Haedal Protocol

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Thu Dec 11 2025

Haedal Protocol Audit Report

1 Executive Summary

1.1 Project Information

Description	A sophisticated DeFi vault system built on Sui blockchain that manages assets across multiple liquidity market types (DLMM and CLMM)
Type	DeFi
Auditors	Alex,PeiQi0
Timeline	Mon Dec 08 2025 - Thu Dec 11 2025
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/haedallsd/haevault/
Commits	d24e26b8f36e50864daf0db36b8bc409f1c33cbf f9881cdd89b22098509c2d34533688df76d4ee9b

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	packages/volatile_vault_v2/Move.toml	059ee138f3e4a049812b990676c513afcf7090f1
MCL	packages/volatile_vault_v2/sources/market_clmm.move	2e0198b17e53703053ad8e73bfcb01f0aacd0ef5
VAU	packages/volatile_vault_v2/sources/vault.move	52b2490d698d772685ecde31949239e5b9705b66

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	1	1	0
Critical	0	0	0
Major	0	0	0
Medium	0	0	0
Minor	1	1	0
Informational	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 Haedal to identify any potential issues and vulnerabilities in the source code of the Haedal Protocol 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 1 issues of varying severity, listed below.

ID	Title	Severity	Status
MCL-1	The `new_position` Function is not Used	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Haedal Protocol Smart Contract :

Vault Manager

- `add_clmm_market` : The vault manager can add a new CLMM market by calling the `add_clmm_market()` function.
- `add_clmm_certified_pool` : The vault manager can add a new certified CLMM pool by calling the `add_clmm_certified_pool()` function.
- `remove_clmm_certified_pool` : The vault manager can remove a certified CLMM pool by calling the `remove_clmm_certified_pool()` function.
- `update_clmm_certified_pool` : The vault manager can update a certified CLMM pool by calling the `update_clmm_certified_pool()` function.

Keeper Manager

- `open_clmm_position` : The keeper manager can open a CLMM position by calling the `open_clmm_position()` function.
- `increase_clmm_liquidity` : The keeper manager can increase CLMM liquidity by calling the `increase_clmm_liquidity()` function.
- `decrease_clmm_liquidity` : The keeper manager can decrease CLMM liquidity by calling the `decrease_clmm_liquidity()` function.
- `close_clmm_position` : The keeper manager can close a CLMM position by calling the `close_clmm_position()` function.

User

- `deposit` : Users can deposit assets into the protocol by calling the `deposit()` function.
- `calculate_aum` : Users can calculate the total value of all DLMM and CLMM assets by calling the `calculate_aum()` function.

- `calculate_clmm_position_amounts` : Users can update their CLMM position information by calling the `calculate_clmm_position_amounts()` function.
- `withdraw_clmm` : Users can withdraw their CLMM assets by calling the `withdraw_clmm()` function.
- `collect_clmm_fee_on_withdraw` : Users can collect their CLMM fees upon withdrawal by calling the `collect_clmm_fee_on_withdraw()` function.
- `collect_clmm_fee` : Users can collect their CLMM fees by calling the `collect_clmm_fee()` function.
- `collect_clmm_reward_on_withdraw` : Users can collect their CLMM rewards upon withdrawal by calling the `collect_clmm_reward_on_withdraw()` function.
- `collect_clmm_reward` : Users can collect their CLMM rewards by calling the `collect_clmm_reward()` function.

4 Findings

MCL-1 The `new_position` Function is not Used

Severity: Minor

Status: Fixed

Code Location:

packages/volatile_vault_v2/sources/market_clmm.move#149-160

Descriptions:

The `new_position()` function is not used anywhere in the `market_clmm` contract.

```
public(package) fun new_position(market: &mut CLMMMarket, position: Position, ctx:
&mut TxContext) {
    assert!(market.certified_pools.contains(&position.pool_id()), EUncertifiedPool);
    let position_id = object::id(&position);
    let wrap_position = CLMMPosition {
        id: object::new(ctx),
        position,
        amounts: vec_map::empty(),
        last_txs: vec_map::empty(),
    };
    market.position_keys.insert(position_id);
    market.positions.add(position_id, wrap_position);
}
```

Suggestion:

It is recommended to remove this function.

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.