Strater_lp_vault

Audit Report

contact@movebit.xyz

https://twitter.com/movebit_

Wed Jan 17 2024

Strater_lp_vault Audit Report

1 Executive Summary

1.1 Project Information

Description	A vault that convert Cetus non-fungible LP token to fungible LP token.
Type	DeFi
Auditors	MoveBit
Timeline	Mon Jan 15 2024 - Wed Jan 17 2024
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Strater-sui/cetus-lp-vault
Commits	5ddbbd084d7462431336cf9069cdeb8c74ef9be5 4d5c6ccd0376a53ecd02824634cc7d63e813cba1

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	Move.toml	183886ab9d6504739e37f96795abd2d939b27989
TMA	sources/tests/test_math.move	170b8dc0c9c9f43cb7430dc47b6343ecbd323301
BUC	sources/bucketus.move	69f73e52621f4d6430e44a77ba15ff68fcb6f8b0

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	5	4	1
Informational	0	0	0
Minor	1	1	0
Medium	2	1	1
Major	2	2	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 Bucket to identify any potential issues and vulnerabilities in the source code of the strater_lp_vault 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
BUC-1	`update_version` Can Be DOS	Major	Fixed
BUC-2	`vault.bucketus_supply` Being Updated Twice	Major	Fixed
BUC-3	Potential Price Manipulation	Medium	Acknowledged
BUC-4	There Is No Check Against Both `tick_lower` And `tick_upper`	Medium	Fixed
BUC-5	`create_vault` Lacks Of Version Control	Minor	Fixed

3 Participant Process

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

Admin

- `Admin` can create a new vault that handles a position in a liquidity pool for tokens of types A and B through `create_vault`

Beneficiary

- `Beneficiary` can claim and transfer fee to certain `recipient` through `claim_fee` and `claim_fee_to` functions

User

- `User` can deposit liquidity and get `BUCKETUS` coins through `deposit` function
- `User` can withdraw liquidity and burn `BUCKETUS` coins through `withdraw` function

4 Findings

BUC-1 `update_version` Can Be DOS

Severity: Major

Status: Fixed

Code Location:

sources/bucketus.move#177-182

Descriptions:

In the `update_version` it changes the `treasury.version` to a new `version` directly and publicly, however, there is no access control and anyone can call it.

When the malicious user calls it, major functions like `claim_fee`, `deposit`, and `withdraw` would not work due to the version check. And eventually, the smart contract will be DOSed.

Suggestion:

It is suggested to add the access control to the `update_version` public function.

Resolution:

It is fixed by the client by adding the access control to the `update_version` function.

BUC-2 `vault.bucketus_supply` Being Updated Twice

Severity: Major

Status: Fixed

Code Location:

sources/bucketus.move#205,302-310

Descriptions:

In the `deposit` function, a user is allowed to deposit a certain `delta_liquidity` and get corresponding `bucketus` coins back.

During this process, the total supply of `bucketus` is updated. However, it is added twice in both the inner `mint` function and the `deposit` function itself.

In this case, the supply is doubled and when a user tries to withdraw his liquidity back, he would get only half the value.

Suggestion:

It is suggested to remove the addition in the `deposit` function and keep the `mint` function the same.

Resolution:

It is fixed by the client by commenting out the duplicated update.

BUC-3 Potential Price Manipulation

Severity: Medium

Status: Acknowledged

Code Location:

sources/bucketus.move#211

Descriptions:

In the `deposit()` function, the calculation of `bucketus_amount` is based on the third-party protocol. Once the relied-upon third party is manipulated, the `bucketus_amount` will be incorrect. This could create arbitrage opportunities.

Suggestion:

It is recommended to confirm if it aligns with the design.

BUC-4 There Is No Check Against Both `tick_lower` And `tick_upper`

Severity: Medium

Status: Fixed

Code Location:

sources/bucketus.move#131-133

Descriptions:

In the `create_vault` function, it used `tick_lower` and `tick_upper` parameters to open the new position, however, there is no check to ensure that `tick_lower < tick_upper` .If the new vault is created with the wrong tick parameters, it will cause the dysfunctionality of the position.

Suggestion:

It is suggested to check that both parameters are in the correct range.

Resolution:

It is fixed by the client by checking the range of both `tick_upper` and `tick_lower` .

BUC-5 `create_vault` Lacks Of Version Control

Severity: Minor

Status: Fixed

Code Location:

sources/bucketus.move#118-154

Descriptions:

In the `create_vault` function, there is no version control compared with other functions.

In this case, Admin may create a new vault in a deprecated module.

Suggestion:

It is suggested to add the version control in the `create_vault` as well.

Resolution:

It is fixed by the client by adding the version control.

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.