Navi Volo

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Tue Apr 29 2025

Navi Volo Audit Report

1 Executive Summary

1.1 Project Information

Description	Volo Liquid Staking V2 is an upgrade based on the original Volo v1. The main upgraded features include Instant Unstaking, Decentralized Reward Management, a new fee rebate mechanism, a new delegation mechanism, and fungible staked SUI.
Type	Staking
Auditors	MoveBit
Timeline	Mon Apr 21 2025 - Tue Apr 29 2025
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Sui-Volo/volo-liquid-staking-v2
Commits	b3ef7f209c9e5c54b123e32314fb908a739b3d62 79d6665b10b35ac516404980e44227e808616f7f 35fc942481c6b91c3a1e48e8d646ac6a5a44bc80

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	liquid_staking/Move.toml	654020ff2674339c694c483a0fbb8ac1129b6108
MIG	liquid_staking/sources/migration/migrate.move	e82ad79d23700254981158a89cc7c7ae28f3a098
NPO	liquid_staking/sources/volo_v1/native_pool.move	a601a585e099050732dfa152c499f2b61492c9b8
MAT	liquid_staking/sources/volo_v1/math.move	8500c22a771b44ab083c4454dd570bcb071fe778
VSE	liquid_staking/sources/volo_v1/validator_set.move	6a34223bfb02ca2f7bc91955486c0f32fb1ba7ef
OWN	liquid_staking/sources/volo_v1/ownership.move	f66384074267d7ba43542c39731876e2793f12bd
UTI	liquid_staking/sources/volo_v1/unstake_ticket.move	afdf6fb12c2b0a2842404013ad4e6dd6b640aed9
VPO	liquid_staking/sources/validator_pool.move	1d91f4864cb85cfe5239e9c230d82b32a2a48a36
CER	liquid_staking/sources/cert.move	54a708aaa28f247cbcb42449a1f6d4410862ec70
MAN	liquid_staking/sources/manage.move	59fdb5a448686e4583113a7b0b1a63007e29a42c

FCO	liquid_staking/sources/fee_config.move	1324132b2beb5ac6c53dae8c1a9c689e789b8473
SPO	liquid_staking/sources/stake_pool.move	191272d45de76e9ad79be8ab6203bbd2ab90fc80
MIG	liquid_staking/sources/migration/migrate.move	e232f6c436dfc39021b5fbcb56052ac1446bc5da
NPO	liquid_staking/sources/volo_v1/native_pool.move	813b240253999209221c6b526f7af03a35404611
SPO	liquid_staking/sources/stake_pool.move	778e3738e09236f052e107fa34bd111d6353e855

1.3 Issue Statistic

Item	Count	Fixed	Partially Fixed	Acknowledged
Total	11	7	1	3
Informational	3	0	0	3
Minor	5	5	0	0
Medium	3	2	1	0
Major	0	0	0	0
Critical	0	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 Navi Protocol to identify any potential issues and vulnerabilities in the source code of the Navi Volo 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 11 issues of varying severity, listed below.

ID	Title	Severity	Status
NPO-1	Unused Constants	Informational	Acknowledged
OWN-1	Single-step Ownership Transfer Can be Dangerous	Minor	Fixed
SPO-1	Centralization Risk	Medium	Partially Fixed
SPO-2	Incorrect Unstake Limit	Medium	Fixed
SPO-3	No Minimum Stake Amount Limit	Minor	Fixed
SPO-4	Test Case Failed to Run	Minor	Fixed
SPO-5	Missing Return Value Checking	Minor	Fixed
SPO-6	Incorrect Event parameter	Minor	Fixed
SPO-7	Redundant Code	Informational	Acknowledged
UTI-1	Missing `UnstakeTicket` Migration Mechanism	Medium	Fixed

VPO-1	Code Optimization	Informational	Acknowledged

3 Participant Process

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

Admin

- Admin can pause or unpause staking, unstaking, rebalance and refresh operations through the `set_paused()` function.
- Admin can collect accumulated fees through the `collect_fees()` function.
- Admin can update stake fee through the `update_stake_fee()` function.
- Admin can update unstake fee through the `update_unstake_fee()` function.
- Admin can update reward fee through the `update_reward_fee()` function.
- Admin can mint operator capabilities through the `mint_operator_cap()` function.
- Admin can modify boosted rewards through the `update_boosted_reward_amount()` function.

Operator

- Operator can set validator weights through the `set_validator_weights()` function.
- Operator can deposit boosted balance through the `deposit_boosted_balance()` function.

Owner

- Owner can pause or unpause the collect fee operation in native pool through the `set_pause()` function.
- Owner can collect accumulated fees through the `collect_fee()` function.
- Owner can upgrade contract through the `migrate()` function.
- Owner can transfer ownership through the `transfer_owner()` function.
- Owner can transfer operator permissions through the `transfer_operator()` function.

MigrationCap Holder

- MigrationCap holder can initialize migration process through the `init_objects()` function.
- MigrationCap holder can create V2 stake pool through the `create_stake_pool()` function.
- MigrationCap holder can export V1 stakes through the `export_stakes()` function.
- MigrationCap holder can collect protocol fees through the `take_unclaimed_fees()` function.
- MigrationCap holder can destroy migration capability through the `destroy_migration_cap()` function.
- MigrationCap holder can add emergency funds through the `deposit_sui()` function.
- MigrationCap holder can import stakes to new pool through the `import_stakes()` function.

User

- User can stake SUI through the `stake_entry()` function.
- User can unstake LST through the `unstake_entry()` function.
- User can delegate stake to specific validators through the `delegate_stake_entry()` function.

4 Findings

NPO-1 Unused Constants

Severity: Informational

Status: Acknowledged

Code Location:

liquid_staking/sources/volo_v1/native_pool.move#31,32,42,44-49

Descriptions:

Unused constants are defined in the contract.

Suggestion:

If the data is useless, they should either be used or removed.

OWN-1 Single-step Ownership Transfer Can be Dangerous

Severity: Minor

Status: Fixed

Code Location:

liquid_staking/sources/volo_v1/ownership.move#34

Descriptions:

The `transfer_owner()` function has a problem with single-step permission transfer. 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.

Suggestion:

It is recommended to use a two-step permission transfer mechanism. Reference: (https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable2Step.sol).

Resolution:

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

SPO-1 Centralization Risk

Severity: Medium

Status: Partially Fixed

Code Location:

liquid_staking/sources/stake_pool.move#327,373-462,;

liquid_staking/sources/volo_v1/ownership.move#34,48

Descriptions:

Centralization risk was identified in the smart contract:

- Admin can pause or unpause staking, unstaking, rebalance and refresh operations.
- Admin can set a very high stake fee.
- Admin can set a very high unstake fee.
- Admin can set a very high reward fee.
- Admin can mint operator capabilities.
- Admin can modify boosted rewards.
- Owner can transfer ownership to any address.
- Owner can transfer operator permissions to any address.
- Owner can pause or unpause the collect fee operation.
- Operator can set validator weights.

Suggestion:

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

Resolution:

The client replied that they have set a cap for each fee in `fee_config`, which mitigates the risk. Also the owner will be stored in the multi-sig wallet to ensure security.

SPO-2 Incorrect Unstake Limit

Severity: Medium

Status: Fixed

Code Location:

liquid_staking/sources/stake_pool.move#291

Descriptions:

The exchange ratio of `lst` and `sui` in the contract may not be one-to-one. If you stake 0.1 `sui`, you can only get 0.09 `lst`. Because of the check of `assert!(lst.value() >= MIN_STAKE_AMOUNT, EUnderMinAmount);`, this part of `lst` cannot be unstaken.

Suggestion:

It is recommended to set a reasonable minimum unstake value for `lst`.

Resolution:

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

SPO-3 No Minimum Stake Amount Limit

Severity: Minor

Status: Fixed

Code Location:

liquid_staking/sources/stake_pool.move#232

Descriptions:

The contract does not set a minimum stake amount, which may result in insufficient balance after deducting the handling fee, and the user receiving 0 `CERT`.

Suggestion:

It is recommended to set a reasonable minimum stake amount that is greater than the minimum handling fee to avoid users receiving 0 `CERT`.

Resolution:

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

SPO-4 Test Case Failed to Run

Severity: Minor

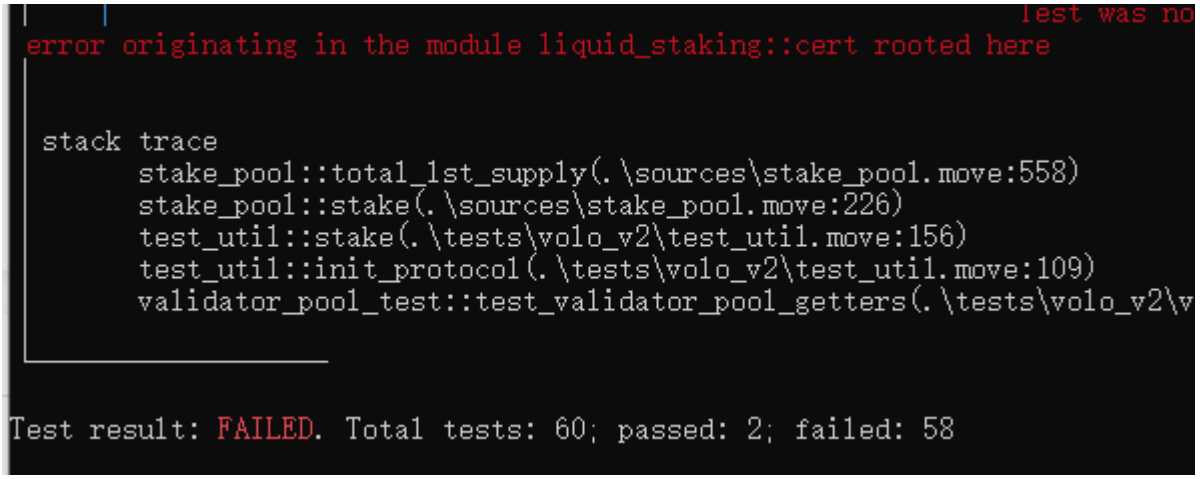
Status: Fixed

Code Location:

liquid_staking/sources/stake_pool.move

Descriptions:

The test cases in the tests directory failed to run.



Suggestion:

It is recommended to modify the test case and conduct actual testing.

Resolution:

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

SPO-5 Missing Return Value Checking

Severity: Minor

Status: Fixed

Code Location:

liquid_staking/sources/stake_pool.move#487

Descriptions:

There is no check on the return value of the `refresh()` function in the `rebalance()` function.

Suggestion:

Please confirm the impact of the `refresh()` function on subsequent operations and tell us why you did not check it. It is recommended to check the return value of functions that have return values.

Resolution:

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

SPO-6 Incorrect Event parameter

Severity: Minor

Status: Fixed

Code Location:

liquid_staking/sources/stake_pool.move#482,483

Descriptions:

The two parameters of the event `DepositBoostedBalanceEvent`, `before_balance` and `after_balance`, have the same values.

Suggestion:

It is recommended to change the value of `after_balance` to the value after `self.boosted_balance` is updated.

Resolution:

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

SPO-7 Redundant Code

Severity: Informational

Status: Acknowledged

Code Location:

liquid_staking/sources/stake_pool.move#177,268,485,486,499;

liquid_staking/sources/validator_pool.move#227

Descriptions:

Some version and pause checks in the contract are redundant because they are also included in the `refresh()` functions called later.

The `continue` statement in the `while` loop of the `validator_pool::refresh()` function is redundant.

Suggestion:

It is recommended to delete the duplicate code.

UTI-1 Missing `UnstakeTicket` Migration Mechanism

Severity: Medium

Status: Fixed

Code Location:

liquid_staking/sources/volo_v1/unstake_ticket.move#111

Descriptions:

The migration contract lacks the `UnstakeTicket` migration mechanism, and the V1 version of the contract will no longer be supported. This will cause users who previously held `UnstakeTicket` to be unable to use this ticket to unstake, and the pledged amount cannot be withdrawn, resulting in serious financial losses.

Suggestion:

It is recommended to add `UnstakeTicket` migration mechanism in the migration contract.

Resolution:

The development team adopted a solution both within the contract script and outside the contract to complete the migration.

VPO-1 Code Optimization

Severity: Informational

Status: Acknowledged

Code Location:

liquid_staking/sources/validator_pool.move#214-221

Descriptions:

The `refresh()` method uses two `if` judgments. They can be processed together to improve code readability.

Suggestion:

It is recommended to merge the two `if` processing blocks.

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.