Audit Report

Anchor Liquidation Queue

October 18, 2021

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 to perform a security audit of the Anchor Liquidation Queue smart contract.

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/Anchor-Protocol/money-market-contracts/

Commit hash: `fed70a085d233625466745e1697043ca0e7eaaa0`
Relevant directories:

- `contracts/liquidation_queue`
- `packages/moneymarket/src/liquidation_queue.rs`

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 submitted contracts implement the new Liquidation mechanism of Anchor protocol, a DeFi savings protocol on the Terra blockchain.

The mechanism consists of a Liquidation Queue, a more capital efficient liquidation model using bids as pooled capital to liquidate undercollateralized loans.

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**. Informational notes do not have a status, since we consider them optional recommendations.

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	Liquidation execution is permissionless and not tied to Anchor contracts, which may cause depletion of the liquidation queue	Critical	Resolved
2	Triggering of liquidations has no economic incentives or gas compensation, which can cause slower liquidations and make the admin team as a single point of failure	Major	Resolved
3	Sent tokens other than in stable denom are lost	Major	Resolved
4	Users can activate any number of bids with any amount when `available_bids` is lower than `bid_threshold`	Minor	Resolved
5	Missing validations on `UpdateConfig` parameters may lead to inconsistent state	Minor	Partially resolved
6	Lack of validation of bid fee as well as max slot and premium rate per slot might lead to panics during liquidation execution	Minor	Resolved
7	`LiquidationAmount` query implies a nonlinear dependency tree	Informational	-
8	Any collateral could be whitelisted which may negatively impact user experience	Informational	-
9	Bid and bids by user query responses mixes stored and current values which might confuse users	Informational	-

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium-High	-
Level of Documentation	Medium	-
Test Coverage	Medium-High	-

Detailed Findings

1. Liquidation execution is permissionless and not tied to Anchor contracts, which may cause depletion of the liquidation queue

Severity: Critical

In the current design, the `ExecuteLiquidation` call in `liquidation_queue/src/bid.rs:273` is permissionless, so any other protocol can piggyback on the liquidity/bids in Anchor's liquidation queue. In a black swan event, liquidations coming from external parties/protocols might leave the Anchor liquidation queue empty, causing Anchor's liquidations to fail.

Recommendation

We recommend restricting access to `ExecuteLiquidation` to the `Custody` contracts of Anchor.

Status: Resolved

2. Triggering of liquidations has no economic incentives or gas compensation, which can cause slower liquidations and make the admin team as a single point of failure

Severity: Major

In the current design, the `ExecuteLiquidation` call in `liquidation_queue/src/bid.rs:273` is expected to come from the `Custody` contract, which in turn is called by the `Overseer::LiquidateCollateral` message. However, an external liquidator may not have explicit economic incentives to execute that function, as the `Custody::LiquidateCollateral` message in `contracts/custody_bluna/src/collateral.rs:156` sets the overseer contract as the recipient of fees and the market contract as the recipient of the repay amount. The caller does not get compensated for the gas cost of the call, and may not have an economic reason either.

In a crash of collateral prices, the Anchor protocol benefits from liquidations happening as fast as possible. Without explicit economic incentives, this burden relies solely on the Anchor team, making them a single point of failure subject to DoS attacks.

Recommendation

We recommend adding compensation for liquidation executors, to incentivize fast and decentralized liquidations.

As an example, the Liquity protocol compensates liquidation executors with 200 LUSD plus 0.5 % of the liquidated collateral (ETH).

A low-hanging fruit change in the code could be assigning the `fee_address` parameter to the `info.sender` executing `Overseer::LiquidateCollateral`.

Status: Resolved

3. Sent tokens other than in stable denom are lost

Severity: Major

When a user submits a bid while sending funds that are not in the stable denom, the check in `contracts/liquidation_queue/src/bid.rs:31-42` does not return an error, which implies that any such tokens are inaccessibly locked in the contract.

Recommendation

We recommend either returning an error if unsupported funds are sent or returning those funds to the sender.

Status: Resolved

4. Users can activate any number of bids with any amount when `available_bids` is lower than `bid_threshold`

Severity: Minor

During bid activation, a loop is responsible for handling the bid activation process by checking `assert_activate_status` against `available_bids`. The value of `available_bids` is only read once outside of the loop in `contracts/liquidation_queue/src/bid.rs:105`, resulting in the condition within `assert_activate_status` being checked with the same `available_bids` value in every iteration.

Consequently, a user can activate any number of bids with any amount whenever the initial `available_bids` is only 1 unit under the collateral's `bid_threshold`.

Recommendation

We recommend making `available_bids` mutable and incrementing it in the loop in `contracts/liquidation_queue/src/bid.rs:147` by the bid's amount.

Status: Resolved

5. Missing validations on `UpdateConfig` parameters may lead to inconsistent state

Severity: Minor

The `update_config` function in `src/contract.rs:137` permits the owner to modify parameters that may result in inconsistent state. Examples of this issue are `stable_denom` and `waiting_period`.

Recommendation

We recommend either validating or restricting `update_config` for these parameters, or implement migration of outstanding bids when `update_config` is executed.

Status: Partially resolved

The Anchor team removed the ability to update `stable_denom`, but keeps allowing changes to the `waiting_period`, acknowledging that changes to that parameter will only affect new bids, not already stored ones.

6. Lack of validation of bid fee as well as max slot and premium rate per slot might lead to panics during liquidation execution

Severity: Minor

The `bid_fee` config parameter is not validated to be less than or equal to one. If it is greater than one, the subtraction in `contracts/liquidation_queue/src/bid.rs:347` may panic, which would prevent liquidations. Similarly, there is currently no validation that `max_slot * premium_rate_per_slot` is less than one. If it is greater than or equal to one, `contracts/liquidation_queue/src/bid.rs:502` or `508` would panic due to underflow or division by zero.

Recommendation

We recommend adding validation to the `bid_fee` config parameter in the `instantiate` and `update_config` functions to ensure that the value is less than or equal to one. Likewise, we recommend adding validation to the product `max_slot * premium_rate_per_slot` to ensure that the value is less than one.

Status: Resolved

7. `LiquidationAmount` query implies a nonlinear dependency tree

Severity: Informational

Several `Config` parameters such as `safe_ratio`, `liquidation_threshold` and `Overseer` address are only used in the `LiquidationAmount` query. That query interacts with the `Overseer` contract, which implies a non-linear tree of contract dependencies.

Recommendation

We recommend moving the query logic of `Query::LiquidationAmount` into the `Overseer`, to follow the single responsibility principle and to have a better separation of concerns.

8. Any collateral could be whitelisted which may negatively impact user experience

Severity: Informational

The `WhitelistCollateral` message does not check that the collateral to be whitelisted is present in the list of collaterals in the `Overseer`, which implies that collaterals not present in Anchor might be added.

There is currently no delisting functionality in the liquidation queue contract. Since collaterals can still be effectively disabled by updating `max_slot` to 0, this issue does not have any impact on the functionality of the contracts, but implies an ever growing list of collaterals, resulting in a downgrade of user experience.

Recommendation

To prevent whitelisting of collaterals not supported by Anchor, we recommend asserting that the newly whitelisted collateral is already present in the list of collaterals in the `Overseer`.

9. Bid and bids by user query responses mixes stored and current values which might confuse users

Severity: Informational

The `BidResponse` to the bid and bids by users queries currently contains in `contracts/liquidation_queue/src/query.rs:209` and `259` the current `bid_amount` and `bid_pending_liquidated_collateral`, but also contains the snapshot values used to calculate those current values. That might be confusing to users, since the snapshot values are inputs to calculating the current `bid_amount` and `bid_pending_liquidated_collateral`.

Recommendation

We recommend either returning the stored `bid.amount` and `bid.pending_liquidated_collateral` with the snapshot values such that users can use those values as inputs for calculations, or to return the current `bid_amount` and `bid_pending_liquidated_collateral` without any snapshot values.