Quack AI

Audit Report

Wed Aug 13 2025

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

Quack AI Audit Report

1 Executive Summary

1.1 Project Information

Description	It is a simple proposal project.
Type	Infrastructure, SocialFi
Auditors	ScaleBit
Timeline	Sun Aug 10 2025 - Sun Aug 10 2025
Languages	Solidity
Platform	Others
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/quackai-labs/quackai-beta
Commits	80f3b069dbfcd9e92e9c9bd4155148f75ff8135b

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
GMO	GovernanceModule.sol	6440913a83fc8fcb0be442c6c3356 b40144de89a

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	4	0	4
Informational	2	0	2
Minor	0	0	0
Medium	2	0	2
Major	0	0	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Arkm to identify any potential issues and vulnerabilities in the source code of the Quack Ai smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 4 issues of varying severity, listed below.

ID	Title	Severity	Status
GMO-1	Missing Time Validation	Medium	Acknowledged
GMO-2	Centralization Risk	Medium	Acknowledged
GMO-3	Unued Function	Informational	Acknowledged
GMO-4	Single-Step Transfer Can be Dangeous	Informational	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the Quack Ai Smart Contract :

Owner

- `Owner` can add a single account to the whitelist via `addWhiteAccount()` .
- `Owner` can add multiple accounts to the whitelist via `addWhiteAccount()` .
- `Owner` can remove an account from the whitelist via `removeWhiteAccount()` .

WhiteAddr

- `WhiteAddr` can create proposals via `createVote()` .
- `WhiteAddr` can vote on proposals via `submit()` .
- `WhiteAddr` can change proposals via `fixVote()` .

4 Findings

GMO-1 Missing Time Validation

Severity: Medium

Status: Acknowledged

Code Location:

GovernanceModule.sol#227-304

Descriptions:

In the `submit` function, there is no validation of the end time, which allows users to vote after the proposal has ended.

Suggestion:

It is recommended to add an end-time validation.

Resolution:

The client replied: In our current business logic, users do not directly interact with the on-chain voting contract. Instead, they sign their votes locally, which are then processed by our backend and batch-submitted to the contract.
This architecture ensures that end-users cannot bypass backend logic to directly submit on-chain votes, nor can they add or alter votes after a proposal's end time.

GMO-2 Centralization Risk

Severity: Medium

Status: Acknowledged

Code Location:

GovernanceModule.sol

Descriptions:

Centralization risk was identified in the smart contract:

- Admin can add whitelisted users through the `addWhiteAccount()` function.
- Admin can remove whitelisted users through the `removeWhiteAccount()` function.

Suggestion:

It is recommended that measures be taken to reduce the risk of centralization, such as a multi-signature mechanism.

GMO-3 Unued Function

Severity: Informational

Status: Acknowledged

Code Location:

GovernanceModule.sol#6-13,92-97

Descriptions:

Unused code may reduce code maintainability.

Suggestion:

It is recommended to remove these unused code segments.

GMO-4 Single-Step Transfer Can be Dangeous

Severity: Informational

Status: Acknowledged

Code Location:

GovernanceModule.sol#35-38

Descriptions:

In the contract, transferring owner privileges in a single step carries certain risks, as user errors may lead to permanent loss of ownership.

Suggestion:

It is recommended to use Ownable2Step instead.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.