YuzuDex CLMM

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Fri Jan 17 2025

YuzuDex CLMM Audit Report

1 Executive Summary

1.1 Project Information

Description	A Clmm Dex.
Type	DeFi
Auditors	MoveBit
Timeline	Tue Dec 31 2024 - Fri Jan 17 2025
Languages	Move
Platform	Movement
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/YuzuDEX/CLMM-Core https://github.com/YuzuDEX/MoveFun https://github.com/YuzuFinance/move-fun
Commits	https://github.com/YuzuDEX/CLMM-Core: c653809a05ce053941fc1af6a9e2a9fefad7dc32 c1818ce6527d3428c4818e1376d782c27d9b19f1 a0fea27c5607aeb4205535b7bd4a6c6348f3090a https://github.com/YuzuDEX/MoveFun: cf76eefc825170f805f458791682f2fbeaea82f6 https://github.com/YuzuFinance/move-fun: b7e3b833084c5c80808d021437048832370575f5 8a3d1c6c37bb1568a2743b9bff4ed88c3bf76896

1.2 Files in Scope

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

ID	File	SHA-1 Hash
PNM	sources/position_nft_manager.move	45c6de2baf8ce73b6a9aea53b5beba562c0a27bb
ROU	sources/router.move	df8f6e323157187d4513272f41ce66794125fd10
FPO	sources/libs/fixed_point.move	8eb5d529e60daa3e2aebfa8c6c72d25d1bac6c2c
SMA	sources/libs/swap_math.move	3797e86ff375ee4c3125d16809f9094f58e95e6d
UMA	sources/libs/unsafe_math.move	e7ec31c78947697d9dce9dbcc2bec7ad9473e8c0
MAT	sources/libs/math.move	dbfd328a10b86003c1b2d35708b12025ea0bedde
LMA	sources/libs/liquidity_math.move	f55677ba1a5ad56923d394e374e9e61710cde7c6
FHE	sources/libs/fa_helper.move	bbf01a41c32457b59bc5b07cfb59360710076029
CHE	sources/libs/coin_helper.move	09309bfa246100bf71e658fde89ab2b2de16daa6
TBI	sources/libs/tick_bitmap.move	e1afb6a6e68654fec9e638407ae941e181386fe2
SPM	sources/libs/sqrt_price_math.move	fbdc4321c3d934483845f67749e9a4e67ea41c4e

I12	sources/libs/i128.move	14c36cac882fa9060bb229ad94120ff75924f99c
TIC	sources/libs/tick.move	472ca9574e770646a805409e6683e79d8f8b85a5
BMA	sources/libs/bit_math.move	6f65716b15c19fed1e24f7970905c06f36e02451
TMA	sources/libs/tick_math.move	6a8202511bdfd6c37b47be68980040b94df71695
EME	sources/emergency.move	7e56e5a6a4d5aea771beecec02d8d49926485c9e
SCR	sources/scripts.move	c8034bc8f645f84ab7b114ac44f6135d19cf21fc
LPO	sources/liquidity_pool.move	f02a36cb8bbad8feae44696d46060f030098dfc4
CON	sources/config.move	05df1675cdcec73e76ec31c6686966515cbcf41f
FTI	sources/fee_tier.move	1d5be178e27499b3cd40b0c90b574af287aa7cb6
RMA	sources/reward_manager.move	43c3c3e9897c7f46474f190fcbda3243d2aebb75

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	9	6	3
Informational	0	0	0
Minor	5	3	2
Medium	1	0	1
Major	3	3	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(Optional)

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 YuzuDex to identify any potential issues and vulnerabilities in the source code of the YuzuDex CLMM 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
CON-1	Missing Event Emission for Critical State Change	Minor	Acknowledged
FTI-1	Missing Validation for Zero `tick_spacing`	Minor	Acknowledged
LPO-1	Lack of Reward Check When Closing Position	Major	Fixed
LPO-2	Insufficient Reward Validation in `update_reward_emissions`	Medium	Acknowledged
PNM-1	Redundant Retrieval of Tick Information	Minor	Fixed
RMA-1	Incorrect Parameter Type for `new_reward_manager`	Major	Fixed
RMA-2	Incorrect Function Call to Update Reward Manager	Major	Fixed
RMA-3	Missing Existence Check for `token_metadata`	Minor	Fixed
ROU-1	Unused Constants	Minor	Fixed

3 Participant Process

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

Admin

- The admin can assert their pool admin privileges through `assert_pool_admin()`.
- The admin can assert their reward admin privileges through `assert_reward_admin()`.
- The admin can assert their emergency admin privileges through `assert_emergency_admin()`.
- The admin can set the pool admin through `set_pool_admin()`.
- The admin can set the reward admin through `set_reward_admin()`.
- The admin can set the emergency admin through `set_emergency_admin()`.
- The admin can set the protocol fee through `set_protocol_fee()`.
- The admin can pause all operations through `pause()`.
- The admin can resume all operations through `resume()`.
- The admin can check if the system is paused through `is_emergency()`.
- The admin can ensure the system is not paused through `assert_no_emergency()`.
- The admin can check if the system is permanently disabled through `is_disabled()`.
- The admin can permanently disable the system through `disable_forever()`.
- The admin can add a new fee tier with specific tick spacing through `add_fee_tier()`.
- The admin can delete an existing fee tier through `delete_fee_tier()`.
- The admin can whitelist a reward token through `whitelist_reward_token()`.
- The admin can whitelist a reward token by coin type through `whitelist_reward_token_by_coin()`.
- The admin can unwhitelist a reward token through `unwhitelist_reward_token()`.
- The admin can unwhitelist a reward token by coin type through `unwhitelist_reward_token_by_coin()`.

- The admin can initialize a pool reward through `initialize_pool_reward()`.
- The admin can initialize a pool reward by coin type through `initialize_pool_reward_coin()`.
- The admin can update a reward manager through `update_reward_manager()`.
- The admin can update reward emissions through `update_reward_emissions()`.
- The admin can add a reward to the pool through `add_reward()`.
- The admin can add a reward to the pool by coin type through `add_reward_coin()`.
- The admin can remove a reward from the pool through `remove_reward()`.

User

- The user can mint a new liquidity position through `mint()`.
- The user can increase liquidity for an existing position through `increase_liquidity()`.
- The user can decrease liquidity for an existing position through `decrease_liquidity()`.
- The user can collect fees from a position through `collect_fee()`.
- The user can collect rewards from a position through `collect_reward()`.
- The user can close (burn) a liquidity position through `burn()`.
- The user can swap tokens through `swap()`, which allows the user to exchange tokens and receive a receipt.
- The user can pay the swap receipt through `pay_swap()`, which enables the user to repay the amount equivalent to their swap receipt.

4 Findings

CON-1 Missing Event Emission for Critical State Change

Severity: Minor

Status: Acknowledged

Code Location:

sources/config.move#50,56,62,68

Descriptions:

The `set_pool_admin()` function updates a critical state variable, pool_admin, without emitting an event to log this change. This omission reduces transparency and makes it difficult for off-chain services or users to track state changes, potentially leading to reduced trust and operational inefficiencies in monitoring contract activities. Similar functions include:

1. `set_pool_admin()` .
2. `set_reward_admin()` .
3. `set_emergency_admin()` .
4. `set_protocol_fee()` .

Suggestion:

Emit an event whenever the `pool_admin ()` variable is updated. Define an event, such as `PoolAdminUpdated` , and include the old and new admin addresses. This will improve traceability and allow external observers to monitor and verify the changes efficiently.

FTI-1 Missing Validation for Zero `tick_spacing`

Severity: Minor

Status: Acknowledged

Code Location:

sources/fee_tier.move#52

Descriptions:

The `add_fee_tier()` function does not validate whether the `tick_spacing` parameter is greater than zero. If `tick_spacing` is set to zero, it could lead to undefined behavior or runtime errors in subsequent operations that rely on this parameter.

Suggestion:

Add an assertion to ensure that `tick_spacing` is greater than zero before proceeding with the rest of the function logic.

LPO-1 Lack of Reward Check When Closing Position

Severity: Major

Status: Fixed

Code Location:

sources/liquidity_pool.move#366

Descriptions:

The `close_position()` function does not check whether the position has any rewards left (`tokens_owed_0` and `tokens_owed_1`) before closing the position. Users may be entitled to rewards for providing liquidity, which are accumulated in `tokens_owed_0` and `tokens_owed_1` during swaps. If the position is closed without verifying the rewards, the user may lose out on their earned rewards because those rewards are tied to the position's data, which will no longer be available after closure.

Suggestion:

Introduce a check to ensure that any rewards are distributed to the user before the position is closed.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

LPO-2 Insufficient Reward Validation in `update_reward_emissions`

Severity: Medium

Status: Acknowledged

Code Location:

sources/liquidity_pool.move#816

Descriptions:

The `update_reward_emissions()` function allows updating the `emissions_per_second` for rewards without verifying if the pool has sufficient rewards to sustain the new emission rate. This could result in scenarios where the rewards are depleted prematurely, leading to user dissatisfaction and potential reputational damage for the protocol.

Suggestion:

Introduce a check to ensure that the total rewards available in the pool are sufficient to support the specified `emissions_per_second` over a reasonable time frame. Calculate the total rewards required for the new emission rate and compare it to the available balance before updating. Example:

```
assert!(
    reward_info.remaining_reward >= emissions_per_second *
MINIMUM_EMISSION_DURATION,
    E_INSUFFICIENT_REWARDS
);
```

PNM-1 Redundant Retrieval of Tick Information

Severity: Minor

Status: Fixed

Code Location:

sources/position_nft_manager.move#81

Descriptions:

The code redundantly retrieves the tick information (tick_lower and tick_upper) using liquidity_pool::get_position_info after calling `liquidity_pool::open_position()`. Since the tick values are already known and explicitly provided when creating the position, this retrieval is unnecessary and adds extra computational overhead, which could negatively impact performance.

Suggestion:

Remove the redundant call to `liquidity_pool::get_position_info()` for tick information. Instead, reuse the tick_lower and tick_upper values that are already available from the parameters passed to `liquidity_pool::open_position`.

```
let position_id = liquidity_pool::open_position(nft_manager_signer, pool, tick_lower,
tick_upper);

let (_, _, current_sqrt_price, _, _, _, _) = liquidity_pool::get_pool_info(pool);

// Remove this redundant call:
// let (_, tick_lower, tick_upper, _, _, ) = liquidity_pool::get_position_info(
//     signer::address_of(nft_manager_signer),
//     pool,
//     position_id,
// );

// Use the tick_lower and tick_upper values directly:
```

Resolution:

This issue has been fixed. The client has adopted our suggestions.

RMA-1 Incorrect Parameter Type for `new_reward_manager`

Severity: Major

Status: Fixed

Code Location:

sources/reward_manager.move#80

Descriptions:

The parameter `new_reward_manager` is defined as a u64, but it should be of type address, as it represents the address of the new reward manager. This mismatch in types can lead to unexpected behavior or runtime errors when the function is called.

Suggestion:

Change the type of the `new_reward_manager` parameter to address to ensure it correctly represents the reward manager's address.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

RMA-2 Incorrect Function Call to Update Reward Manager

Severity: Major

Status: Fixed

Code Location:

sources/reward_manager.move#86

Descriptions:

The function `liquidity_pool::update_reward_emissions()` is called instead of `liquidity_pool::update_reward_manager()`. This is a logical error because the intended functionality is to update the reward manager, not the reward emissions.

Suggestion:

Replace the call to `liquidity_pool::update_reward_emissions()` with the correct function, `liquidity_pool::update_reward_manager()`, to properly update the reward manager.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

RMA-3 Missing Existence Check for `token_metadata`

Severity: Minor

Status: Fixed

Code Location:

sources/reward_manager.move#52

Descriptions:

The `unwhitelist_reward_token()` function directly removes token_metadata from the `whitelistedRewardTokens` table without verifying whether the `token_metadata` object exists in the table. If `token_metadata` does not exist, this could result in an error or unexpected behavior, such as an abrupt transaction failure.

Suggestion:

Add a check to ensure that `token_metadata` exists in the `whitelistedRewardTokens` table before attempting to remove it.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

ROU-1 Unused Constants

Severity: Minor

Status: Fixed

Code Location:

sources/router.move#23,25

Descriptions:

The constants `E_LIQUIDITY_AMOUNT_LESS_THAN_EXPECTED` and `E_NOT_ENOUGH_AMOUNT_OUT_WHEN_DECREASING_LIQUIDITY` are defined but not used anywhere in the code. Defining unused constants increases the codebase's complexity without adding value, potentially causing confusion and making maintenance harder.

Suggestion:

It is recommended to remove these unused error codes.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

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.