Pontem Liquidity Swap Formal Verification

Audit Report

contact@movebit.xyz

https://twitter.com/movebit_

Sat Apr 20 2024

Pontem Liquidity Swap Formal Verification Audit Report

1 Executive Summary

1.1 Project Information

Description	An Concentrate liquidity Based AMM Swap.
Type	DeFi
Auditors	MoveBit
Timeline	Sun Feb 25 2024 - Sat Apr 20 2024
Languages	Move
Platform	Aptos
Methods	Architecture Review, Formal Verification
Source Code	https://github.com/pontem-network/liquidswap_v1
Commits	de53c626e94fe185f60debd404c1f1b33b827581

1.2 Files in Scope

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

ID	File	SHA-1 Hash
EME	sources/emergency.move	420f6f7eda92bf92dead58de263b5f73a8abbe81
LTO	sources/lb_token.move	0d0296f6659a74f18e6ad5a8a47e88ec9311ccaa
CON	sources/config.move	885b838ef8bee78d7df9360623ee4f89bf98f1f0
POO	sources/pool.move	4c6deebf98e4edd827acb4c72dac3a5fba38ea57
ORA	sources/oracle.move	f52ec8c11cc404a19173acbab108fecd0a6b923e
TRE	sources/treasury.move	86054e2bf173ea1176f1598cdfbb1ba6ffff9bae

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	2	1	1
Informational	0	0	0
Minor	1	0	1
Medium	0	0	0
Major	1	1	0
Critical	0	0	0

1.4 MoveBit Audit Breakdown

MoveBit aims to confirm the soundness of the formal verification by concept review, property discovery, gathering, verification, and compliance with the auditing techniques. Possible specifications included (but are not limited to):

- Assertions
- Aborts conditions
- Return value confirmation
- Invariant
- High-level properties
- Best practice

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Specification"** strategy to perform a complete formal verification to ensure the completeness of the entire process. The main entrance and scope of verification are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The specification of the scope can be mainly separated into these categories:

(1) Local Property:

Including the possible aborts conditions, requirements, and expected global state change.

(2) High-level Property:

Including the feature that is highly relevant to the project. The detail of the properties can be found in section 3.

(3) Helper Function:

Including the function that is used to obtain the value during the specification, like ghost variable and opaque function.
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 Pontem to identify any potential issues and vulnerabilities in the source code of the Liquidity Swap V1 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 2 issues of varying severity, listed below.

ID	Title	Severity	Status
POO-1	Unexpected Coin Value (Property 2 Not Hold)	Major	Fixed
POO-2	Unexpected Pool Status (Property 6 Not Hold)	Minor	Acknowledged

3 Specification Process

Here are the relevant actors with their respective abilities within the Liquidity Swap V1 Smart Contract :
This section listed all the modules/functions we verified and demonstrated the details.
Overall, we catalog the specification into the `local properties` and `high-level properties` .

Local Properties

Module	Function	Status	Comment
lb_token.move	create_token_collection	Verified	The other functions in the lb_token are almost helper functions, and we use opaque to leverage these functions.
Oracle.move	create_oracle	Verified	-
Oracle.move	increase_oracle_length	Verified	-
Oracle.move	update_oracle	Verified	It does contain bv to int calculation, and it may lead to prover error when called by '`swap_inner`'. Solved with a helper spec.
Oracle.move	get_oracle_lengths	Verified	-

Module	Function	Status	Comment
Oracle.move	get_oracle_active_id	Verified	-
Oracle.move	get_sample_data_unsafe	Verified	-
Oracle.move	get_sample_data	Verified	-
Oracle.move	get_sample_timestamp_and_lifetime	Verified	-
Oracle.move	get_sample_cummulative_data	Verified	-
Oracle.move	check_sample_has_filled	Verified	-
Oracle.move	sample_exists	Verified	-
Oracle.move	get_max_oracle_length	Verified	-
Oracle.move	binary_search	Partially Verified	-
Pool.move	initialize	Verified	-

Module	Function	Status	Comment
Pool.move	register_pool	Not Verified	It occurs an unknown prover error 'task panicked'. Solved with opaque.
Pool.move	update_static_fee_parameters	Verified	-
Pool.move	update_fees_configuration	Verified	-
Pool.move	swap_inner	Verified	It occurs bv to uint conversion, which the move prover has not fully supported yet. Solved with a helper function.
Pool.move	flashloan	Verified	Partially
Pool.move	pay_flashloan	Verified	Partially
Pool.move	mint	Partially Verified	It occurs an unknown prover error 'task panicked' in mint_bins, mint function. Solved with opaque.

Module	Function	Status	Comment
Pool.move	update_bin	Verified	-
Pool.move	burn	Partially Verified	It occurs an unknown prover error 'task panicked'. Solved with opaque.
Pool.move	unwrap_liq_nft	Verified	-
Pool.move	burn_bin_liquidity	Verified	-
Pool.move	is_coin_sorted_inner	Verified	-
Treasury.move	register	Verified	-
Treasury.move	deposit	Verified	-
Treasury.move	withdraw	Verified	-
Treasury.move	get_balance	Verified	-

Module	Function	Status	Comment
Treasury.move	exists_at	Verified	-

High-level Properties

No.	Property	Criticality	Implementation	Enforcement	Status
1	The NFT produced by the lp_token should be unique.	Critical	It should be verified in the Aptos framework::token	Enforced by aptos_token module	Verified
2	The pool must contain both of the two tokens during the swap.	Major	The `coin_x` and `coin_y` should both be zero or both be non-zero.	Formally Specified: Struct `pool`	Manual Checked
3	Each swap should only take one kind of coin at a time to another coin.	Major	The `swap_inner` function should never have `x` and `y` to be both non-zero.	Formally Specified: `swap_inner`	Verified
4	When a swap trade exceeds a tick, the liquidity remaining in this tick should	Critical	The `reserves_x` or `reserves_y` of the current bin_step should be zero when active_bin_id changed.	Formally Specified: `swap_inner`	Verified

No.	Property	Criticality	Implementation	Enforcement	Status
	contain only one kind of coin.				
5	When any LP deposits/withdrawn token to a pool, the reserved coin in the target bin should increase/decrease.	Major	It should be verified by: (1) Each step in the `update_bin` should correctly update the value of the bin and return the correct coin value/type. (2) After mint\burn, the value of `pool.coin_x` or `pool.coin_y` should increase\decrease.	Formally Specified: `mint`, `mint_bin`, `burn`, `update_bin`	Manual Checked
6	All coins should be processed within the pre-setting position.	Major	It should be verified by: (1) The swap should be started at the `active_bin_id`. (2) The mint should be deposited in the bin vector.	Formally Specified: `swap_inner`, `mint`, `burn`	Verified
7	When any LP deposits/withdrawn coins to a pool, while the bin is higher or lower than the active bin, it shall only need to provide one kind of coin.	Major	It should be verified by: The `added_x` and `added_y` in the mint function are zero when dealing with the `bin != active_bin_id`.	Formally Specified: `mint`, `mint_bin`	Verified

No.	Property	Criticality	Implementation	Enforcement	Status
8	The treasury received the correct amount of fee.	Major	The `treasury` received the fee that is equal to the setting value.	Formally Specified: `deposit`	Verified
9	The flash loan must be paid back with the full amount, meanwhile, the pool cannot be modified.	Critical	It should be verified by: (1) When the pool is locked, it should not be able to lend flashloan. (2) The `flashloan` struct should have no abilities. (3) There should be only one flashloan during the lending process. (4) The amount when lent/paid should be correct.	Formally Specified: `Flashloan` , `Pay_Flashlon`	Verified
10	After a swap trade, the k of the pool should be rising due to the trading fee.	Major	When the trading fee is not zero, the treasury should grow after trading.	Formally Specified: `swap_inner`	Verified
11	When a pool is locked, there should be no operation happened.	Major	The swap should abort when the pool is locked.	Formally Specified: `swap_inner`	Verified

4 Findings

POO-1 Unexpected Coin Value (Property 2 Not Hold)

Severity: Major

Status: Fixed

Code Location:

sources/pool.move#108-139

Descriptions:

The **property 2** requires:

- The coin_x and coin_y of a pool should both be zero (at its initial state) or both be non-zero.
- The coin_x and coin_y after any operation should not be zero for a non-empty pool.

However, a series of functions has violated this property. We denoted the `pre_x and pre_y` as the value of coin_x and coin_y before the execution, and `post_x and post_y` as the value after the execution. When `pre_x != 0 && pre_y !=0`, they allowed the value after the execution to be zero, which is `post_x == 0 || post_y == 0`. These functions include: `swap_inner, mint, burn, flashloan, pay_flashloan`

We believe a swap pool should not allow the situation, as the concentrated liquidity should follow the `k = x*y` when considering the sum of bin_steps. Otherwise, it may lead to unexpected errors.

However, for `flashloan` and `pay_flashloan`, it may not be necessary to ensure this property.

Suggestion:

It is recommended to implement assertions to the functions `swap_inner, mint, burn` as follows:

`assert!(coin::value(pool.coin_x) == 0 || coin::value(pool.coin_y) == 0, ERROR_SHOULD_NOT_EMPTY);`

Resolution:

The development team has confirmed and made certain modifications to ensure this situation will not happen.

POO-2 Unexpected Pool Status (Property 6 Not Hold)

Severity: Minor

Status: Acknowledged

Code Location:

sources/pool.move#1031-1098

Descriptions:

The **property 6** requires:

- Each step in the update_bin should correctly update the value of the bin and return the correct coin value/type.
- After minting, the pool.coin_x or pool.coin_y should rise.

During the specification, we found the state of the `pool.coin_x` and `pool.coin_y` had been reassigned after the loop in the `mint_bin` function, and this reassign of the `pool` led to the violation of this property. These functions include:

`mint_bin, update_bin`

The reassigned `pool` shows the situation that, none of the `coin_x` and `coin_y` are increase after the mint.

Suggestion:

Make sure during the execution of the `mint_bin`, `mint`, and `update_bin` functions, the state of `pool.coin_x` and `pool.coin_y` will not be changed unexpectedly.

As a reminder, it can be a move-prover error and may not indicate any deflect in the source code.

Resolution:

The development team has confirmed that the state of `pool.coin_x` and `pool.coin_y` is not modified during the loop inside functions `mint_bin` and `update_bin`.

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