Audit Report

Apollo Osmosis Fixed Width Range Vault

v1.0

September 4, 2024

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

THIS AUDIT REPORT WAS PREPARED EXCLUSIVELY FOR AND IN THE INTEREST OF THE CLIENT AND SHALL NOT CONSTRUE ANY LEGAL RELATIONSHIP TOWARDS THIRD PARTIES. IN PARTICULAR, THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THIRD PARTIES AND PROVIDE NO WARRANTIES REGARDING THE FACTUAL ACCURACY OR COMPLETENESS OF THE AUDIT REPORT.

FOR THE AVOIDANCE OF DOUBT, NOTHING CONTAINED IN THIS AUDIT REPORT SHALL BE CONSTRUED TO IMPOSE ADDITIONAL OBLIGATIONS ON COMPANY, INCLUDING WITHOUT LIMITATION WARRANTIES OR LIABILITIES.

This audit has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Apollo DAO to perform a security audit of Apollo Fixed Width Range Vault.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/apollodao/osmosis-fixed-width-range-vault
Commit	`aa7b2ec47125835df14fbc6ebb8e1dbffa6b2acc`
Scope	All contracts were in scope.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

The Apollo Fixed Width Range Vault contract is an auto-compounding vault that wraps a concentrated liquidity LP position on Osmosis that implements the `cw-vault-standard`. It provides a number of operations that facilitate position management. When users deposit or redeem from the vault, it claims the spread and incentive rewards and compounds them into the position.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	Medium-High	The codebase had a medium-high level of complexity due to the rebalancing logic and complex sub-message handling.
Code readability and clarity	Medium-High	The code follows CosmWasm best practices and is highly readable.
Level of documentation	Medium	The project did not provide detailed documentation apart from a high-level README document.
Test coverage	Medium	Overall the project had a medium amount of test coverage with good unit testing, but we were unable to generate test coverage metrics due to a failing test case.

Summary of Findings

No	Description	Severity	Status
1	Slippage is not correctly applied during liquidity balancing	Major	Resolved
2	The contract's position can arbitrarily be set to zero	Major	Resolved
3	Reply handler errors may block deposits and redemptions	Major	Resolved
4	Inability to rebalance would prevent all deposits and redemptions	Minor	Resolved
5	Depositor funds may remain unused	Minor	Resolved
6	Lack of configuration validations	Minor	Partially Resolved
7	Config update is lacking incentive denom validation	Minor	Resolved
8	Overflow checks not enabled for release profile	Minor	Resolved
9	Remove unnecessary and unused code	Informational	Resolved
10	Error message can be improved	Informational	Resolved
11	Unimplemented queries	Informational	Acknowledged
12	Ineffective sub-message error handling	Informational	Resolved

Detailed Findings

1. Slippage is not correctly applied during liquidity balancing

Severity: Major

In `contracts/osmosis-cl-vault-fixed-width/sc/helpers/mod.rs:377-383`, a swap message is created to balance the inventory held by the vault post-subscription or -redemption. This message ensures that the `token_out_min_amount` is not lower than an estimate plus the specified slippage. However, the variable `token_out_amount` is calculated using an estimate of the swap to which slippage is subsequently applied. As the output estimate already includes any price impact the further application of slippage is not effective.

Recommendation

We recommend calculating the `token_out_amount` value by multiplying a spot or TWAP price with the amount of token being swapped to which any desired slippage should be applied.

Status: Resolved

2. The contract's position can arbitrarily be set to zero

Severity: Major

The `execute_compound` function passes the argument `disregard_funds` to the `BalanceLiquidity` and `CreatePosition` messages in `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:52` and `61`.

If the argument `disregard_funds` is identical to the contract's current position in the concentrated liquidity pool, an empty position will be created as `CreatePosition` will ignore the provided funds when effectively creating the new position after having withdrawn the position in full due to the `MsgWithdrawPosition` message added in line `73`. The previous deposits will sit as a balance in the contract until a new `Compound` message is processed not ignoring the affected funds.

In addition to preventing any incentives and rewards from being generated in the meantime, this action could move the market significantly if the action is successful. As the `Compound` entry point is permissionless an attacker could combine this as part of a larger attack aimed to affect the price of the pool.

Recommendation

We recommend disallowing arbitrary users from making use of the `disregard_funds` feature. This can be achieved by raising an error if the `disregard_funds` vector is not empty and `info.sender` does not match the contract's address.

Status: Resolved

3. Reply handler errors may block deposits and redemptions

Severity: Major

The deposit and redeem functions of the contract first dispatch a compound sub-message before proceeding with their functionality. The `execute_compound` function in `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:18` dispatches `MsgCollectIncentives` and `MsgCollectSpreadRewards` sub-messages to the Osmosis `concentrated-liquidity` module. The Osmosis module then collects incentives and spread rewards for the contract address and returns the coins that the reply handler of the contract passes to the `send_fees_to_treasury` function.

The issue exists in the reply handler of the contract in `contracts/osmosis-cl-vault-fixed-width/src/reply.rs:163` and `169`. The reply handlers do not account for the situation where the Osmosis responses return no coins to be collected. In this case, the reply handlers would error when the `send_fees_to_treasury` function attempts to send an empty vector of coins. This would create a situation where deposits or redemptions could be blocked if there are no incentives or spread rewards to be collected.

An example scenario would be if there are some deposits or withdrawals within a single block. The first operation would successfully claim the spread rewards and incentives, but then subsequent calls would error because the amount of rewards and incentives would be returned as `None`.

Recommendation

We recommend adding a check in the reply handlers in `contracts/osmosis-cl-vault-fixed-width/src/reply.rs:163` and `169` to validate that the coins returned in the reply are not empty. If the coins are empty, the handler can simply emit the appropriate attributes and continue.

Status: Resolved

4. Inability to rebalance would prevent all deposits and redemptions

Severity: Minor

During execution of a rebalance operation in `contracts/osmosis-cl-vault-fixed-width/src/helpers/mod.rs:225-398`, a binary search is used to calculate how many tokens should be swapped to keep the target inventory ratio. If the search cannot be completed, an error is thrown in `contracts/osmosis-cl-vault-fixed-width/src/helpers/mod.rs369-373`.

This could occur frequently in a particularly illiquid market with an imbalanced vault and would prevent both deposits and redemptions from taking place and locking users' funds.

We classify this issue as minor as the scenario in which this occurs is unlikely.

Recommendation

We recommend allowing users to redeem with the current asset ratio in the case described above where the binary search is unable to complete.

Status: Resolved

5. Depositor funds may remain unused

Severity: Minor

The share of the pool that a user receives during the deposit of funds into the vault is calculated using the delta in liquidity provided. However, during the creation of the new position in `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:159-171` it is possible that not all deposited funds are added.

Under certain market conditions, the difference in the amount of liquidity supplied and added to the position could be non-negligible resulting in the depositor receiving a smaller proportion of share funds.

Recommendation

We recommend refunding user funds left over from the deposit post-creation of the new vault position to the sender.

Status: Resolved

6. Lack of configuration validations

Severity: Minor

The `osmosis-cl-vault-fixed-width` contract performs validation of its configuration through the `ConfigUnchecked::check` function in `contracts/osmosis-cl-vault-fixed-width/src/state.rs:149-244`. However, some fields lack validation which would prevent some features from being rendered unusable if an incorrect value is assigned by mistake.

- If `discount_factor` remains unvalidated, it could be set to a value such as `Decimal::one()` which would cause the vault to sell staked assets far below the market value. Therefore the allowed range should be limited, for example to be between 0 and 25%.
- If the `range_width` exceeds one, an overflow will occur in `contracts/osmosis-cl-vault-fixed-width/src/helpers/mod.rs:169`, resulting in an arbitrarily large `lower_tick`.
- If `balancing_tolerance` is set to zero, it could make balancing unreachable within the iterations due to dust rounding.
- If `redemption_rate_age_tolerance` is set to zero, the check in `contracts/osmosis-cl-vault-fixed-width/src/helpers/mod.rs:162` would always error. On the other hand, an excessively large value would allow old rates to be taken as valid.
- If `redemption_fee` is set to one, the ban message crafter in `contracts/osmosis-cl-vault-fixed-width/src/reply.rs:125` will result in a transfer of zero value, causing the whole transaction to revert.
- If `deposit_fee` is set to one, `contracts/osmosis-cl-vault-fixed-width/src/execute/basic_vault.rs:55` results in no remaining balance being available for `AddLiquidity` which will make `MsgAddToPosition` fail and the transaction revert.
- If `incentive_denoms` contains duplicate assets, the `BasketLiquidate` message of `execute_compound` in `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:108` will fail, reverting the whole transaction.
- If `incentive_denoms` has a large enough number of assets, the for loop in `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:91` may run out of gas, preventing compounding.

Recommendation

We recommend implementing validations for the fields listed above.

Status: Partially Resolved

7. Config update is lacking incentive denom validation

Severity: Minor

The `execute_update_config` function in `contracts/osmosis-cl-vault-fixed-width/src/execute/basic_vault.rs:120-139` allows the admin to modify the configuration's `incentive_denoms` vector. However, as it relies on `ConfigUnchecked::check` for validating the contents of the vector, the pool tokens `token0` and `token1` would be allowed to be part of the `incentive_denoms` vectors. This results in redundant operations as it will first swap one of the pool tokens for the other and then the `BalanceLiquidity` message will swap them again for the correct amounts.

Please note that this is not the case upon instantiation, as additional validation to cover this scenario is included in `contracts/osmosis-cl-vault-fixed-width/src/contract.rs:50-58`.

Recommendation

We recommend validating that the `incentive_denoms` vector does not contain either of the pool assets as part of the `ConfigUnchecked::check` function. The additional validation included in instantiation should consequently be removed, as it will become redundant.

Status: Resolved

8. Overflow checks not enabled for release profile

Severity: Minor

The following packages and contracts do not enable overflow-checks for the release profile:

- `contracts/osmosis-cl-vault-fixed-width/Cargo.toml`
- `Cargo.toml`

Recommendation

We recommend enabling overflow checks in the workspace and all packages, including those that do not currently perform calculations. Note that enabling overflow checks in packages other than the workspace manifest will lead to compiler warnings.

Status: Resolved

9. Remove unnecessary and unused code

Severity: Informational

The `osmosis-cl-vault-fixed-width` contract contains several pieces of unnecessary or unused code. Although not a security issue, this decreases readability and maintainability while slightly increasing gas costs in some scenarios.

- In `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:130` the `CONFIG` is loaded, but never used.
- In `contracts/osmosis-cl-vault-fixed-width/src/reply.rs:146` the balances of the pool assets are queried and assigned to `_pool_balances` but never used.
- The `PREVIOUS_LIQUIDITY` storage constant in `contracts/osmosis-cl-vault-fixed-width/src/state.rs:37` is never used.
- The `FORCE_WITHDRAW_WHITELIST` storage constant in `contracts/osmosis-cl-vault-fixed-width/src/state.rs:33` is only used in queries but information is never saved using it.
- In `contracts/osmosis-cl-vault-fixed-width/src/state.rs:208` and `216` `Decimal` values are validated to not be below zero. These conditions will never be met as the type only allows unsigned values.
- The `get_pool_asset_balances` function in `contracts/osmosis-cl-vault-fixed-width/src/helpers/mod.rs:120-131` is never used.
- In `contracts/osmosis-cl-vault-fixed-width/src/execute/contract.rs:51` a duplicate pool query is performed.
- When performing the binary search in `contracts/osmosis-cl-vault-fixed-width/src/helpers/mod.rs:318` the pool is queried on each iteration. However, the value could be retrieved prior to the loop saving computational resources.
- The `balance_liquidity` function calculates `tokens_after_swap` and includes it as a return value in `contracts/osmosis-cl-vault-fixed-width/src/helpers/mod.rs:386-395`. However, this information is never checked by the callee in a later reply handler, making it unused.
- During the contract instantiation, the `get_cl_pool_with_id` query is called twice with the same `pool_id` in `contracts/osmosis-cl-vault-fixed-width/src/contract.rs:38` and `51`.

Recommendation

We recommend removing unused code and resolving inefficiencies.

Status: Resolved

10. Error message can be improved

Severity: Informational

The `ConfigError` message in `contracts/osmosis-cl-vault-fixed-width/src/contract.rs:42` currently does not provide an informative error message to users. It is best practice to provide meaningful errors where possible. In this case, the minimum funds that must be sent are not communicated.

Recommendation

We recommend improving the error message mentioned above to specify the amount of minimum funds that must be sent.

Status: Resolved

11. Unimplemented queries

Severity: Informational

Many of the queries in the `osmosis-cl-vault-fixed-width` contract are unimplemented. Users wil hencel be unable to access vital information about the contract's state. The following queries are currently unimplemented:

- `PreviewDeposit`
- `PreviewRedeem`
- `TotalAssets`
- `ConvertToShares`
- `ConvertToAssets`

Recommendation

We recommend implementing the unimplemented queries in the `osmosis-cl-vault-fixed-width` contract.

Status: Acknowledged

12. Ineffective sub-message error handling

Severity: Informational

Sub-messages dispatched by the contract are all sent with the `ReplyOn::Always` directive. This option is useful when the contract specifically handles the return reply with a specific operation as is done for the `InternalMsgCompound` reply handler. In the reply handler in `contracts/osmosis-cl-vault-fixed-width/src/reply.rs:183` all other sub-message errors will simply be caught in the last condition. This is potentially error-prone and unnecessary as the sub-message errors will be effectively handled without the reply handling. Specifically, the following messages do not need to be dispatched with the `ReplyOn::Always` directive:

- `MsgCollectIncentives` in `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:64`
- `MsgCollectSpreadRewards` in `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:68`
- `MsgCreatePosition` in `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:182`
- `MsgAddToPosition` in `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:233`
- `MsgWithdrawPosition` in `contracts/osmosis-cl-vault-fixed-width/src/execute/compound.rs:263`

Recommendation

We recommend updating the sub-messages above to only reply on success rather than always.

Status: Resolved