Haedal

Audit Report

MOVEBIT

contact@bitslab.xyz

https://twitter.com/movebit_

Mon Dec 04 2023

Haedal Audit Report

1 Executive Summary

1.1 Project Information

Description	Haedal is a liquid staking protocol built on Sui that allows anyone to stake their SUI tokens.
Type	Staking
Auditors	MoveBit
Timeline	Mon Nov 20 2023 - Mon Dec 04 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review

1.2 Files in Scope

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

ID	File	SHA-1 Hash
HAS	sources/hasui.move	043c51935bddf988961f1a5790f253dda55b90f6
TQU	sources/table_queue.move	02257831926d5787f5a90d68061ec8747c0136fb
MOV	haedal_v2/Move.toml	1732bc625cfa7fd969afa11e890c31ebba91124f
INT	haedal_v2/sources/interface.move	8bf4d0d584e8fd20c12c3932576648a9b40f341c
UTI	haedal_v2/sources/util.move	c09ed85724835f264a8e9b61970f7ef1f2821785
STA	haedal_v2/sources/staking.move	42f5621d5bf950835eb9f2b6e2ac84c6cb4b53fd
CON	haedal_v2/sources/config.move	0f5766ff21302f57d2b4101871147f4e67b38534
VAU	haedal_v2/sources/vault.move	c4f6a0535cfb5b57379e78c401b2142831aaddec
OPE	haedal_v2/sources/operate.move	9186d4025380a8f395ef3922089087121eadd880
MAN	haedal_v2/sources/manage.move	7a566f3d9757c16b256d1d65a7d28308d9255526

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	5	3	2
Informational	3	1	2
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

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 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 5 issues of varying severity, listed below.

ID	Title	Severity	Status
CON-1	Unused Config	Informational	Acknowledged
STA-1	Incomplete Function About Protocol Fee	Medium	Fixed
STA-2	Users Cannot Unstake Small Amounts In `request_unstake_instant` Due To Service Fee	Minor	Fixed
STA-3	Unused Struct	Informational	Acknowledged
STA-4	`EUnstakeNotEnoughSui` Is Not Appropriate For Certain Edge Condition	Informational	Fixed

3 Participant Process

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

Admin

- The Admin can initialize the `Staking` object through `initialize()` .
- The Admin can set the deposit fee through `set_deposit_fee()` .
- The Admin can set the protocol fee through `set_reward_fee()` .
- The Admin can set the validator reward fee through `set_validator_reward_fee()` .
- The Admin can set the service fee through `set_service_fee()` .
- The Admin can set the withdraw time limit through `set_withdraw_time_limit()` .
- The Admin can set validator count through `set_validator_count()` .
- The Admin can Migrate the data version through `migrate()` .
- The Admin can collect the protocol fee through `collect_rewards_fee()` .
- The Admin can collect the service fee through `collect_service_fee()` .
- The Admin can toggle the stake status through `toggle_stake()` .
- The Admin can toggle the unstake status through `toggle_unstake()` .
- The Admin can toggle the claim status through `toggle_claim()` .
- The Admin can stake the users' `SUI` to validators through `do_stake()` .
- The Admin can update the exchange rate of `haSUI/SUI` through `update_total_rewards_onchain()` .
- The Admin can unstake the `StakedSui` objects from inactive validators through `unstake_inactive_validators()` .
- The Admin can unstake the `StakedSui` to approve the claim `SUI` through `do_unstake_onchain()` .
- The Admin can unstake all the `StakedSui` when the validators are risky through `unstake_pools()` .

Operator

- Operator can toggle the staking status with `toggle_stake()` .

- Operator can toggle the unstaking status with `toggle_unstake()`.
- Operator can toggle the claim status with `toggle_claim()`.
- Operator can stake `SUI` to validators with `do_stake()`.
- Operator can update the `haSUI/SUI` exchange rate with `update_total_rewards_onchain()`.
- Operator can unstake `StakedSui` from inactive validators with `unstake_inactive_validators()`.
- Operator can unstake `StakedSui` for `SUI` claims with `do_unstake_onchain()`.
- Operator can unstake all `StakedSui` from risky validators with `unstake_pools()`.

User

- The User can stake their `SUI` and get `haSUI` through `request_stake()` and `import_stake_sui_vec()`.
- The User can burn their `haSUI` and get `SUI` immediately through `request_unstake_instant()`.
- The User can burn the `haSUI` and get `UnstakeTicket` through `request_unstake_delay()`.
- The User can claim the `SUI` from `UnstakeTicket` through `claim_v2()`.

4 Findings

CON-1 Unused Config

Severity: Informational

Status: Acknowledged

Code Location:

haedal_v2/sources/config.move

Descriptions:

The `deposit_fee` and `validator_reward_fee` are configured in the `config module`, but they are not used in the main logic contract.

Suggestion:

It is recommended to remove the related operations if there is no future usage plan for `deposit_fee` and `validator_reward_fee`.

Resolution:

It is acknowledged by the dev team.

STA-1 Incomplete Function About Protocol Fee

Severity: Medium

Status: Fixed

Code Location:

haedal_v2/sources/staking.move#552

Descriptions:

The calculation of protocol fees is performed in the `update_total_rewards_onchain` function, but no substantive functionality for collecting protocol fees has been found in the contract, which will result in the admin not being able to collect the protocol fee.

Suggestion:

it is recommended to add the collect protocol fee operations.

Resolution:

The client modified the code and fixed this issue.

STA-2 Users Cannot Unstake Small Amounts In `request_unstake_instant` Due To Service Fee

Severity: Minor

Status: Fixed

Code Location:

haedal_v2/sources/staking.move#314-347

Descriptions:

In `request_unstake_instant` users can unstake any amount from the vault balance and there's some service fee being taken from users.

However, this design will prohibit small amount unstakes.Since there are no min unstaking threshold, users can unstake as small as 1 mist.

However, let's say if `max_exchange_sui_amount` = 10 mist, then take `service_fee` as the default one which is 90.

```
let fee_amount = ((max_exchange_sui_amount as u128) * (service_fee as u128) /
(FEE_DENOMINATOR as u128) as u64);
```

`fee_amount = 10*90/1000 = 0`. And this will fail the below assertion even though `service_fee` is certainly above 0.

```
assert!((service_fee == 0 || fee_amount > 0), EUnstakeInstantNoServiceFee);
```

Similar issues arise when `service_fee` is approching 0 but not euqal to 0 yet. And this effectively disabling users to unstake in certain amounts.

Suggestion:

It is suggested to set a min unstaking threshold, or enable users to unstake small amounts by removing service fee.

Resolution:

The client modified the code and fixed this issue.

STA-3 Unused Struct

Severity: Informational

Status: Acknowledged

Code Location:

haedal_v2/sources/staking.move#121,184

Descriptions:

The structs `EpochClaim` and `SystemUnstaked` are not used in the `staking` module.

Suggestion:

It is recommended to remove unused structs if there is no plan to use them in the future.

STA-4 EUnstakeNotEnoughSui Is Not Appropriate For Certain Edge Condition

Severity: Informational

Status: Fixed

Code Location:

haedal_v2/sources/staking.move#322;

haedal_v2/sources/staking.move#356

Descriptions:

In the `request_unstake_instant` function, there is an assertion to make sure that the `max_exchange_sui_amount` is in certain range.

```
assert!(max_exchange_sui_amount <= sui_vault_amount && max_exchange_sui_amount
> 0, EUnstakeNotEnoughSui);
```

However, if `max_exchange_sui_amount > sui_vault_amount` is met, it means maximum exchange amount exceeds the amount available in the vault, which does not align with `EUnstakeNotEnoughSui`.

Suggestion:

It is recommended to split the assertions and use error codes like `EExceedsVaultAmount` for the frist condition.

Resolution:

The client modified the code and set the minimum unstake threshold to 1 SUI.

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.

MOVEBIT VERIFIED

MOVEBIT