Audit Report

Aperture

v1.0

March 8, 2022

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security

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

Introduction

Purpose of This Report

Oak Security has been engaged by `Terraform Labs Pte Ltd` to perform a security audit of Aperture.

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 GitHub repository:

https://github.com/Aperture-Finance/Aperture-Contracts

Commit hash: `1ceb7af2fb6c2687d431d8577ce750134f8c3302`

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

Aperture provides an ecosystem that presents users with investment opportunities aggregated across different blockchains, paired with a frictionless cross-chain experience. The strategies under the scope of this audit involve both, a stable yield strategy that utilizes Anchor earn, and a delta neutral investment strategy that involves Mirror protocol.

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 improve 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 a security audit and vice versa.

Summary of Findings

No	Description	Severity	Status
1	Missing tax deductions	Minor	Acknowledged
2	Price oracle block height not validated which may return out of date values	Minor	Acknowledged
3	Loss of precision during fee calculation	Minor	Resolved
4	Consider adding more validations to Solidity contracts	Minor	Resolved
5	Delta neutral position should implement a “migrate only if newer” pattern	Informational	Acknowledged
6	`migrate_position_contracts` allows anyone to perform migrations	Informational	Acknowledged
7	Certain cross-chain and Terra transfers are overly permissive	Informational	Acknowledged
8	Use of the term “info” may impact readability	Informational	Resolved

Code Quality Criteria

Criteria	Status	Comment
Code complexity	**Medium-High**	-
Code readability and clarity	**Medium-High**	-
Level of documentation	**Medium-High**	The Aperture documentation provides a good overview of the high-level functionality, and the inline comments describe the underlying mechanisms and implementation well.
Test coverage	**Medium-High**	-

Detailed Findings

1. Missing tax deductions

Severity: Minor

The stable yield manager does not deduct taxes. While Terra's tax rate has been set to zero, the tax mechanism is still implemented and the rate might be increased again in the future. It is still best practice to include functionality to deduct taxes.

We consider this to only be a minor issue since the contract owner can recover from tax mismatches by simply sending funds back to the contract. Additionally, the likelihood of the Terra team to increase taxes again is small.

Recommendation

We recommend implementing a tax rate query and deducting taxes from native assets to ensure future compatibility.

Status: Acknowledged

The Aperture team states that exclusion of tax deductions is an intentional design choice the team made to achieve reduced gas costs.

2. Price oracle block height not validated which may return out of date values

Severity: Minor

The `get_mirror_asset_oracle_uusd_price` function in `contracts/delta_neutral_position/src/util.rs:39-53` does not validate that the price information for the specified mAsset is up-to-date. Currently, the contract takes the returned rate and returns it as the valid uusd price for the mAsset to the caller without performing a block height validation to ensure that the price feed is accurate. The mirror price response includes both `last_updated_base` and `last_updated_quote` values for this purpose.

The Mirror mint contract does provide the functionality to return an error if the price is not within 60 seconds of the `block_time`, but the `get_mirror_asset_oracle_uusd_price` query does not currently provide the optional `block_time` parameter so this operation will not occur.

Recommendation

We recommend either performing a check to ensure that the Mirror price response's `last_updated_base` and `last_updated_quote` values match the current block

height or supplying the `block_time` parameter when querying the Mirror mint contract to rely on Mirror's price expiration functionality.

Status: Acknowledged

3. Loss of precision during fee calculation

Severity: Minor

The contract `EthereumManager` performs multiplication after division while calculating the fee in line 185. This can lead to imprecision due to rounding.

Recommendation

We recommend performing division after multiplication to avoid losing precision in the result.

Status: Resolved

4. Consider adding more validations to Solidity contracts

Severity: Minor

Consider adding the following validations in the Solidity contract to prevent accidental value updates.

1. `EthereumManager.sol`: Consider adding an upper limit for `CROSS_CHAIN_FEE_BPS` to prevent any accidental update that can cause wrong fee calculation and result in loss of funds for the end-user.
2. `EthereumManager.sol`: Consider adding zero address validation while updating the `FEE_SINK` address.

Recommendation

We recommend adding more validations to prevent any accidental fund loss.

Status: Resolved

5. Delta neutral position should implement a “migrate only if newer” pattern

Severity: Informational

The delta neutral positions contracts are currently migrated in an asynchronous fashion. This is fine, but the pattern can be improved by adding validation to ensure that the migration is only occurring if the version supplied is newer.

Recommendation

We recommend performing following the migrate “only if newer pattern” defined in the CosmWasm documentation.

Status: Acknowledged

6. `migrate_position_contracts` allows anyone to perform migrations

Severity: Informational

The `migrate_position_contracts` function in `contracts/delta_neutral_position_manager/src/contract.rs:144` does not restrict who can migrate the contracts. Currently, anyone can call this function and pass in whichever positions or position contract addresses they wish to migrate. While this is not inherently malicious, the best practice would be to restrict the entry-point.

We consider this issue to be informational since the code ID that’s used after migration can only be set by the administrator.

Recommendation

We recommend restricting the `migrate_position_contracts` to only be callable by the contract’s administrator.

Status: Acknowledged

The Aperture team states that due to the large expected number of delta-neutral position contracts, they have made a design choice to make migration publicly callable to allow both the position holder and the Aperture controller to perform migrations.

7. Certain cross-chain and Terra transfers are overly permissive

Severity: Informational

The public functions `initiate_outgoing_token_transfer` and `process_cross_chain_instruction` as found in `contracts/terra_manager/src/crosschain.rs:28` and `378` respectively are overly permissive. These functions are callable from any terra address, yet are only intended to be called by the Aperture strategy and manager contracts. In order to limit the attack surface of these contracts, it would be better to add access control to the contracts.

Recommendation

We recommend restricting the contracts to allow only known/used contracts and addresses where possible to avoid unnecessary attack surface to smart contracts.

Status: Acknowledged

8. Use of the term “info” may impact readability

Severity: Informational

The delta neutral position contract uses the term `info` in both `contracts/delta_neutral_position/src/rebalance.rs:123` and `contracts/delta_neutral_position/src/contract.rs:472` to represent terraswap pool info. This may impact readability and maintainability of the code because the name may conflict or be confused with `info`, which is commonly used to represent CosmWasm `MessageInfo`.

Recommendation

We recommend reserving the use of `info` to only represent CosmWasm `MessageInfo` and providing a more descriptive name for the instances described above.

Status: Resolved