

**Artemis Financial Vulnerability Assessment Report**

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| **1.0** | **[Date]** | **[Your name]** |  |

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

[Femi Abdul]

* Interpreting Client Needs

[As part of the client needs interpretation phase, it was identified that security requirements were not clear. The client needs to provide explicit security requirements to ensure that the development team can integrate appropriate security measures into the application.

* Areas of Security

[In the areas of security, the vulnerability assessment focused on the following key aspects:

Architecture Review: The analysis of the application architecture revealed potential vulnerabilities in the flow from Input Validation to Code Quality. There is a need for a more comprehensive architecture review to identify and address architectural weaknesses.

Input Validation: Secure Input and Representations were found to be adequately handled.

APIs: Secure API Interactions were appropriately implemented.

Cryptography: Encryption Use and Vulnerabilities were properly addressed.

Client/Server: Secure Distributed Composing was implemented effectively.

Code Error: Secure Code Handling showed areas for improvement, especially in error handling.

Code Quality: While Secure Coding Practices/Patterns were in place, there were instances where Code Quality could be enhanced.

Encapsulation: Secure Data Structures were adequately maintained.]

* Manual Review

[Manual review identified potential issues in the following areas during the code review process:

Views: There were vulnerabilities identified in the presentation layer.

Models: The data modeling components revealed potential security weaknesses.

Controllers: Security concerns were noted in the application's control logic.

Data Access: Vulnerabilities in data access patterns were identified.

Services: Security issues were found in the application services.

Plug-Ins: There were potential vulnerabilities in the plug-ins used.

APIs: Secure API Interactions were validated during the manual review.

* Static Testing

[Static testing revealed that certain aspects of the code, especially in the areas of error handling and code quality, require improvement. A more thorough static testing process is recommended to identify and address potential vulnerabilities in the codebase.]

* Mitigation Plan

[To mitigate the above-mentioned vulnerabilities, the following potential mitigation techniques can be applied:

Client Needs: Request the client to provide explicit security requirements for the application.

Architecture Review: Conduct a detailed architecture review to identify and address architectural weaknesses.

Code Error and Code Quality: Implement improvements in error handling and code quality practices.

Manual Review: Address vulnerabilities identified during manual reviews in Views, Models, Controllers, Data Access, Services, Plug-Ins, and APIs.

Static Testing: Enhance static testing processes to identify and rectify code vulnerabilities..]