Cellana Smart Contract

Audit Report

contact@movebit.xyz

https://twitter.com/movebit_

Tue Feb 20 2024

Cellana Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	The Innovative Decentralized Exchange on Aptos
Type	DeFi
Auditors	MoveBit
Timeline	Wed Jan 31 2024 - Tue Feb 06 2024
Languages	Move
Platform	Aptos
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Cellana-Finance/v1-core
Commits	1cece7af6ac18d606120ed66c747fa791fb3467b 83c1012afa5e22ac31383e6502f11895495c7e14

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	Move.toml	5bbb532897700b570f24dbec5d3be1255bc126bf
RPO	sources/rewards_pool.move	16c6d001f2cebc9bbad86a5a33d27ca0928d3742
TWH	sources/token_whitelist.move	e8b7b768d15e04c710e442201e07bcb330fcfe24
ROU	sources/router.move	167b3bf34fb97380dd11f2cd7a55dd2d523f2db2
VMA	sources/vote_manager.move	c16d7266dd0098419e2d5f9b0df42d31db52f4e9
EPO	sources/epoch.move	2a80c3260ee106435b5bd8e2898a05fc13c8786c
CWR	sources/coin_wrapper.move	71170d86af345cb14d42dad16230640ddfce033a
LPO	sources/liquidity_pool.move	fe61ef796be9f5a504a6abd4a0b34627be0fa2a6
CTO	sources/cellana_token.move	0e09f3836509e179712c3819bc3b723bb0d79854
GAU	sources/gauge.move	8473e0b2595bf570793e6f25a0468cb7d1dfd47e
RPC	sources/rewards_pool_continuous.move	be53de90303cb105f011476e1be1794212eee73e

VES	sources/voting_escrow.move	831b928fefcd2925122c97172cf4a370fbc81ac2
MIN	sources/minter.move	bab2775d89d0ff910fa81541f03269cb4c68c7ca
PMA	sources/package_manager.move	2a90e4cd09ac707a5884a78682e2931d36b6743c

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	5	5	0
Informational	1	1	0
Minor	1	1	0
Medium	3	3	0
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

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 Cellana-Finance to identify any potential issues and vulnerabilities in the source code of the Cellana 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 5 issues of varying severity, listed below.

ID	Title	Severity	Status
LPO-1	Unused Constant	Informational	Fixed
MIN-1	Calculation Formula Does Not Match The Comment	Minor	Fixed
ROU-1	Insufficient Validation for `amounts_out`	Medium	Fixed
ROU-2	Incorrect Condition Statement	Medium	Fixed
ROU-3	Logic Design of The `swap_route_entry` Function	Medium	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Cellana Smart Contract : The administrator can assign some management rights to an address, such as rates, outages, etc. For convenience, these rights are also classified as administrators.

Admin

- The `Admin` can create the new gauge for `LPs` to stake and earn emissions from a new fees pool for `veCELL` voters to claim fees through `create_gauge_entry()` .
- The `Admin` can set the status of the gauge to active through `enable_gauge()` .
- The `Admin` can set the status of the gauge to inactive through `disable_gauge()` .
- The `Admin` can set a new admin through `update_operator()` .
- The `Admin` can set the governance through `update_governance()` .
- The `Admin` can add tokens to the whitelist through `whitelist_native_fungible_assets()` and `whitelist_coin<CoinType>()` .
- The `Admin` can publish a new package through `upgrade()` .
- The `Admin` can set the pending team account of the minter through `update_team_account()` .
- The `Admin` can set the team rate of minter through `set_team_rate()` .
- The `Admin` can set the pauser of liquidity pool through `set_pauser()` .
- The `Admin` can pause/unpause the liquidity pool through `set_pause()` .
- The `Admin` can set the fee manager through `set_fee_manager()` .
- The `Admin` can set the stable fee through `set_stable_fee()` .
- The `Admin` can set the volatile fee through `set_volatile_fee()` .
- The `Admin` can set the swap fee through `set_pool_swap_fee()` .

User

- The `User` can allocate voting weights to liquidity pools through an `NFT` through `vote()` .
- The `User` can vote for the current epoch with the same pools and weights as the last vote through `poke()` .

- The `User` can propose all rewards for designated `NFT` including fees and incentives through `claim_rewards_all<CoinType1,CoinType2, CoinType3, CoinType4, CoinType5, CoinType6, CoinType7,CoinType8, CoinType9, CoinType10, CoinType11, CoinType12, CoinType13, CoinType14,CoinType15,>()`
- The `User` can extract specified incentive through `claim_emissions_entry()`.
- The `User` can extract all incentives through `claim_emissions_multiple()`.
- The `User` can distribute emissions to pools based on the votes and collect swap fees from pools for voters to claim later through `advance_epoch()`.
- The `User` can mint a `veCELL NFT` and lock `$CELL` from the owner's primary store through `create_lock_entry()`.
- The `User` can mint a `veCELL NFT` for others through `create_lock_for()`.
- The `User` can increase the lockup duration of a `veCELL NFT` by the given number of epochs through `extend_lockup`.
- The `User` can deposit more `$CELL` into a `veCELL NFT` through `increase_amount_entry`
- The `User` can withdraw `$CELL` from an expired `veCELL NFT` and deposit into their primary store through `withdraw_entry`.
- The `User` can merge two `veCELL NFT` into one through `merge()`.
- The `User` can split an nft into multiple nfts through `split_entry()`.
- The `User` can claim all rebase rewards for a given `NFT` through `claim_rebase()`
- The `User` can swap token through `swap_entry()`, `swap_coin_for_asset_entry<FromCoin>()`, `swap_asset_for_coin_entry<ToCoin>()` or `swap_coin_for_coin_entry<FromCoin, ToCoin>`.
- The `User` can swap token via specify path through `swap_route_entry()`, `swap_route_entry_from_coin<FromCoin>()`, `swap_route_entry_to_coin<ToCoin>` or `swap_route_entry_both_coins<FromCoin, ToCoin>()`.
- The `User` can add whitelisted tokens as incentives to the pool through `incentivize_entry()` and `incentivize_coin_entry<CoinType>()`.

- The `User` can create a whitelisted tokens pool through `create_pool()`, `create_pool_coin<CoinType>` or `create_pool_both_coins<CoinType1, CoinType2>()`.
- The `User` can add liquidity through `add_liquidity_entry()`, `add_liquidity_both_coins_entry<CoinType1, CoinType2>()` or `add_liquidity_coin_entry<CoinType>()`.
- The `User` can add liquidity and stake the lp token into gauge through `add_liquidity_and_stake_entry()`, `add_liquidity_and_stake_both_coins_entry<CoinType1, CoinType2>()` or `add_liquidity_and_stake_coin_entry<CoinType>()`.
- The `User` can burn liquidity tokens to withdraw staked tokens through `remove_liquidity_entry()`, `remove_liquidity_coin_entry<CoinType>()`, `remove_liquidity_both_coins<CoinType1, CoinType2>` or `remove_liquidity_both_coins_entry<CoinType1, CoinType2>()`.
- The `User` can withdraw liquidity tokens and burn them to withdraw staked tokens through `unstake_and_remove_liquidity_entry()`, `unstake_and_remove_liquidity_coin_entry<CoinType>()` or `unstake_and_remove_liquidity_both_coins_entry<CoinType1, CoinType2>()`.
- The `User` can transfer a given amount of liquidity tokens from the sender to the receiver through `transfer()`.
- The `User` can update fee information through `update_claimable_fees()`.
- The `User` can stake liquidity tokens in gauge through `stake()`.
- The `User` can unstake liquidity tokens in gauge through `unstake()`.

4 Findings

LPO-1 Unused Constant

Severity: Informational

Status: Fixed

Code Location:

sources/liquidity_pool.move#46,55,57

Descriptions:

There are unused constants in the entire module.

Suggestion:

It is recommended to remove unused constants if there's no further design.

Resolution:

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

MIN-1 Calculation Formula Does Not Match The Comment

Severity: Minor

Status: Fixed

Code Location:

sources/minter.move#133

Descriptions:

The calculation formula in the `current_rebase` function comments is `Rebase = weekly emission * (total veCELL voting power / total $CELL supply) ^ 3`, while the actual calculation formula performed by the function is `Rebase = weekly emission * (total veCELL voting power / total $CELL supply) ^ 3 / 2`.

Suggestion:

It is recommended to confirm if it aligns with the design.

Resolution:

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

ROU-1 Insufficient Validation for `amounts_out`

Severity: Medium

Status: Fixed

Code Location:

sources/router.move#130

Descriptions:

In the `swap_route_entry` function, the assertion at `L130` only validates the last value in the `amounts_out` array, which is insufficient to verify that all values in the array are correct.

Suggestion:

It is recommended to verify each value in the `amounts_out` array individually.

Resolution:

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

ROU-2 Incorrect Condition Statement

Severity: Medium

Status: Fixed

Code Location:

sources/router.move#253

Descriptions:

In the `optimal_liquidity_amounts` function, the conditional statement `if (amount_2 <= amount_2_desired)` is always `true`. According to the context logic, the parameter `amount_2` should be changed to `amount_2_optimal`.

Suggestion:

It is recommended to modify the parameter `amount_2` to `amount_2_optimal`.

Resolution:

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

ROU-3 Logic Design of The `swap_route_entry` Function

Severity: Medium

Status: Fixed

Code Location:

sources/router.move#119

Descriptions:

The swap_route_entry function first swaps the first token from the `from_token` array with the first token from the `to_token` array. Then, it swaps the resulting token with the second token from the `to_token` array, and so on. Finally, it transfers the token from the last swap to the recipient. Shouldn't the correct design be to swap each token in the `from_token` array with the corresponding token in the `to_token` array

Suggestion:

It is recommended to confirm if it aligns with the design.

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.