Mazu Finance Smart Contract

Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

Fri May 17 2024

Mazu Finance Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	A yield farming project
Type	DeFi
Auditors	MoveBit
Timeline	Wed Apr 24 2024 - Wed Apr 24 2024
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/thounyy/mazu
Commits	24983097c8489b16f136ed72b19bc13a1a1761b1 069abf588551f064a044e6f3f963b12e54739397

1.2 Files in Scope

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

ID	File	SHA-1 Hash
VES	package/sources/vesting.move	7b1ee5d0d85be38af0adcf77a07acbaf8304fe53
MAT1	package/sources/math.move	9a76ed0655734437870a919fd878df0a84a7e0f7
STA	package/sources/staking.move	5d6534db384ef2c018f4057d42c8f6c1899b6df4
MUL	package/sources/multisig.move	602aee82eaee56d7d97d24041febab489e510904
COI	package/sources/coin.move	913b18c6a3ba41cd3a464cf480084096cbaba1d3
AIR	package/sources/airdrop.move	3bae24c47e06ddaa4de0aae294ae596e4522f4d3

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	4	1	3
Informational	2	0	2
Minor	1	1	0
Medium	1	0	1
Major	0	0	0
Critical	0	0	0

1.4 MoveBit Audit Breakdown

MoveBit 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 by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

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) Formal Verification

Perform formal verification for key functions with the Move Prover.

(4) 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 Mazu to identify any potential issues and vulnerabilities in the source code of the Mazu Finance 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
AIR-1	Inconsistent With Multi-Sign Design	Medium	Acknowledged
COI-1	Unused Constants	Minor	Fixed
COI-2	Improper `CoinMetadata` Processing	Informational	Acknowledged
MUL-1	Anyone Can Delete Expired Proposals	Informational	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the Mazu Finance Smart Contract :

Admin

- The `Admin` can create a proposal through `create_proposal()` .
- The `Admin` can cancel a proposal through `delete_proposal()` .
- The `Admin` can vote for a proposal through `approve_proposal()` .
- The `Admin` can remove their votes through `remove_approval()` .
- The `Admin` can execute a proposal when it reach on its threshold through `execute_proposal()` .
- The `Admin` can propose and execute an airdrop through `propose()` , `start()` , `drop()` .
- The `Admin` can propose and execute a transfer through `propose_transfer()` , `start_transfer()` and `complete_transfer()` .
- The `Admin` can propose and change the metadata of a coin through `propose_update_metadata()` , `start_update_metadata()` and `complete_update_metadata()` .
- The `Admin` can propose and change member list through `propose_modify()` , `start_modify()` and `complete_modify()` .
- The `Admin` can propose and set the time of a staking through `propose_start()` , `start_start()` and `complete_start()` .
- The `Admin` can propose and dispatch locked assets through `propose()` , `start()` , `new()` and `complete()` .

User

- The `User` can use `Ticket` to claim an airdrop through `claim()` .
- The `User` can stake coins through `stake()` .
- The `User` can claim rewards through `claim()` .
- The `User` can unstake coins through `unstake()` .
- The `User` can use `Locked` to claim reward when time is up through `unlock()` .

4 Findings

AIR-1 Inconsistent With Multi-Sign Design

Severity: Medium

Status: Acknowledged

Code Location:

package/sources/airdrop.move#75

Descriptions:

When airdropping a proposal, the airdrop quantity is entered after the proposal. The airdrop proposal is only for permissions rather than quantity, which may result in the airdrop quantity not being the expected quantity.

Suggestion:

It is recommended to include the parameters in the proposal.

Resolution:

The client replied that the airdrop would be done by them before configuring the multi-sig (only one member is the client). It will be simpler and everyone has to trust them to get the right addresses anyway.

COI-1 Unused Constants

Severity: Minor

Status: Fixed

Code Location:

package/sources/coin.move#20,24;

package/sources/vesting.move#18,19

Descriptions:

There are unused constants in the contract.

```
const MAX_SUPPLY: u64 = 888_888_888_888_888_888;
const EUnknownStakeholder: u64 = 0;
const MAX_TEAM: u64 = 88_888_888_888_888_888 * 2;
const MAX_PRIVATE_SALE: u64 = 88_888_888_888_888_888;
```

Suggestion:

It is recommended to remove unused constants if there's no further design.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

COI-2 Improper `CoinMetadata` Processing

Severity: Informational

Status: Acknowledged

Code Location:

package/sources/coin.move#78

Descriptions:

On Sui, `CoinMetadata` contains crucial information about a token, such as name and symbol. If this information is altered, some users might perceive it as a new token, potentially causing confusion among users.

Suggestion:

It is recommended to freeze `CoinMetadata` if the coin information will not be modified in the future.

MUL-1 Anyone Can Delete Expired Proposals

Severity: Informational

Status: Acknowledged

Code Location:

package/sources/multisig.move#64

Descriptions:

Anyone can delete expired proposals. This period is hard-coded to 7 days.A malicious attacker who can constantly delete proposals will make historical proposals cannot be traced back.

Suggestion:

Please make sure this matches your design.

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