Haedal Farming

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Wed Oct 22 2025

Haedal Farming Audit Report

1 Executive Summary

1.1 Project Information

Description	Haedal is a prime liquid staking protocol natively built on Sui. It provides users with robust liquid staking infrastructure, allowing anyone to stake their SUI & WAL tokens to contribute to the governance and decentralization of the network, while earning continual consensus rewards and unleashing LST liquidity to be used in DeFi On top of its liquid staking protocol, Haedal is also building a series of simple yield products including Haedal Market Maker and more, which generate continuous additional on-chain yields for Haedal and its LST ecosystem Haedal serves as a core pillar of the Sui DeFi by merging native liquid staking and yield strategies with user-friendly accessibility. Aim to empower users to maximize capital efficiency through innovative liquid staking and algorithmic DeFi yield solutions, and build Haedal into the ultimate place to stake and earn on Sui
Type	Staking

Auditors	MoveBit
Timeline	Tue Aug 26 2025 - Wed Oct 22 2025
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/haedallsd/farming-contract
Commits	d734f693e91de4a6fc14340171739d955fd8eb8f 7835153827bd4ec271a27b099fd1ae53c991cf85

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	Move.toml	849f15afb9bdc985ee95c15f11eeddcad25a6bf7
INT	sources/interface.move	8de61576e38f95f9ee7038743172456f3976179a
COR	sources/core.move	6d49466e925633f41da27f8eaf3b8f1c80b8af32
OPE	sources/operate.move	87a954d3829cd90c1359d260f940b3b0f59cafee
MAN	sources/manage.move	d0b177adb6972c0c02fd2e30eb3a1bd689a43cd0
ROB	sources/robot.move	e8c65323655848df77230127db3a6ebeebafe143
COR	sources/core.move	00aa19f691fbb0c8e92204b2135114dfaebcb36c
MIN	sources/minorsign.move	a0a55da9d29c4b09346815a3ced90ebe21ddde95
BRE	sources/breaker.move	9f9c2ec4b676e5f2ac1f2de402fbb9ada44959ce
OPE	sources/operate.move	5797ab233acff19d9b279fd52725d58f0a93e27a

MAN	sources/manage.move	6bfa8dee1cb162df109df391a8c5439ee89bd3a0

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	1	0	1
Informational	0	0	0
Minor	1	0	1
Medium	0	0	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 Haedal to identify any potential issues and vulnerabilities in the source code of the Haedal Farming 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
MAN-1	Centralization Risk	Minor	Acknowledged

3 Participant Process

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

Admin:

- `add_pool` : Administrator creates a new staking pool. It checks whether a pool of the same type already exists and reverts with `EPoolDuplicate` if so.
- `set_pool` : Enables or disables a specific pool by toggling its `active` status. Used for activating or pausing staking operations.
- `add_reward_config` : Adds a new reward token configuration for a pool. It validates the start time and prevents adding a duplicate reward type for the same pool.
- `set_reward_config` : Updates an existing reward configuration's reward rate and start time, allowing administrators to adjust ongoing reward plans.
- `funding_bank` : Injects funds into the reward bank to ensure there are sufficient tokens available for distribution. Typically used when reward reserves run low.
- `extract_bank` : Withdraws reward tokens from the reward bank, usually for administrative withdrawal, fund migration, or manual redistribution.
- `flip_boost` : Toggles the boost multiplier feature on or off for a particular reward type, controlling whether boost-based reward acceleration is active.
- `settle` : Forces the settlement of a specific deposit, often during contract upgrades, system maintenance, or manual balance reconciliation.
- `set_boost` : Assigns a custom boost multiplier to a specific address (for example, granting VIP privileges or special bonus rates).
- `after_set_boost` : Synchronizes the user's debt and share data after a boost multiplier has been changed, ensuring consistency in reward accounting.
- `add_minor_signs_to_acl` : Add a new minor signer address.
- `del_minor_signs` : Remove a minor signer address.
- `add_breaker_to_acl` : Add a new breaker address.

- `del_breaker_to_acl` : Remove a breaker address.
- `add_robot_to_acl` : Add a new robot address.
- `del_robot_to_acl` : Remove a robot address.

User:

- `deposit` : User makes their first deposit into the pool, creating a corresponding `Deposit` object. The function checks to ensure the user has not already participated in the same pool.
- `add_deposit` : Allows a user to add more tokens to an existing staking position, increasing their total deposit amount in the same pool.
- `get_deposit` : Retrieves the address of the `Deposit` object associated with the user in the specified pool, used for viewing deposit details.
- `withdraw` : Enables the user to withdraw their staked tokens from the pool. It verifies the balance, ownership, and pool association before proceeding.
- `harvest` : Lets the user claim their accumulated reward tokens (`RewardCoin`) from the reward bank, transferring them to the user's account.
- `remaining` : Returns the amount of unclaimed rewards for a user, calculated per reward token type, allowing users to check pending rewards.

4 Findings

MAN-1 Centralization Risk

Severity: Minor

Status: Acknowledged

Code Location:

sources/manage.move#41,50,56,62,68

Descriptions:

The module defines multiple Caps:

- AdminCap
- OperatorCap
- MinorSignCap
- BreakerCap
- RobotCap

Currently, all critical privilege assignments are fully controlled by AdminCap, for example:

```
public entry fun set_operator_cap_to_address(_: &AdminCap, account: address, ctx: &mut TxContext)
public entry fun set_minor_sign_cap_to_address(_: &AdminCap, account: address, ctx: &mut TxContext)
public entry fun set_breaker_cap_to_address(_: &AdminCap, account: address, ctx: &mut TxContext)
public entry fun set_robot_cap_to_address(_: &AdminCap, account: address, ctx: &mut TxContext)
```

This implies:

- All critical capability issuance is centralized in a single administrator.

- If the AdminCap is misused or compromised, an attacker can fully control the system, including issuing arbitrary OperatorCap, BreakerCap, or RobotCap, and performing data migration or upgrade operations.
- The system lacks decentralization or multi-signature protection, creating a single point of failure.

Suggestion:

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

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.