Audit Report

Astroport Governance

v1.0

March 15, 2022

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

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 Delphi Labs to perform a security audit of the Astroport Governance features.

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/astroport-fi/astroport-core
Commit hash: `bd8f8599e1b1867b6e6a005bc4cef209699683e6`
Relevant directories:

- `contracts/xastro_token`
- `contracts/pair_stable_bluna`
- relevant files in `packages`

https://github.com/astroport-fi/astroport-governance
Commit hash: `6b132f946b72f8c7d9f9b7a8c28ed2a021cc1d78`

https://github.com/astroport-fi/astrozap
Commit hash: `faf4bfacd7ad775773a4b418b163fb3825bd97d8`

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 code for this audit includes functionality of the Astral Assembly, which allows xASTRO stakers and the core team to create proposals and vote on them, the xASTRO token & bLUNA stableswap pool update and the AstroZap contract that allows imbalance LPing in constant product pools.

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	Claiming allocation will cause proposed receiver's original allocation to be overwritten, leaving funds inaccessible in the contract	Critical	Resolved
2	Original receivers that transferred allocation will not be able to receive new allocations	Major	Resolved
3	Leftover amount after providing liquidity is not refunded	Major	Resolved
4	Misconfigured schedule duration could cause division by zero error, leaving funds inaccessible	Minor	Resolved
5	`UnlockedTokens` query message does not include `cliff` period during calculation	Minor	Resolved
6	Extra funds sent to AstroZap contract are lost	Minor	Resolved
7	Duplicate accounts creation in xAstro token instantiation would cause inflated xAstro total supply	Minor	Acknowledged

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium-High	-
Code readability and clarity	Medium-High	-
Level of documentation	Medium-High	-
Test coverage	Medium	There were no test cases for bLUNA stableswap pool update.

Detailed Findings

1. Claiming allocation will cause proposed receiver's previous allocation to be overwritten, leaving funds inaccessible in the contract

Severity: Critical

In `contracts/builder_unlock/src/contract.rs:305-311` of the `astroport-governance` repository, the proposed receiver's allocation is overwritten via `PARAMS.save` and `STATUS.save` without verifying whether they have existing `AllocationParams` and `AllocationStatus`. If the proposed receiver decides to claim a new allocation while having an existing allocation, their existing allocation's `ASTRO` token will be stuck in the contract.

Recommendation

We recommend reverting with an error if the proposed receiver has an existing `AllocationParams` or `AllocationStatus` when claiming a new allocation.

Status: Resolved

2. Original receivers that transferred allocation will not be able to receive new allocations

Severity: Major

In `contracts/builder_unlock/src/contract.rs:303-311` of the `astroport-governance` repository, if the original receiver had transferred their ownership of allocation to a new receiver, their `AllocationParams` is removed via `PARAMS.remove` in line `308`. However, their `AllocationStatus` is not removed via `STATUS.remove`, which means that the original receiver will have an outdated `AllocationStatus` but no associated `AllocationParams` with it.

This is problematic since the original receiver will be unable to receive new allocations due to lines `153-158`.

Recommendation

We recommend removing the original receiver's `AllocationStatus` in `contracts/builder_unlock/src/contract.rs:303-311`.

Status: Resolved

3. Leftover amount after providing liquidity is not refunded

Severity: Major

In `contracts/astrozap/src/contract.rs:84` of the `astrozap` repository, `offer_asset` is calculated by using Newton's method. Since the method does not have `100%` accuracy (in the referenced white paper, the specific example shows a `0.04%` error), it will cause a remainder of tokens to be left in the contract. Moreover, Newton's method does not guarantee convergence within `MAX_ITERATIONS` (see `contracts/astrozap/src/math.rs:74`). It can output erroneous values for `offer_asset`.

Recommendation

We recommend refunding the leftover amounts when LP tokens are transferred to users at `contracts/astrozap/src/contract.rs:293`.

Status: Resolved

4. Misconfigured schedule duration could cause division by zero error, leaving funds inaccessible

Severity: Minor

In `contracts/builder_unlock/src/contract.rs:427` of the `astroport-governance` repository, `schedule.duration` is used as a denominator to calculate the amount of tokens that can be unlocked between start time and end time. If the value is zero, a division by zero error would occur and will cause the specific allocation to be locked in the contract.

We consider this to be a minor issue since it can only be caused by the owner.

Recommendation

We recommend verifying the value of `schedule.duration` to not be zero when creating new allocations.

Status: Resolved

5. `UnlockedTokens` query message does not include `cliff` period during calculation

Severity: Minor

In `contracts/builder_unlock/src/contract.rs:362-372` of the `astroport-governance` repository, `query_tokens_unlocked` uses

`compute_unlocked_amount` to compute the number of tokens that are unlocked according to the current timestamp. The function does not include the schedule's cliff period when calculating the withdrawable token amount, this would cause an incorrect amount returned to the caller. An example calculation that includes a cliff period can be found in `contracts/builder_unlock/src/contract.rs:442-445`.

Recommendation

We recommend including the cliff period during calculation in `query_tokens_unlocked`.

Status: Resolved

6. Extra funds sent to AstroZap contract are lost

Severity: Minor

In `contracts/astrozap/src/contract.rs:74-79` of the `astroport-governance` repository, user's sent funds are being used in `handle_deposits` to verify that the funds user claimed to have deposited are actually deposited to the contract. The current implementation does not return an error though if additional native tokens are sent to the contract, leaving them stuck in the contract.

We consider this to be a minor issue, since it is caused by a user error.

Recommendation

We recommend reverting with an error if a user sends extra funds that are not used during the `Enter` execute message phase.

Status: Resolved

7. Duplicate accounts creation in xAstro token instantiation would cause inflated xAstro total supply

Severity: Minor

In `contracts/xastro_token/src/contract.rs:104-108` of the `astroport-core` repository, duplicate accounts are not verified when creating initial accounts during the contract instantiation phase. If the same account address is passed twice in `create_accounts`, the account's balance would be overwritten via `BALANCES.save` but `total_supply` would still record the balance amount of both. As a result, `xAstro` token's total supply would be inflated.

We consider this to be a minor issue since it can only be caused by the owner.

Recommendation

We recommend returning an error if duplicate accounts exist in the `initial_balances` vector that's passed into `create_accounts`.

Status: Acknowledged

The Astroport team states that `xASTRO` will not have an initial supply, hence this is not an issue.