Bay Swap Corporation

Smart Contract

Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

05/30/2023

Bay Swap Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	Bay Swap is a DEX (Decentralized Exchange) built on SUI Blockchain.
Type	DeFi
Auditors	MoveBit
Timeline	May 18, 2023 – May 29, 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/BaySwap/dex-core
Commits	5d6c905ffba791fe7df06741cea56f050878aa03 de31b34e138f05210b765886867e46aaca6a9fdf

1.2 Files in Scope

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

ID	**Files**	**SHA-1 Hash**
LMO	sources/limit_order.move	0665ae29a16d3e388e7331b1afb52911d44fea76
RT	sources/router.move	3ab31172a2655930dae4ca2c57ee438ae83187ff

HP	sources/helper.move	e980a2647c1b706c4cb236f5be15a3be5b903d15
STB	sources/stable_curve.move	b3956c0eb6a7089a1cea6e4efc3162cb69a21eb1
LMOE	sources/limit_order_entry.move	1d4edef4c0536f30c301143cc45f52e17eed4ba0
CV	sources/curves.move	1a9f47e45f5b067b94272ea1064aa57930a1d54e
ET	sources/entry.move	18138172b99ab51814fe39fef97395b387d655aa
MT	sources/math.move	4c0eef97e163e20e229f87ecb082250ba8e25e5a
LQP	sources/liquid_pool.move	dcef5025c8c15b232a2ba5cb8040f4efa386666c

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	9	7	2
Informational			
Minor	7	5	2
Medium	2	2	
Major			
Critical			

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 **Bay Swap** to identify any potential issues and vulnerabilities in the source code of the **Bay Swap** 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 **9** issues of varying severity, listed below.

ID	Title	Severity	Status
LQP-01	Lack of Minimum Liquidity	Medium	Fixed
LQP-02	Incorrect Precision of `DAO_FEE_SCALE`	Medium	Fixed
LQP-03	Unused Struct	Minor	Fixed
LQP-04	Gas Optimization	Minor	Acknowledged
RT-05	Redundant Assert	Minor	Fixed
RT-06	Inconsistent Code	Minor	Fixed
MT-07	Incorrect Return Type	Minor	Acknowledged
HP-08	Unused Constant	Minor	Fixed
LMO-09	Inconsistent Code	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the **Bay Swap** Smart Contract:

Admin

- Admin can set the fee and the fee receiver address through `set_dao_fee()` , `set_stable_fee()` , `set_uncorrelated_fee()` , `set_pool_fee()` , `set_pool_dao_fee()` , `set_dao_account()` .
- Admin can execute the swap order through `execute_swap_order()` .

User

- User can register a pool through `register_pool()` .
- User can register a pool and add liquidity to the pool through `register_pool_and_add_liquidity()` .
- User can add liquidity to the pool and remove the liquidity from the pool through `add_liquidity()` and `remove_liquidity()` .
- User can add liquidity by using only one type of coin through `zap_in_x()` , `zap_in_y()` .
- User can swap their coins through `swap_from_x_to_y()` , `swap_from_y_to_x()` .
- User can place their swap order through `place_swap_order_from_x_to_y()` , `place_swap_order_from_y_to_x()` .
- User can cancel their swap order through `cancel_swap_order()` .

4 Findings

LQP-01 Lack of Minimum Liquidity

Severity: Medium

Status: Fixed
Code Location: sources/liquid_pool.move#L244.
Descriptions: In the `mint` function no minimum liquidity was added when adding liquidity for the first time.

Suggestion: Add the minimum liquidity and lock it in the pool, when adding liquidity for the first time.

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

LQP-02 Incorrect Precision of `DAO_FEE_SCALE`

Severity: Medium

Status: Fixed

Code Location: sources/liquid_pool.move#L64.

Descriptions: In `helper.move`, `DAO_FEE` ranges from 0 to 1000, so the `DAO_FEE_SCALE` in `liquid_pool` should be 10000 rather than 100.

Suggestion: It is recommended to modify the `DAO_FEE_SCALE` to 10000.

Resolution: The client modified the range of `DAO_FEE` and fixed this issue.

LQP-03 Unused Struct

Severity: Minor

Status: Fixed

Code Location: sources/liquid_pool.move#L170.

Descriptions: There are some unused struct `EventSetPoolDaoFee`.

Suggestion: It is recommended to remove the unused struct if there's no further design.

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

LQP-04 Gas Optimization

Severity: Minor

Status: Acknowledged

Code Location: sources/liquid_pool.move#L689-701.

Descriptions: Using `==` to judge the boolean value will increase gas consumption.

Suggestion: It is recommended to modify as `assert!(!pool.is_locked, EPoolIsLocked)`.

Resolution: The client followed our suggestion and fixed the issue, but there are some same issues have not been fixed in the `liquid_pool.move` L702-L710.

RT-05 Redundant Assert

Severity: Minor

Status: Fixed

Code Location: sources/router.move#L53, L56.

Descriptions: `curves::assert_valid_curve<Curve>()` and `assert!(helper::is_sorted<X, Y>() == true, EWrongCoinOrder)` is already present in the `liquidity_pool::register` function and is redundant here.

Suggestion: It is recommended to remove the redundant assert to reduce gas consumption.

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

RT-06 Inconsistent Code

Severity: Minor

Status: Fixed

Code Location: router.move#L168.

Descriptions: The function `swap_coin_x_for_coin_y_unchecked` is Inconsistent with `swap_coin_y_for_coin_x_unchecked` , there is a `curves::assert_valid_curve<Curve>()` in it.

Suggestion: It is recommended to modify the code of `swap_coin_x_for_coin_y_unchecked` .

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

MT-07 Incorrect Return Type

Severity: Minor

Status: Acknowledged

Code Location: sources/math.move#L4.

Descriptions: The return type of the `mul_div_u128()` function should be `u128` , but is `u64` here.

Suggestion: It is recommended to modify the return type to `u128` .

Resolution: The client confirmed the issue and replied to the data truncation is never happen.

HP-08 Unused Constant

Severity: Minor

Status: Fixed

Code Location: sources/helper.move, sources/entry.move, sources/limit_order_entry.move, sources/limit_order.move, sources/liquidity_pool.move, sources/router.move.

Descriptions: There are some unused constants in the contracts.

For example:

`helper.move` : `SYMBOL_PREFIX_LENGTH` , `EInvalidDenominator` , `EUnreachableCode` ;

`entry.move` : `EWrongCoinOrder` ;

`limit_order_entry.move` : `EPassedCoinLessThanSwapAmount` ;

`limit_order.move` : `EInvalidExecutionFee` ;

`liquidity_pool.move` : `EUnreachableCode` ;

`router.move` : `EPassedCoinLessThanSwapAmount` , `EPassedCoinXLessThanAddLiquidAmount` , `EPassedCoinYLessThanAddLiquidAmount` , `EInvalidSetOfCoinX` , `EInvalidSetOfCoinY` , `EInvalidTwoA` .

Suggestion: It is recommended to remove the unused constant if there's no further design.

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

LMO-09 Inconsistent Code

Severity: Minor

Status: Fixed

Code Location: sources/limit_order.move#L172.

Descriptions: The function `place_swap_order_from_y_to_x` is Inconsistent with `place_swap_order_from_x_to_y` in the `!is_existed_orders_for_pool<X, Y, Curve>(storage)` branch.

Suggestion: It is recommended to modify the code of `place_swap_order_from_y_to_x` .

Resolution: The client followed our suggestion and fixed the 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.