Meso

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Wed May 14 2025

Meso Audit Report

1 Executive Summary

1.1 Project Information

Description	Meso Finance is a decentralized, non-custodial, pool-based lending platform built on the Aptos designed to provide users with efficient and secure lending services. Meso Finance enables users to supply assets to earn interest and borrow against them to unlock liquidity
Type	DeFi
Auditors	MoveBit
Timeline	Tue May 06 2025 - Wed May 14 2025
Languages	Move
Platform	Aptos
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/MesoLendingFi/meso-token
Commits	d8ea680e865137ef1800a47d1023d7e16057eaeb d1320a08d6d21f2928c6f26a33ec890ff94dd3c6

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	Move.toml	c4375c84cfbf8e2cb6c08f607d5c7cae6ecc145f
MTO	sources/meso_token.move	81cf44810215cef05b6ac35cef6f1e434cd3b7dc
GST	sources/global_state.move	39e5aa422805faa385ad7035d0717e9bb32630af
TVE	sources/token_vesting.move	5604e766992169e4d16f49d28aa93087388b35e1

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	4	4	0
Informational	2	2	0
Minor	1	1	0
Medium	1	1	0
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(Optional)

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 Meso to identify any potential issues and vulnerabilities in the source code of the Meso 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
GST-1	Single-step Ownership Transfer Can be Dangerous	Medium	Fixed
TVE-1	`add_claimers_entry` Lack Of Length Detection	Minor	Fixed
TVE-2	`ENOT_BENEFICAIARY_MUST_NOT_ZERO` Unused Parameters	Informational	Fixed
TVE-3	Rename `get_avaiable_token` to `get_avaliable_token`	Informational	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Meso Smart Contract :
Admin

- `set_policies_entry` : Administrators update certain configurations related to the token release strategy
- `add_claimers_entry` : Add multiple claimants and the amount of tokens they are allocated to the specified lock-up configuration
- `remove_claimer_entry` : Remove a recipient from the designated lock-up plan and reset the amount of tokens allocated to them to zero
- `collect` Recover undistributed or unclaimed tokens after the lock-up plan ends
- `update_operator` Update the current operator address
- `update_governance` Update the current governance address
- `freeze_stores` Freeze the token storage of designated users and prohibit these users from performing token transfer operations
- `unfreeze_stores` : Unfreeze the token storage of the designated users and restore their token transfer permissions

User

- `claim_entry` : Users claim the tokens they can claim in the lock-up plan

4 Findings

GST-1 Single-step Ownership Transfer Can be Dangerous

Severity: Medium

Status: Fixed

Code Location:

sources/global_state.move#59-70

Descriptions:

Single-step ownership transfer means that if a wrong address was passed when transferring ownership or admin rights it can mean that role is lost forever. If the admin permissions are given to the wrong address within this function, it will cause irreparable damage to the contract. The `update_operator()` function and `update_governance()` function directly transfers the admin role to a new address, which poses the above-mentioned risks.

```
public entry fun update_operator(governor: &signer, new_operator: address) acquires
AdministrativeData {
        only_governance(governor);
        unchecked_mut_admin_data().operator = new_operator;
    }

public entry fun update_governance(
        governance: &signer,
        new_governance: address,
    ) acquires AdministrativeData {
        only_governance(governance);
        unchecked_mut_admin_data().governance = new_governance;
    }
```

Suggestion:

It is recommended to use a two-step ownership transfer process.

Resolution:

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

TVE-1 `add_claimers_entry` Lack Of Length Detection

Severity: Minor

Status: Fixed

Code Location:

sources/token_vesting.move#291-306

Descriptions:

When you call vector::zip, the order is claimers and allocates. Make sure the lengths of the two vectors are the same; otherwise, zip will be truncated. However, the code does not check whether the lengths are consistent, which may result in some allocations not being processed

Suggestion:

Add check:

```
assert!(vector::length(&claimers) == vector::length(&allocates), error_code)
```

Resolution:

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

TVE-2 `ENOT_BENEFICAIARY_MUST_NOT_ZERO` Unused Parameters

Severity: Informational

Status: Fixed

Code Location:

sources/token_vesting.move#46

Descriptions:

The parameter ENOT_BENEFICAIARY_MUST_NOT_ZERO is not used in the contract

Suggestion:

Add the corresponding logic or delete the redundant code

Resolution:

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

TVE-3 Rename `get_avaiable_token` to `get_avaliable_token`

Severity: Informational

Status: Fixed

Code Location:

sources/token_vesting.move#188

Descriptions:

There is a word spelling mistake in the get_avaiable_token function

Suggestion:

Modify 'get_avaiable_token' to 'get_avaliable_token'

Resolution:

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

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.