Ferra CLMM

Audit Report

MOVEBIT

contact@bitslab.xyz

https://twitter.com/movebit_

Fri Sep 26 2025

Ferra CLMM Audit Report

1 Executive Summary

1.1 Project Information

Description	This is a Concentrated Liquidity Market Maker (CLMM) implementation on the Sui blockchain, providing efficient automated market making with concentrated liquidity positions.
Type	DEX
Auditors	MoveBit
Timeline	Tue Aug 05 2025 - Mon Aug 25 2025
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Ferra-Labs/ferra-clmm
Commits	9c2f3b6df009f892c97afc12d3fc579a39d04eb8 07a29d10525e75d35bd4791d7160528214dea5de d23097aebde760ea40b21710f7c7c2385bd303af f5a09aca59cd32893b71e396938cf7628c2f5c6a 86d2deb4634c7ab9b46f99894ed9887b1d989bfe 7b66118e00201de2e91d5a99f93e77b1f5049324

1.2 Files in Scope

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

ID	File	SHA-1 Hash
ACL	sources/acl.move	337886a9fb66058c0409e5a5a7f92a4e5e392897
FAC	sources/factory.move	3aa6809ccdcaa1821cf79d31a1b4c8d3ad56bf6e
POS	sources/position.move	959b483e05c1936c32f5b6545d193d0d8f952f74
REW	sources/rewarder.move	c128ab36f8e2c99ecf5a69bd6fe21218abc82a68
CON	sources/config.move	c946a3d52c61498aa9ce9769d5455cb0c342ee7e
POO	sources/pool.move	96a55788d305357415dfe6e94afdeb22a4a2cfed
UTI	sources/utils.move	22c2f32fe02418e858eae6bbf76daf9609b2addb
CMA	sources/math/clmm_math.move	cb67ee2eace9d5b9ad03dc26b884e049e8bfe997
TMA	sources/math/tick_math.move	aa89330ecee04527f55b2922428d93846b27d73f
TIC	sources/tick.move	d6480476c749df6d91eb51a0517797a4c823e43c

ACL	sources/acl.move	18c00b107263e8c5ad46377b86f513abe1b03bee
FAC	sources/factory.move	c13092618deae69586c139f48731a2576dd1cdb0
POS	sources/position.move	56a660683fd29316ebd870d5230dcd5f90cdba18
REW	sources/rewarder.move	f48fafcf4badcac10ca624b43211a8fe3f316d3c
CON	sources/config.move	11f02d8642208ea6d632a8faaf8c459423c6a172
POO	sources/pool.move	169601304215e6c1ed65fb9e42a64f955a9804ad
UTI	sources/utils.move	8b176a48c56aa5ab4ceee1a30a4932b2d3580e5e
TIC	sources/tick.move	a4fc9139087486483bca3da33a038d2b892e0056

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	7	6	1
Informational	2	2	0
Minor	1	1	0
Medium	4	3	1
Major	0	0	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 Ferra to identify any potential issues and vulnerabilities in the source code of the Ferra 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 7 issues of varying severity, listed below.

ID	Title	Severity	Status
CON-1	Centralization Risk	Medium	Fixed
CON-2	Lack of Events Emit	Informational	Fixed
FAC-1	Incomplete Token Whitelist Checking	Medium	Acknowledged
POO-1	Precision Loss in Reward Calculation	Medium	Fixed
POO-2	Missing Check for `lock_until`	Minor	Fixed
POO-3	Incorrect Event Parameter	Informational	Fixed
UTI-1	Precision Loss in `str()`	Medium	Fixed

3 Participant Process

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

Admin

- Admin can update package version through `update_package_version()`.
- Admin can withdraw rewards emergently through `emergent_withdraw()`.
- Admin can update protocol fee rate through `update_protocol_fee_rate()`.
- Admin can delete fee tiers through `delete_fee_tier()`.
- Admin can add whitelist tokens through `add_whitelist_token()`.
- Admin can remove whitelist tokens through `delete_whitelist_token()`.
- Admin can pause the pool through `pause()`.
- Admin can unpause the pool through `unpause()`.
- Admin can update fee rate through `update_fee_rate()`.
- Admin can update position URL through `update_position_url()`.
- Admin can set display metadata through `set_display()`.
- Admin can collect protocol fees through `collect_protocol_fee()`.
- Admin can update emission through `update_emission()`.
- Admin can initialize rewarder through `initialize_rewarder()`.
- Admin can add or update fee tier through the `add_update_fee_tier()` function.
- Admin can set whether creating a pair is allowed through the `set_allow_create_pair()` function.
- Admin can set the upgrade capability through the `set_upgrade_cap()` function.
- Admin can set the publisher through the `set_publisher()` function.

- Admin can create governance proposals through the `propose()` function.
- Admin can vote on proposals through the `vote()` function.
- Admin can execute approved proposals through the `execute()` function.
- Admin can cancel proposals through the `cancel()` function.

User

- User can open a position through `open_position()`.
- User can add liquidity through `add_liquidity()`.
- User can add fixed amount liquidity through `add_liquidity_fix_coin()`.
- User can remove liquidity through `remove_liquidity()`.
- User can lock position through `lock_position()`.
- User can close position through `close_position()`.
- User can collect fees through `collect_fee()`.
- User can collect rewards through `collect_reward()`.
- User can perform flash loan through `flash_loan()`.
- User can perform flash swap through `flash_swap()`.
- User can repay flash loan through `repay_flash_loan()`.
- User can repay flash swap through `repay_flash_swap()`.
- User can repay add liquidity through `repay_add_liquidity()`.
- User can create a new pool through `create_pool()`.
- User can deposit rewards through `deposit_reward()`.

4 Findings

CON-1 Centralization Risk

Severity: Medium

Status: Fixed

Code Location:

sources/config.move;

sources/pool.move

Descriptions:

Centralization risk was identified in the smart contract:

- Admin can pause the pool through the `pause()` function.
- Admin can unpause the pool through the `unpause()` function.
- Admin can update fee rate through the `update_fee_rate()` function.
- Admin can withdraw rewards emergently through the `emergent_withdraw()` function.
- Admin can update protocol fee rate through `update_protocol_fee_rate()`.
- Admin can add roles through `add_role()`.
- Admin can remove roles through `remove_role()`.
- Admin can set roles through `set_roles()`.

Suggestion:

It is recommended that measures be taken to reduce the risk of centralization, such as a multi-signature mechanism.

Resolution:

This issue has been fixed. The client use a proposal mechanism for administrator configuration operations.

CON-2 Lack of Events Emit

Severity: Informational

Status: Fixed

Code Location:

sources/config.move#180,189

Descriptions:

Some functions in the contract lack events logging, which is essential for blockchain transparency, off-chain data tracking, and frontend integration. Event logs allow external systems to monitor contract activities without querying the blockchain state directly.

- `add_whitelist_token()`
- `delete_whitelist_token()`

Suggestion:

It is recommended to add event emission for this operations.

Resolution:

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

FAC-1 Incomplete Token Whitelist Checking

Severity: Medium

Status: Acknowledged

Code Location:

sources/factory.move#89

Descriptions:

Incomplete token whitelist checking logic exists in the create_pool function, allowing attackers to create liquidity pools containing malicious tokens. An attacker could pair a malicious token (e.g., one containing a backdoor, reentrancy attack, or infinite minting vulnerability) with any whitelisted token to create a pool.

```
assert!(
    config::is_in_whitelist<CoinTypeA>(global_config) ||
    config::is_in_whitelist<CoinTypeB>(global_config),
    1,
);
```

Suggestion:

It is recommended that the logic be modified to require both tokens to be on the whitelist.

POO-1 Precision Loss in Reward Calculation

Severity: Medium

Status: Fixed

Code Location:

sources/pool.move#392,527,585,611,759,808,1086,1258

Descriptions:

The reward calculation mechanism truncates fractional seconds during milliseconds-to-seconds conversion, leading to systematic under-distribution of rewards. This precision loss occurs due to integer division before multiplication, discarding remainders in each calculation cycle.

```
    rewarder::settle(
        &mut pool.rewarder_manager,
        pool.liquidity,
        clock::timestamp_ms(clock) / 1000,
    );
```

```
public(friend) fun settle(
    manager: &mut RewarderManager,
    liquidity: u128,
    current_time: u64
) {
    let last_time = manager.last_updated_time;
    manager.last_updated_time = current_time;
```

Suggestion:

It is recommended that a variable be added to the reward calculation to record the number of lost milliseconds. The next reward calculation should then be corrected to avoid missing rewards each cycle.

Resolution:

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

POO-2 Missing Check for `lock_until`

Severity: Minor

Status: Fixed

Code Location:

sources/pool.move#361

Descriptions:

The `open_position` function lacks a `lock_until` check. If `lock_until` is less than the current time, it will be meaningless. Alternatively, if `lock_until` is too large, the position may never be unlocked.

Suggestion:

It is recommended to check that `lock_until` is within a reasonable range.

Resolution:

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

POO-3 Incorrect Event Parameter

Severity: Informational

Status: Fixed

Code Location:

sources/pool.move#1306

Descriptions:

In the `flash_swap_internal()` function, the values of the parameters `before_sqrt_price` and `after_sqrt_price` in `SwapEvent` are the same, causing event to be logged incorrectly.

Suggestion:

It is recommended to modify the value of the parameter `before_sqrt_price` in `SwapEvent`.

Resolution:

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

UTI-1 Precision Loss in `str()`

Severity: Medium

Status: Fixed

Code Location:

sources/utils.move#16

Descriptions:

In the `ferra_clmm::utils::str()` function, the line `let digit = (num as u8) % 10;` incorrectly truncates the `num` value to a `u8` before performing the modulo operation. This `u8` cast causes an overflow and truncation for any `num` value greater than 255, leading to an incorrect digit extraction.

The function incorrectly returns `250` instead of the correct `256`:

- `(256 as u8)` is `0`, `0 % 10 = 0`, `digits` gets `0`;
- `num` becomes `25`, `(25 as u8)` is `25`, `25 % 10 = 5`, `digits` gets `5`;
- `num` becomes `2`. `(2 as u8)` is `2`. `2 % 10 = 2`. `digits` gets `2`.
- Result: `250`.

Suggestion:

The `u8` cast in the digit extraction logic should be removed or reordered to ensure the modulo operation is performed on the `u64` value before any truncation. The result of `num % 10` (which will always be between 0 and 9) can then be safely cast to `u8`.

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.