Audit Report

Aperture 2

v1.0

June 27, 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 Aperture Lens, Inc. 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: `23c1af82355fd60a9956d84e96cd10c143fb7326`

The following CosmWasm smart contracts have been excluded from the scope of this audit:

- `contracts/anchor_earn_proxy`
- `contracts/delta_neutral_position`
- `contracts/delta_neutral_position_manager`

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 audit scope includes a proxy contract that interacts with Anchor protocol, a delta neutral position contract integrated with various Defi protocols, a manager contract that supervises the delta-neutral positions, a Terra manager contract that leverages Wormhole protocol to perform cross-chain investment and strategies, and finally, a Solidity smart contract that handles the Ethereum-side logic.

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.

Summary of Findings

No	Description	Severity	Status
1	Lack of token whitelist validation when swapping and creating a position or executing a strategy may lead to loss of user funds	Major	Partially resolved
2	Contracts are not compliant with CW2 Migration specification	Minor	Acknowledged
3	Inability to update the admin address of the `terra_manager` prevents incidence response	Minor	Resolved
4	Missing allowance check when executing a strategy with CW20 token causes meaningless error message	Minor	Resolved
5	Missing tax deductions	Minor	Acknowledged
6	Cross-chain fee BPS value may be set to be larger than `MAX_CROSS_CHAIN_FEE_BPS`	Minor	Resolved
7	Usage of deprecated `safeApprove` function opens up front-running possibilities	Minor	Resolved
8	Custom implementation of admin role increases code complexity	Informational	Resolved
9	Removing a strategy from `terra_manager` may lead to unlinked data	Informational	Acknowledged
10	Overflow checks not enabled for release profile	Informational	Acknowledged
11	Re-entrancy guard TODO comment in Solidity codebase	Informational	Resolved

Code Quality Criteria

Criteria	Status	Comment
Code complexity	High	-
Code readability and clarity	Medium-High	Complex code sections are documented with thorough code comments.
Level of documentation	Medium-High	Sufficient documentation was provided in GitBook and code comments.
Test coverage	Medium-High	`cargo tarpaulin` reports a test coverage of 74.15%.

Detailed Findings

1. Lack of token whitelist validation when swapping and creating a position or executing a strategy may lead to loss of user funds

Severity: Major

In `ethereum_hardhat/contracts/EthereumManager.sol:412` and `478`, the `swapTokenAndCreatePosition` and `swapTokenAndExecuteStrategy` functions allow users to swap tokens and create or execute a strategy based on the provided strategy and chain identifier. However, there is no validation that makes sure the swapped token `toToken` is whitelisted for that strategy identifier `strategyId` and chain identifier `strategyChainId`.

In contrast, the `createPosition` function in line `391` verifies that the `assetInfos`, `strategyId`, and `strategyChainId` arguments are valid using the `isTokenWhitelistedForStrategy` mapping in line `277` to ensure the assets are whitelisted for the specified strategy and chain identifier via the `validateAndTransferAssetFromSender` function.

The impact of this issue depends on how the contract on the other chain handles the non-whitelisted token. A possibility that might happen is that the recipient contract will reject the sent VAA message because the strategy does not support the asset token, causing a loss of funds for the sender.

Recommendation

We recommend validating `toToken`, `strategyId`, and `strategyChainId` to be whitelisted in the `swapTokenAndCreatePosition` and `swapTokenAndExecuteStrategy` functions. This can be done by checking the `isTokenWhitelistedForStrategy` mapping using a require statement as seen in lines `276-281`.

Status: Partially resolved

This issue is partially resolved since the patch only adds a check against `isTokenWhitelistedForStrategy` to the `swapTokenAndCreatePosition` function. The team states that the `swapTokenAndExecuteStrategy` function is left out because the `strategyId` is managed by the Aperture Manager contract on the `strategyChainId` chain. Thus, it is not possible to perform this validation on the originating chain under the current design.

The team states that it is impossible to perform all validations on the originating chain, and some validations can only be performed on the strategy chain. For example, aside from tokens, the position open params can also be invalid, and only the strategy chain's contract can interpret the params. Therefore, they plan to introduce a mechanism for the strategy

chain to reject invalid VAAs and refund the tokens back to the sender on the originating chain.

2. Contracts are not compliant with CW2 Migration specification

Severity: Minor

The smart contracts do not adhere to the CW2 Migration specification standard. This may lead to unexpected problems during contract migration and code version handling.

Recommendation

We recommend supporting the CW2 standard in all the protocol contracts. For reference, see https://docs.cosmwasm.com/cw-plus/0.9.0/cw2/spec.

Status: Acknowledged

The team states they will make their CosmWasm contracts CW2-compliant in a future iteration.

3. Inability to update the admin address of the `terra_manager` prevents incidence response

Severity: Minor

In `contracts/terra_manager/src/contract.rs`, the admin address is saved in the storage in the instantiate message. After that, the admin can't change its address. This may lead to severe implications if the private keys controlling the admin account are ever compromised, in which case an update of the admin address would help to prevent exploits.

Recommendation

We recommend implementing a message that allows the admin to change its address. For a battle-tested implementation, we suggest using https://docs.rs/cw-controllers/latest/cw_controllers/struct.Admin.html#method.execute_update_admin

Status: Resolved

4. Missing allowance check when executing a strategy with CW20 token causes meaningless error message

Severity: Minor

In `packages/aperture_common/src/token_utils.rs:47`, when creating the message to move assets from the sender to the current contract, the execution uses a `cw20::Cw20ExecuteMsg::TransferFrom`.

This type of message requires that the sender provided an allowance for the spender contract. If no allowance is granted, message execution will fail without a meaningful error message.

Recommendation

We recommend implementing a guard in order to verify if the sender granted sufficient allowance to the contract and returning a meaningful error message otherwise.

Status: Resolved

5. Missing tax deductions

Severity: Minor

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.

A non-zero tax rate could be reinstated via a governance proposal due to circumstances where the expected income from the tax rewards increases significantly. In this situation, stablecoin transactions on Terra would expand to a state where a meaningful portion of the staking rewards income is derived from tax rewards rather than the vast majority coming from swap fees.

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 low.

See this discussion for more details about the tax rate changes on Terra.

Recommendation

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

Status: Acknowledged

The team states that since the chance of Terra Classic governance increasing the tax rate is low, and since Aperture has no immediate plans to launch strategies on the Terra Classic chain, they decided to make no changes at this time.

6. Cross-chain fee BPS value may be set to be larger than `MAX_CROSS_CHAIN_FEE_BPS`

Severity: Minor

In `ethereum_hardhat/contracts/EthereumManager.sol:119`, the `_crossChainFeeBPS` value is not validated to be below the `MAX_CROSS_CHAIN_FEE_BPS` value as seen in line `131`. As a result, an admin setting a `_crossChainFeeBPS` higher than `MAX_CROSS_CHAIN_FEE_BPS` will cause the calculated cross-chain fee in line `328` to be larger than intended.

Recommendation

We recommend validating the provided `_crossChainFeeBPS` value to be less than or equal to `MAX_CROSS_CHAIN_FEE_BPS` as seen in lines `130` to `133` with a require statement.

Status: Resolved

7. Usage of deprecated `safeApprove` function opens up front-running possibilities

Severity: Minor

In `ethereum_hardhat/contracts/EthereumManager.sol:187` and `338`, the `safeApprove` function is used to give approvals for other contracts to use the current contract's token balance. However, the `safeApprove` functionality is deprecated as it opens up possibilities for sandwich attacks similar to the classic ERC20 approve() function.

Recommendation

We recommend modifying the implementation to use the `safeIncreaseAllowance` function.

Status: Resolved

8. Custom implementation of admin role increases code complexity

Severity: Informational

In `contracts/terra_manager/src/contract.rs` a custom implementation of an admin role is used. Using a battle-tested reference implementation reduces the complexity of the codebase.

Recommendation

We recommend using https://docs.rs/cw-controllers/latest/cw_controllers/struct.Admin.html from the `cw-controllers` crate.

Status: Resolved

9. Removing a strategy from `terra_manager` may lead to unlinked data

Severity: Informational

In `contracts/terra_manager/src/contract.rs:63`, when executing `ExecuteMsg::RemoveStrategy`, the selected strategy metadata is removed from `STRATEGY_ID_TO_METADATA_MAP`.

As positions in `POSITION_TO_STRATEGY_LOCATION_MAP` are potentially pointing to a deleted strategy, this may lead to partial/unlinked data in the storage.

Recommendation

We recommend implementing a way to disable strategies instead of removing them.

Status: Acknowledged

The team states that removing a strategy essentially disables that strategy in the sense that new positions are prevented from being opened and that existing positions can still be modified using data stored in the strategy metadata map. Thus, the current setup works per their design, and perhaps a better name for `RemoveStrategy` would be `DeactivateStrategy`.

10. Overflow checks not enabled for release profile

Severity: Informational

The `contracts/terra_manager/Cargo.toml` package does not enable `overflow-checks` for the release profile.

While enabled implicitly through the workspace manifest, a future refactoring might break this assumption.

Recommendation

We recommend enabling overflow checks in all packages, including those that do not currently perform calculations, to prevent unintended consequences if changes are added in future releases or during refactoring. Note that enabling overflow checks in packages other than the workspace manifest will lead to compiler warnings.

Status: Acknowledged

The team states that they decided not to add a release profile to individual packages that would cause compiler warnings because their development guideline requires all CosmWasm packages to be placed in a workspace with a proper release profile.

11. Re-entrancy guard TODO comment in Solidity codebase

Severity: Informational

In `ethereum_hardhat/contracts/EthereumManager.sol:290`, there's an outstanding TODO comment indicating whether or not to implement a reentrancy guard in `recordNewPositionInfo` functionality. If the reentrancy guard is not applied, an attacker can keep reentering `recordNewPositionInfo` functionality, allowing the attacker to create fake positions under them. That said, there aren't direct security risks since the rest of the code execution validates the attacker sent assets as specified in the `assetInfos` and `fromToken` argument.

Recommendation

We recommend resolving and removing the TODO comment from the codebase by implementing a reentrancy guard.

Status: Resolved