. Cross-Site Scripting (XSS): XSS attacks might enable attackers to inject malicious scripts into the application, affecting other users and compromising their data
5. Denial of Service (DoS) Attacks: Attackers may attempt to overwhelm the application's resources, causing service disruptions and affecting customer experience.

## Areas of Security

Based on the nature of Artemis Financial's web-based software application and potential threats, the relevant areas of security that apply to their application are:

1. Access Control: To prevent unauthorized access to sensitive financial data, proper access control mechanisms must be in place, such as authentication and authorization.
2. Data Security: Measures to secure data both in transit and at rest are crucial to protect customer information from unauthorized access or data breaches.
3. Input Validation: Implementing proper input validation techniques will help defend against injection attacks and XSS vulnerabilities.
4. Web Application Firewall (WAF): Implementing a WAF can help detect and block malicious traffic and attempts to exploit vulnerabilities.

## Manual Review

After a manual review of the codebase provided by Artemis Financial, the following vulnerabilities were identified:

1. Insecure Data Handling: The **customer** class contains an account balance variable that can be directly manipulated without proper validation or security checks, leaving the application open to unauthorized changes in account balances.
2. Lack of Input Validation: The **CRUDController** class accepts a parameter 'business\_name' directly from the client without proper input validation, leaving it susceptible to injection attacks.

## Static Testing

[Insert text.]

## Mitigation Plan

To address the identified security vulnerabilities, the following mitigation plan is recommended:

1. Secure Data Handling: Implement proper validation and authentication mechanisms for any data manipulation, including financial account balances.
2. Input Validation: Apply strict input validation checks for all user-provided data to prevent injection attacks.
3. Web Application Firewall: Set up a Web Application Firewall to monitor and filter incoming traffic, blocking potential threats.
4. Regular Security Updates: Stay updated with security patches and updates for all software components, including open-source libraries.
5. Security Training: Provide security awareness training for all developers and personnel involved in the software development process.
6. Third-party Code Review: Conduct regular reviews of third-party libraries and components used in the application to identify and address potential vulnerabilities.