Bucket Protocol

Audit Report

contact@movebit.xyz

https://twitter.com/movebit_

Mon Jan 08 2024

Bucket Protocol Audit Report

1 Executive Summary

1.1 Project Information

Description	Collateralized Debt Position (CDP) protocol within the Sui ecosystem
Type	DeFi
Auditors	MoveBit
Timeline	Tue Dec 26 2023 - Mon Jan 08 2024
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Bucket-Protocol/v1-core
Commits	73635637d389f8b8a87bb4914c7e5e494177545e 55eea23068adbd7114fa8ca765dd8ef7faac8324

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MAT	framework/sources/math.move	14b2d89adeec3f8c120648ddb757d38c76e59a0f
VLO	framework/sources/vesting_lock.move	a6fccadf49e3d13e8d2dd8bd02cda7a4d855d67e
LTA	framework/sources/linked_table.move	0169cd8710d12220fdaa53acc4c95aa177c7d85e
RES	protocol/sources/reservoir.move	a62c00e13d835dc3ec1e93caced07393dcf143e8
REV	protocol/sources/events/reservoir_events.move	d9eef37ea26c63a8b02856296297b80c8f2037e0
BEV	protocol/sources/events/buck_events.move	6d30d65903205aee7bae14a4c4c11ed1b2facdd4
BEV1	protocol/sources/events/bucket_events.move	71d3fb2e8dbf2d38126e448b7c1eb0062db0cfa0
TEV	protocol/sources/events/tank_events.move	8dce2891977ce26b7a9134246401f63b264bb27f
WEV	protocol/sources/events/well_events.move	d54fc22da7bdfb32aaa91dc8e7245dcfc3e934e2
WEL	protocol/sources/well.move	d7f1acfee9e4c2431ca54e21c3e17ffe0b869c52
CON	protocol/sources/config/constants.move	5cc296d14589b43709a64b2973ff6e78c4ff24f7

TAN	protocol/sources/tank.move	f317209a74425cdf65043781b47c0a4fb81eac2d
BOT	protocol/sources/bottle.move	e4d01400c478dc6379ac8c914f33cc6d4526d79e
BUC	protocol/sources/buck.move	7dc9b03235723aaf0918e707e82582f3750a586e
BKT	protocol/sources/bkt.move	f5b1fff63a4510c7e1c1870ac603ee2fdcb9f00d
BUC1	protocol/sources/bucket.move	5ace09f77384268e485d001424cf2e44c3f4a0d9

1.3 Issue Statistic

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

ID	Title	Severity	Status
BUC-1	`stake_amount` Can Be Manipulated During Flash Borrow	Critical	Fixed
BUC-2	Lack of Access Control	Major	Fixed
BUC-3	Lack of Version Validation	Medium	Fixed
BUC-4	Lack of Events Emit	Minor	Fixed
MAT-1	`mul_factor_u128` May Overflow	Medium	Fixed

3 Participant Process

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

Admin

- Admin can create a `vesting_lock` to lock arbitrarily `amountBKT` through `allocate_bkt`.

User

- User can top up collateral of user through `top_up`.
- User can borrow BUCK through `borrow`.
- User can repay BUCK and get collateral of user through `repay`.
- User can redeem BUCK and get collateral of user through `redeem`.
- User can withdraw `bkt_reward` and `collateral_withdrawal` from Tank through withdraw.
- User can deposit BUCK into Tank through `deposit`.
- User can stake BKT into Well through `stake`.
- User can get rewards from Well through `claim`.

4 Findings

BUC-1 `stake_amount` Can Be Manipulated During Flash Borrow

Severity: Critical

Status: Fixed

Code Location:

protocol/sources/buck.move#421

Descriptions:

We noticed in the `buck` module that the `stake_amount` used can be manipulated during `flash_borrow`. The `flash_borrow` function can split the token from the `collateral_vault` to the user, and then the `flash_repay` function collects the token and the fee. But there is a problem during this process.

Consider these steps:

1. prepare a position that can be liquidated
2. call `flash_borrow` from another address.
3. due to the fact that the balance of the `collateral_vault` has been modified, the liquidation function liquidate will call `update_snapshot` to update the value of the `total_collateral_snapshot`.
4. the user then returns the flash loan.
5. Finally, when the user calls the `compute_new_stake` function, the total_collateral_snapshot becomes smaller, resulting in the `new_stake_amount` becoming larger.

So when the user returns the token, the value of `collateral_vault` is not reduced, but the amount of `stake_amount` can be manipulated, which leads to a protocol vulnerability.

Suggestion:

It is recommended to restrict the user from calling other functions related to the protocol during `flash_borrow`.

Resolution:

The client followed the suggestion and fixed this issue.

BUC-2 Lack of Access Control

Severity: Major

Status: Fixed

Code Location:

protocol/sources/buck.move#753

Descriptions:

The `update_reservoir_fee_rate` function lacks access control that allows anyone to call the function to modify `charge_fee_rate` and dis `charge_fee_rate` .

Suggestion:

It is suggested to add access control for the `update_reservoir_fee_rate` function.

Resolution:

The client followed the suggestion and fixed this issue.

BUC-3 Lack of Version Validation

Severity: Medium

Status: Fixed

Code Location:

protocol/sources/buck.move#250 568 718 753

Descriptions:

There are some functions without validation for the `VERSION` parameter in the public function. However, it is validated in other functions which cause the old module can still call the upgraded BucketProtocol object. Ensuring the smart contract version is correct is crucial for preventing potential security risks.

Suggestion:

It is recommended to add version validation for those functions.

Resolution:

The client followed the suggestion and fixed this issue.

BUC-4 Lack of Events Emit

Severity: Minor

Status: Fixed

Code Location:

protocol/sources/buck.move#501-524 753

Descriptions:

The smart contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for those sensitive functions.

Resolution:

The client followed the suggestion and fixed this issue.

MAT-1 `mul_factor_u128` May Overflow

Severity: Medium

Status: Fixed

Code Location:

framework/sources/math.move#13-16

Descriptions:

In the function `mul_factor_u128()`, since both `number` and `numerator` are `u128`, their product might exceed the maximum value that a `u128` can hold.

Suggestion:

It is recommended to prevent overflow, you need to ensure that the multiplication `number * numerator` does not exceed the maximum value of `u128`.

Resolution:

The client followed the suggestion and fixed this issue.

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.