MOVEBIT
Securing the Move Ecosystem

Interest Protocol DEX Smart Contract **Audit Report**

https://twitter.com/movebit_

contact@movebit.xyz

Interest Protocol DEX Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	The DEX modules of Interest Protocol on the Sui Network.
Type	DEX
Auditors	MoveBit
Timeline	Apr 19th, 2023 – Mar 8th, 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/interest-protocol/sui-defi
Commits	3c447b70decb3e5befa1cceb21d357d19b680718 b7fd55307e26b53a8f435a1c880828f92a4dd42e

1.2 Files in Scope

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

ID	Files	SHA-1 Hash

COR	dex/sources/core.move	6cea9127bb3d8c71a1f39a5400331bac68d817ce
CUR	dex/sources/curve.move	634ab39428db02f2f828a13c20265a443deedd76
INT	dex/sources/interface.move	a344459a893d82e42e11942724737761d24edf0c
MCF	dex/sources/master-chef.move	481c0bcdee1a57a562ce704563aec806a1c8b9e7
ROU	dex/sources/router.move	d8bc42242fc5a79d1cbd5c2a1a2eaf86812754b3
IPX	ipx/sources/ipx.move	e825949037b3c32ce5228e81b2674c2f3ae0a581
COM	library/sources/comparator.move	2498162aa803f509c9a3d83ed6faf5dbda88f873
MTH	library/sources/math.move	eec0d470ef438e0d50a54f5625d84d82399a7bed
REB	library/sources/rebase.move	966bca5d4c39aa32d21a6ea97b84bf549571637e
UTL	library/sources/utils.move	702caaea6dfddb81db2a06ea88e654c0568e56b7

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	10	10	
Informational	4	4	
Minor	4	4	
Medium	0	0	
Major	1	1	

Critical	1	1	

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

Code scope sees **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 of Findings

This report has been commissioned by **Interest Protocol** to identify any potential issues and vulnerabilities in the source code of the **DEX** 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 have identified 10 issues of varying severity, listed below.

ID	Title	Severity	Status
COR–01	Redundant Code	Informational	Fixed
COR–02	Unexpected Writing	Informational	Fixed
COR–03	Writing Style	Informational	Fixed
MTH–04	Divide Zero	Minor	Fixed
COR–05	Missing Event	Minor	Fixed
MCF–06	Zero Allocation Point Value	Minor	Fixed

COR-07	Wrong Liquidity	Major	Fixed
COR-08	Flash Loan Manipulate	Critical	Fixed
COR-09	Admin Authorization	Minor	Fixed
COR-10	Minimum Liquidity	Informational	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the `Interest Protocol DEX` SmartContract：

Admin

- Admin can change the fee receive account through `dex::core::update_fee_to`
- Admin can change its cap through `dex::core::transfer_admin_cap`
- Admin can add a new pool in the farm through `dex::master_chef::add_pool`
- Admin can update the IPX through `dex::master_chef::update_ipx_per_ms`
- Admin can update the allocation points through `dex::master_chef::set_allocation_points`
- Admin can change the farm admin cap through `dex::master_chef::transfer_admin`
- Admin can change the admin of IPX through `ipx::ipx::transfer_admin`
- Admin can remove a minter of IPX through `ipx::ipx::remove_minter`
- Admin can add a minter of IPX through `ipx::ipx::add_minter`

User

- User can create a volatile pool through `dex::interface::create_v_pool`
- User can create a stable pool through `dex::interface::create_s_pool`
- User can swap coins in the pool through `dex::interface::swap_x` | `dex::interface::swap_y` | `dex::interface::one_hop_swap` | `dex::interface::two_hop_swap`
- User can add liquidity in the pool through `dex::interface::add_liquidity`
- User can remove liquidity in the pool through `dex::interface::remove_liquidity`
- User can add stake in the farm through `dex::interface::stake`
- User can remove stake in the farm through `dex::interface::unstake`
- User can get reward through `dex::interface::get_rewards`

- User can update pool through `dex::interface::update_pool` | `dex::interface::update_all_pools`
- User can burn the IPX owned by itself through `dex::interface::burn_ipx`
- User can transfer IPX through `ipx::ipx::transfer`

4 Findings

COR-01 Redundant Code

Severity: Informational

Status: Fixed

Descriptions: The functions `creat_pool_s` and `creat_pool_v` share similar logic, it would be better to wrap them in a common function.

Code Location: dex/core.move #L 163/257

Suggestion: Shall pack the code that shares similar functionality to a single function, and call it when used.

Resolution: The client followed our suggestion and fixed the issue.

COR-02 Unexpected Writing

Severity: Informational

Status: Fixed

Descriptions: There are some mistakes in the comments, it would be better to correct them.

Code Location: dex/core.move #L 1190

Suggestion: Shall be `return (u64, u64, u64) (balance_x, balance_y, lp_coin_supply)` .

Resolution: The client followed our suggestion and fixed the issue.

COR-03 Writing Style

Severity: Minor

Status: Fixed

Descriptions: The `assert` statement should be put as front as possible to make the statement more clearly and may save some gas.

Code Location: dex/core.move #L 1039

Suggestion: Move the `assert` statement to the front.

Resolution: The client followed our suggestion and fixed the issue.

MTH-04 Divide Zero

Severity: Minor
Status: Fixed
Descriptions: The mathematical functions in `library::math` do not check for division by zero, which may lead to unexpected abort.

Code Location: library/math.move #L 14-40

Suggestion: Add an assert statement to make sure `y` is not zero.

Resolution: The client followed our suggestion and fixed the issue, and add the assertions.

IPX-05 Missing Event

Severity: Minor
Status: Fixed
Descriptions: Some essential functions are missing the related event, including `update_fee_to`, `transfer_admin_cap`, which may lead to the deflect in the log.

Code Location: dex/core.move #L 1166/1179

Suggestion: Add a new event structure and statement.

Resolution: The client followed our suggestion and fixed the issue, and add the new event.

MCF-06 Zero Allocation Point Value

Severity: Minor
Status: Fixed
Descriptions: The `allocation_points` is better not to be zero when the administrator adds a new pool, otherwise it can be meaningless.

Code Location: dex/master-chef.move #L 681

Suggestion: Add an assert statement `allocation_points != 0` to the start of the function.

Resolution: The client followed our suggestion and fixed the issue and add an assert statement.

COR-07 Wrong Liquidity

Severity: **Major**

Status: **Fixed**

Descriptions: When adding liquidity to the swap pool, there is no statement that checks the amount of `coin_x` and `coin_y` are satisfied with the proportion to `reserve_x` and `reserve_y` . Furthermore, there is no refund for users when they send exceeding numbers of `coin_x` and `coin_y` .

Code Location: dex/core.move #L 391

Suggestion: Check the value of `coin_x` and `coin_y` in an assert statement to make sure they are in proportion to `reserve_x` and `reserve_y` , or refund the extra `coin_x` or `coin_y` to the user.

Resolution: The client followed our suggestion and fixed the issue.

COR-08 Flash Loan Manipulate

Severity: **Critical**

Status: **Fixed**

Descriptions: In the flash loan, the adversary can drain the tokens in the pool by lending a large amount of a single coin to significantly unbalance the pool. We take pool `eth, usdt` as an example. Assume the pool contains `<1 eth, 1000 usdt>` , the attacker first lends 0.9 eth, which makes the pool `<0.1 eth, 1000 usdt>` , then the attacker called `swap_x_token` of the pool `eth, usdt` and uses `0.1 eth` to swap `492 usdt` , and then repay the flash loan. The result is the attacker has gained great profit while the pool suffered losses.

Code Location: dex/core.move #L 807

Suggestion: Add an assert statement in the `repay_flash_loan` to make sure the k always increases after the repayment.

Resolution: The client followed our suggestion and add pool locker and k assertion.

COR-9 Admin Authorization

Severity: Minor

Status: Fixed

Descriptions: We noticed that the admin cap is allowed to be transferred to `@0x0`, which means it could be deactivated.

Code Location: dex/core.move #L 1179

Suggestion: Restrict the admin cap.

Resolution: The client followed our suggestion and fixed the issue, and assert the admin cap should not be transferred to @0x0.

COR-10 Minimum Liquidity

Severity: Informational

Status: Fixed

Descriptions: We noticed that the minimum liquidity volatile and stable pools are set to constant 10. This value should be larger to make sure the pool won't be maliciously unbalanced.

Code Location: dex/core.move #L 286

Suggestion: Raise the minimum liquidity value a little bit.

Resolution: The client followed our suggestion and fixed the issue, and raised it to 100.

Appendix 1

Issue Level

- **Informational:** Informational items 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.

- **Discussion** issues are uncertain properties, which need further communication with the designer.

Issue Status

- **Fixed:** The issue has been 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

Securing the Move Ecosystem

https://twitter.com/movebit_

contact@movebit.xyz