Audit Report

Astroport LP Token Balance Tracking

DRAFT – DO NOT PUBLISH

v0.2

April 27, 2023

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 Delphi Labs Ltd. to perform a security audit of Astroport's LP Token Balance Tracking feature.

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 changes to the following contract since our previous audit, which was based on commit `30f7bf348da4600d0b3f56f0e89de9e0c0495299`:

Repository	https://github.com/astroport-fi/astroport-core
Commit	`3a82eecc83fcc444fb86ac98b222303d43fda691`
Scope	The changes to the contract in `contracts/pair` and `packages/astroport/src/pair.rs` were in scope.

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

This audit has been performed on updates to Astroport's pair contract, which now includes asset balance tracking functionality, enabling real-time insights for better pool management.

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	-
Test coverage	Medium-High	-

Summary of Findings

No	Description	Severity	Status
1	Pending “TODO” comment indicates that a function is not needed	Informational	Resolved
2	Inconsistent usage of `query_pools` function	Informational	Resolved
3	Factory address cannot be queried	Informational	Resolved
4	Misleading owner during config query	Informational	Resolved

Detailed Findings

1. Pending “TODO” comment indicates that a function is not needed

Severity: Informational

The `decimal2decimal256` function was moved out of the `pair` contract into `packages/astroport/src/lib.rs:72-78`, where a “TODO” comment was included stating that the function was no longer necessary. Keeping unused code reduces maintainability.

Recommendation

We recommend removing the `decimal2decimal256` function and using `Decimal256::from(v: Decimal)`, as the comment states.

Status: Resolved

2. Inconsistent usage of `query_pools` function

Severity: Informational

In several instances of the `pair` contract, the `contract_addr` argument for the `query_pools` function is used inconsistently. For example, lines `contracts/pair/src/contract.rs:334` and `633` use `&env.contract.address`, while lines `778`, `940`, and `989` use `&config.pair_info.contract_addr`.

Recommendation

We recommend using the `&config.pair_info.contract_addr` argument throughout the codebase to improve readability.

Status: Resolved

3. Factory address cannot be queried

Severity: Informational

In `contracts/pair/src/contract.rs:1064-1073`, the `query_config` function does not include `config.factory_addr` in the `ConfigResponse` struct. Consequently, users cannot query the factory address from the `pair` contract’s queries.

Recommendation

We recommend including the factory address in the `ConfigResponse` struct.

Status: Resolved

4. Misleading owner during config query

Severity: Informational

The `owner` value is set to `None` in `contracts/pair/src/contract.rs:1071`. Users may interpret this as the contract not being subject to any updates, as no contract owner exists. This is misleading though, as the factory owner can call `XYKPoolUpdateParams::EnableAssetBalancesTracking`, which enables asset tracking in the pool.

Recommendation

We recommend renaming `owner` to `pool_owner` to distinguish between the pool contract's owner and the factory owner.

Status: Resolved