Audit Report

Stargaze Reserve Auctions

v1.0

June 20, 2023

Table of Contents

Table of Contents 2
License 3
Disclaimer 3
Introduction 5
Purpose of This Report 5
Codebase Submitted for the Audit 5
Methodology 6
Functionality Overview 6
How to Read This Report 7
Code Quality Criteria 8
Summary of Findings 9
Detailed Findings 10
1. Unhandled zero-amount transfers in marketplace contract 10
2. Protocol does not enforce minimum reserve price to be in STARS denom 10
3. Missing parameter validations in the reserve auction contract 11
4. Incorrect errors for InvalidTradingFeeBps and InvalidBidRemovalRewardBps 11
5. MAX_FIXED_PRICE_ASK_AMOUNT is not enforced during ask price update 12
6. TODO comment in the codebase 12
7. Duplicate finder fee validation 12
8. Reserve prices can be updated for expired auctions 13

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 Stargaze Foundation to perform a security audit of Stargaze's Reserve Auctions.

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/public-awesome/marketplace
Commit	`c7f471f3960e53e45400cea96817f45d78e83869`
Scope	• `contracts/reserve-auction` • Changes to `contracts/marketplace` since our last audit, which was performed at commit `07ad739453965f1322c21d95478df601d78738d9`. • `packages/common/src/lib.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 Stargaze reserve auction contract allows bidders to bid on NFTs with increasing value for a certain time duration. Additionally, NFT royalties are respected when handling payouts.

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	**Low-Medium**	-
Code readability and clarity	**Medium-High**	-
Level of documentation	**Medium-High**	Detailed documentation was provided in the README file.
Test coverage	**High**	cargo tarpaulin reports a `93.53%` test coverage.

Summary of Findings

No	Description	Severity	Status
1	Unhandled zero-amount transfers in marketplace contract	Major	Resolved
2	Protocol does not enforce minimum reserve price to be in `STARS` denom	Major	Resolved
3	Missing parameter validations in the reserve auction contract	Minor	Resolved
4	Incorrect errors for `InvalidTradingFeeBps` and `InvalidBidRemovalRewardBps`	Minor	Resolved
5	`MAX_FIXED_PRICE_ASK_AMOUNT` is not enforced during ask price update	Minor	Resolved
6	TODO comment in the codebase	Informational	Resolved
7	Duplicate finder fee validation	Informational	Resolved
8	Reserve prices can be updated for expired auctions	Informational	Resolved

Detailed Findings

1. Unhandled zero-amount transfers in marketplace contract

Severity: Major

In several occurrences when performing `BankMsg::Send`, there is no validation to ensure the amount to transact is greater than zero. Consequently, the transaction will fail and revert.

1. In `contracts/reserve-auction/src/helpers.rs:143-146`, the `settle_auction` function will fail if the remaining amount is zero after deducting the protocol and royalty fees.
2. In `packages/common/src/lib.rs:78-82`, the `royalty_payout` function will fail if the royalty's share is zero.
3. In `contracts/marketplace/src/execute.rs:1209-1216` and lines `1223-1230`, the `payout` function will fail if the remaining amount is zero after deducting the royalty, network, and finder fees.
4. In `contracts/marketplace/src/execute.rs:1023-1030` and lines `1074-1081`, the `execute_remove_stale_bid` and `execute_remove_stale_collection_bid` functions will fail if the value of `params.bid_removal_reward_percent` is set to zero or to the maximum bps value.

Recommendation

We recommend applying the following suggestions:

1. Validate that the remaining amount is not zero before sending funds to the seller.
2. Validate that the royalty share is not zero before sending funds to the royalty receiver.
3. Modify the if condition in lines `1197` and `1219` to use less than or equal to (<=).
4. Validate that the amount is not zero before sending funds to the bidder and operator.

Status: Resolved

2. Protocol does not enforce minimum reserve price to be in `STARS` denom

Severity: Major

In `contracts/reserve-auction/src/helpers.rs:131`, the `fair_burn` function is called to burn a portion of the bid amounts, expecting the bid denom to be `STARS`. This is problematic because there is no validation ensuring that `config.min_reserve_price` is in the `STARS` denom.

Additionally, when settling the auction in lines `143-146`, the `settle_auction` function distributes funds with the `STARS` denom. Consequently, if the `config.min_reserve_price` is denominated in a different denom, incorrect funds will be distributed instead.

Recommendation

We recommend validating that the `min_reserve_price` is denominated in `STARS` during the contract instantiation and inthe `SudoMsg::UpdateParams` message handler.

Status: Resolved

3. Missing parameter validations in the reserve auction contract

Severity: Minor

In `contracts/reserve-auction/src/instantiate.rs:26-29`, the `msg.min_bid_increment_bps` and `msg.create_auction_fee` parameters are not validated to be less than the max bps value and greater than zero.

The former would cause the `min_bid` function in line `contracts/reserve-auction/src/execute.rs:277` to increase to a value greater than 100% of its original value, while the latter would cause the `execute_create_auction` function to fail in line `96` because it forces the caller to send zero amount of `STARS`, which is prevented by the `must_pay` function.

We classify this issue as minor because the governance can recover this situation using `SudoMsg::UpdateParams` message.

Recommendation

We recommend validating `create_auction_fee` and `min_bid_increment_bps` parameters in contract instantiation and `SudoMsg::UpdateParams` message handlers.

Status: Resolved

4. Incorrect errors for `InvalidTradingFeeBps` and `InvalidBidRemovalRewardBps`

Severity: Minor

In `contracts/marketplace/src/error.rs:64-68`, the `InvalidTradingFeeBps` and `InvalidBidRemovalRewardBps` errors are misleading as they use the same error message as `InvalidFindersFeeBps`. This implies that the caller will receive an error message regarding invalid finder fee bps, although the issue is caused by trading fee or bid removed reward bps.

Recommendation

We recommend modifying the custom errors mentioned above to their intended error message.

Status: Resolved

5. `MAX_FIXED_PRICE_ASK_AMOUNT` is not enforced during ask price update

Severity: Minor

In `contracts/marketplace/src/execute.rs:412`, the `execute_update_ask_price` function does not validate the `price.amount` does not exceed `MAX_FIXED_PRICE_ASK_AMOUNT`. This invariant is enforced in line `308` during the `execute_set_ask` function, but it is not enforced when updating the asking price.

Recommendation

We recommend validating that the price amount does not exceed the `MAX_FIXED_PRICE_ASK_AMOUNT` constant in the `execute_update_ask_price` function.

Status: Resolved

6. TODO comment in the codebase

Severity: Informational

In `packages/common/src/lib.rs:65`, a TODO comment is in the codebase. TODOs in production code are a deviation from best practices.

Recommendation

We recommend resolving and removing the TODO comment.

Status: Resolved

7. Duplicate finder fee validation

Severity: Informational

In `contracts/marketplace/src/execute.rs:474-478` and lines `484-488`, duplicate validations of `finders_fee_bps` are performed. In fact, the only difference between these two checks are variable names (`fee` vs `finders_fee_bps`), but the validations are identical.

Recommendation

We recommend removing one of the two validations to improve readability and efficiency.

Status: Resolved

8. Reserve prices can be updated for expired auctions

Severity: Informational

In `contracts/reserve-auction/src/execute.rs:158`, the `execute_update_reserve_price` function does not validate whether the auction is expired. An expired auction should not be modifiable.

Recommendation

We recommend preventing the seller from updating an expired auction's reserve price.

Status: Resolved