Audit Report

White Whale

v1.0

February 14, 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 Keep the Peg Corp. to perform a security audit of the White Whale smart contracts.

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/White-Whale-Defi-Platform/contracts/tree/oak_audit

Commit hash: `a715c71088875877adba7e87ca5d69cd04a661e9`

The scope of the audit has been limited to the following contracts and related packages:

- `stablecoin-vault`
- `stable-arb-terra`
- `profit-check`
- `treasury`
- `memory`
- `dapp-template`
- `terraswap-dapp`

- `vault-dapp`

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 smart contracts audited implement the vault architecture with flash loan functionality used by the White Whale arbitration bots, including the TerraSwap and Anchor integrations and treasury functionality.

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	Separation of flash loan payback verification into separate contract could allow administrator to steal funds and introduces misconfiguration risks	Major	Resolved
2	Systemic dependence on Anchor may impede the ability to stabilize the UST peg via arbitrage during bank run cycle	Major	Acknowledged
3	Fixed fee buffer might lead to failure of large flash loans	Minor	Resolved
4	Flash loan and treasury related messages could run out of gas due to too many whitelisted contracts, assets or dapps	Minor	Resolved
5	Unvalidated and too high fee parameters might cause errors in production	Minor	Resolved
6	Anchor exchange rate might be out of date and as a consequence transactions might fail	Minor	Resolved
7	Denomination check is incomplete	Minor	Resolved
8	Zero amounts being sent in messages will fail	Informational	Resolved
9	Canonical address transformations are inefficient	Informational	Resolved
10	Copy and paste of base dapp file into every dapp is inefficient	Informational	Resolved
11	Leftover boilerplate state migration code	Informational	Resolved
12	Repeated typo with wrong address in documentation of stablecoin-vault and stablecoin-arb contract's instantiate message	Informational	Resolved

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium-High	The codebase consists of a large number of modules designed to be able to support several future use cases. This makes the architecture and the individual modules complex with many inter-module dependencies. The architecture is very well designed. However, the modularization of the different sub-contracts introduces a high complexity and a cluster of addresses that needs to be meticulously managed, for example when updating sub-contracts or admin rights. We recommend either reducing the complexity or implementing a very solid and transparent procedure for migration, setting new admins and state updates. It could make sense to implement grace periods such that migration or admin updates need to be confirmed within one hour/the next x block(s) or are automatically reversed.
Code readability and clarity	Medium-High	-
Level of documentation	Medium-High	The codebase is well supported with user-facing documentation, architecture descriptions and code comments.
Test coverage	Medium-High	-

Detailed Findings

1. Separation of flash loan payback verification into separate contract could allow administrator to steal funds and introduces misconfiguration risks

Severity: Major

The `stablecoin-vault` relies on the amount being lent in a flash loan to be tracked in the separate `profit-check` contract, which then checks that the amount has been repaid in full. This means that the actual flashloan functionality and the repay check are implemented in two different admin-configured contracts. As a result, a compromised admin could configure the `profit-check` address in the `stablecoin-vault` contract to point to a custom version that does not require full flash loan repayment for its own loans and use the flashloan function to extract user funds from the vault.

In addition, having two independent contracts in a 1-to-1 that have to be configured separately with the corresponding address pointers to each other is error-prone and could lead to accidental misconfiguration.

Recommendation

We recommend implementing the repay check within the same contract that implements the flashloan functionality.

Status: Resolved

2. Systemic dependence on Anchor may impede the ability to stabilize the UST peg via arbitrage during bank run cycle

Severity: Major

The litepaper explains how bank run cycles can be a threat to the UST peg. If systemic risks materialize in the form of an event that has an impact on the whole Terra ecosystem, this will likely impact Anchor as well. In addition, a malfunction of Anchor or a bank run on Anchor might trigger substantial corrections in the Terra ecosystem. The fact that only a fixed amount of UST – i.e. the `stable_cap` – is meant to remain in the vault will decrease the ratio of UST/aUST in the vault with increasing liquidity in the vault. With a growing amount of liquidity in the vault, the ability to stabilize the UST peg via arbitrage will depend more and more on Anchor being seamlessly working, which is not guaranteed during a bank run cycle.

Moreover, there is no validation when setting the stable cap in `contracts/stablecoin-vault/src/contract.rs:770` via `set_stable_cap`, meaning that the value could be set unrealistic values accidentally.

Recommendation

We recommend that the stable peg is the maximum of a minimal `stable_cap` and a percentage of the vault that is able to ensure arbitrage in favour of the UST peg. Alternatively, the team could diversify the funds across multiple asset classes that are not correlated with aUST. In addition, we would like to highlight that the `stable_cap` should be validated when set via `set_stable_cap` to be within an adequate range – i.e. the percentage of the vault should be in the interval of (0, 1).

Status: Acknowledged

The White Whale team recognises that the UST vault is heavily dependent on Anchor protocol but doesn't recognise it as a systemic risk due to the nature of the lending market. Nonetheless, the team still recognises that such an event might happen.

Rather than implementing a percentage of liquid assets the team will keep their current implementation where a target amount of UST is kept liquid.

The team explained that they will monitor the average trade sizes and adjust this parameter depending on their data. In a worst-case scenario, their bots will be able to artificially reduce the active trade size to the liquid amount of UST, in order to be able to do arbitrage, even during a bank run.

The team decided not to implement a percentage cap of liquid assets because this might decrease the profitability of the protocol. The team decided not to diversify the vault assets because this would change the underlying product by creating price exposure.

3. Fixed fee buffer might lead to failure of large flash loans

Severity: Minor

In `contracts/stablecoin-vault/src/contract.rs:37`, `FEE_BUFFER` is set to a constant value. The buffer is used in line `268` to assure that the contract has enough liquidity to pay taxes and fees. The messages for the fees and taxes are then created in lines `295` and `299` as functions of the amount and fees/taxes. If fees and taxes exceed the buffer, large flash loans might fail because they will not be profitable.

Recommendation

We recommend calculating the relevant fees and taxes before the transaction or making the `FEE_BUFFER` a function of the transaction amount and the fees/taxes.

Status: Resolved

4. Flash loan and treasury related messages could run out of gas due to too many whitelisted contracts, assets or dapps

Severity: Minor

In `contracts/stablecoin-vault/src/contract.rs:244` the `handle_flashloan` function might run out of gas if the number of `whitelisted_contracts` gets very large.

As this is unlikely and can only happen through governance motions this is only a minor issue.

Similarly, in `contracts/treasury/treasury/src/contract.rs` an unlimited number of assets might cause executions of dapp-messages to fail by running out of gas, if the transactins uses the query `compute_total_value`.

Lastly, in `contracts/treasury/treasury/src/contract.rs:160`, the removal of a dapp might fail if there are too many dapps and the message runs out of gas.

This might not be a big issue, since the [hite Whale architecture figure] indicates that dapps will be limited.

Recommendation

We recommend setting a maximum number for `whitelisted_contracts`, `dapps` and `assets`.

Status: Resolved

5. Unvalidated and too high fee parameters might cause errors in production

Severity: Minor

In `contracts/stablecoin-vault/src/contract.rs:838-856` all fees in `set_fee` can be set to values equal to or larger than 1. These can cause errors, e.g. in line `428`, where `Decimal::from_ratio(amount - treasury_fee, total_share)` causing the withdrawal functionality to fail.

Similarly, we highlight for informational purposes, that the validation of `new_fee.share > Decimal::one()` in `contracts/treasury/dapps/vault/src/commands.rs:293` allows for a fee equal to 1, which would trigger inefficient messages.

Recommendation

We recommend adding a validation such that all fees are strictly smaller than 1.

Status: Resolved

6. Anchor exchange rate might be out of date and as a consequence transactions might fail

Severity: Minor

In `contracts/stablecoin-vault/src/contract.rs:271` the `query_aust_exchange_rate` functionality is used in the simplified form without the current block height as an input parameter. Anchor in this case uses stored values to calculate the exchange rate, which might be out of date. This implies that there may be imprecisions in the Anchor withdrawal message below in lines `451-454` and `655`, which could lead to a failure in execution.

Recommendation

We recommend implementing `query_aust_exchange_rate` to interact with Anchor using the current block height.

Status: Resolved

7. Denomination check is incomplete

Severity: Minor

In `packages/white_whale/src/denom.rs` the length (size of the String) of a denomination is checked to be smaller than or equal to five in order to determine whether a given token is a native token. This might fail because a coin's `Denom` can have between 2 and 127 characters.

Recommendation

We recommend implementing an enum with native asset and asset variants, for example like TerraSwap does.

Status: Resolved

8. Zero amounts being sent in messages will fail

Severity: Informational

In `contracts/treasury/dapps/terraswap/src/commands.rs:133`, `120` and `172`, zero amounts might be sent. These messages will always fail since zero transfers revert.

Recommendation

We recommend validating amounts to be greater than zero before sending a message.

Status: Resolved

9. Canonical address transformations are inefficient

Severity: Informational

While previously recommended as a best practice, usage of canonical addresses is no longer encouraged. The background is that canonical addresses are no longer stored in a canonical format, so the transformation just adds overhead without much benefit. Additionally, the codebase is more complicated with address transformations.

Recommendation

We recommend removing any transformation from human to canonical addresses and using the new `Addr` type for validated addresses instead.

Status: Resolved

10. Copy and paste of base dapp file into every dapp is inefficient

Severity: Informational

The file `contracts/treasury/dapps/terraswap/src/dapp_base/common.rs` is copied into every dapp contract. This is inefficient, replicates the code and is prone to human error if hard-coded addresses change and have to be modified everywhere.

Recommendation

We recommend putting the file into a package and importing it.

Status: Resolved

11. Leftover boilerplate state migration code

Severity: Informational

The files `contracts/stablecoin-vault/src/contract.rs` and `contracts/treasury/treasury/src/contract.rs` contain commented-out state migration code in the `migrate` function, which seems to have been copied over from a template file.

Recommendation

We recommend removing the unused code.

Status: Resolved

12. Repeated typo with wrong address in documentation of stablecoin-vault and stablecoin-arb contract's instantiate message

Severity: Informational

In the documentation of both Stablecoin Vault and Stablecoin Arb,the paragraph `InstantiateMsg` states that:

`pool_address`	String	Contract address of Profit Check Contract
`treasury_address`	String	Contract address of Profit Check Contract

This is a typo which might likely confuse developers.

Recommendation

We recommend replacing "*Contract address of Profit Check Contract" by the* actual description.

Status: Resolved