Audit Report

Astroport Tokenfactory LP Tokens

v1.0

July 2, 2024

Table of Contents

License

THIS WORK IS LICENSED UNDER A CREATIVE COMMONS ATTRIBUTION-NODERIVATIVES 4.0 INTERNATIONAL 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 REPORT WAS PREPARED EXCLUSIVELY FOR AND IN THE INTEREST OF THE CLIENT AND SHALL NOT CONSTRUE ANY LEGAL RELATIONSHIP TOWARDS THIRD PARTIES. IN PARTICULAR, THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THIRD PARTIES AND PROVIDE NO WARRANTIES REGARDING THE FACTUAL ACCURACY OR COMPLETENESS OF THE AUDIT REPORT.

FOR THE AVOIDANCE OF DOUBT, NOTHING CONTAINED IN THIS AUDIT REPORT SHALL BE CONSTRUED TO IMPOSE ADDITIONAL OBLIGATIONS ON COMPANY, INCLUDING WITHOUT LIMITATION WARRANTIES OR LIABILITIES.

COPYRIGHT OF THIS REPORT REMAINS WITH THE AUTHOR.

This audit has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Astroport Protocol Foundation to perform a security audit of Astroport Tokenfactory LP Tokens implementation.

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/astroport-fi/astroport-core
Commit	`ee3ab6fa147b15fa7f3045c4260a65b2d7101255`
Scope	Changes between commit `02eafc677d9c402a55e45e956fd185d67ac3a8a6` and commit `ee3ab6fa147b15fa7f3045c4260a65b2d7101255` excluding the `contracts/pair_xyk_sale_tax` directory.
Fixes verified at commit	`cca2cbcf600a38cb737465c15709685648e2e2a6` Note that only fixes to the issues described in this report have been reviewed at this commit. Any further changes such as additional features have not been reviewed.

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 Astroport Tokenfactory LP Tokens audit covers the implementation of tokenfactory LP tokens instead of previously used CW20 LP tokens, except for the `pair_xyk_sale_tax` contract. Additionally, potential side effects that may occur during deployment on supported chains, i.e. Neutron, Injective, Sei, and Terra were investigated.

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	Medium	-
Code readability and clarity	Medium-High	Most functions and state variables are documented with clear and concise comments.
Level of documentation	Medium	Documentation is available in the `README` files.
Test coverage	Medium-High	`cargo tarpaulin` reports test coverage of `80.15%`.

Summary of Findings

No	Description	Severity	Status
1	Inconsistent support for `SetBeforeSendHook` message in `Pair` and `Pair Concentrated` contracts	Major	Resolved
2	`WithdrawLiquidity` assets parameter is always expected to be empty	Minor	Acknowledged
3	Indexer for `ProvideLiquidity` is hardcoded to zero	Minor	Resolved
4	Depreciation of `CosmosMsg:Stargate` could affect future compatibility	Minor	Acknowledged
5	Wrong error message returned	Informational	Resolved
6	Missing `TRACER_CONFIG` validation during querying	Informational	Resolved
7	Comments are not reflected in code	Informational	Resolved

Detailed Findings

1. Inconsistent support for `SetBeforeSendHook` message in `Pair` and `Pair Concentrated` contracts

Severity: Major

An inconsistency exists within the `tf_before_send_hook_msg` function called in `contracts/pair/src/contract.rs:180` and `contracts/pair_concentrated/src/contract.rs:238`. This function internally constructs a `CosmosMsg::Stargate` message intended to execute the `SetBeforeSendHook` message.

However, the `SetBeforeSendHook` message is not implemented on chains like Injective and Sei. This missing functionality leads to failures during the instantiation of `pair` and `pair_concentrated` contracts when deploying the protocol on these chains.

Conversely, the functionality works as expected on chains that support the Osmosis protocol buffer definition.

Recommendation

We recommend gating the functionality of calling `tf_before_send_hook_msg` under a feature flag, so it does not get executed on chains like Injective and Sei.

Status: Resolved

2. `WithdrawLiquidity` assets parameter is always expected to be empty

Severity: Minor

The `WithdrawLiquidity` message has been newly added to each pair. Previously, it was part of the `periphery/liquidity_manager` contract, which has been removed from the source code.

For each pair's `WithdrawLiquidity` message, the `assets` parameter is expected, which is a vector of `Asset` objects. This parameter is mandatory. However, for example in `contracts/pair/src/contract.rs:545` it is checked whether `assets` are empty. If so, the logic continues to execute the function. Otherwise, an error is returned.

This is unintuitive because the user must send an empty vector each time to withdraw his liquidity, which is not a good practice and may lead to incorrect calls of the `WithdrawLiquidity` message.

Recommendation

We recommend analyzing why `WithdrawLiquidity` expects an `assets` parameter and then checks whether it is empty. If the logic is expected not to use the parameter, this parameter should be removed along with the code ensuring it is empty.

Status: Acknowledged

3. Indexer for `ProvideLiquidity` is hardcoded to zero

Severity: Minor

In individual pairs, for example, in `contracts/pair/src/contract.rs:463`, the liquidity-providing operation implements three types of events. One for the first provision, minting LP's with index 0, another for all subsequent liquidity provisions minting LP's with increment index, and a third one, which saves information about the `astroport-pool.v1.ProvideLiqudity` operation.

While the first two indexers work correctly, the third one has a hardcoded value of "0". This is problematic because it does not allow for the correct indexing of `ProvideLiquidity`-related events. If any external code or offchain component relies on these events, its operation may be impaired.

Recommendation

We recommend analyzing the reason why the index is always set to zero, and if this is unexpected from the contract logic implementing an appropriate `ProvideLiquidity` event indexing mechanism.

Status: Resolved

4. Depreciation of `CosmosMsg::Stargate` messages could affect future compatibility

Severity: Minor

The codebase employs the `CosmosMsg::Stargate` message type to execute chain-native module calls. However, with the release of CosmWasm 2.0, the Stargate message type has been renamed to `Any`. Consequently, if a chain adopts CosmWasm 2.0, deployed contracts utilizing the Stargate message type will experience functionality issues.

This notification highlights this potential incompatibility to ensure client awareness and facilitate proactive planning for future compatibility with CosmWasm 2.0.

Recommendation

We recommend proactive planning for future compatibility with CosmWasm 2.0.

Status: Acknowledged

5. Wrong error message returned

Severity: Informational

In several places in the codebase, a misleading `Unauthorized` error is returned in situations where `liquidity_token` is not empty:

- `contracts/pair_transmuter/src/contract.rs:85`
- `contracts/pair_stable/src/contract.rs:153`
- `contracts/pair_concentrated/src/contract.rs:214`
- `contracts/pair/src/contract.rs:156`

Recommendation

We suggest adapting the code to return the correct descriptive error type and message.

Status: Resolved

6. Missing `TRACER_CONFIG` validation during querying

Severity: Informational

The `query_tracker_config` function in `contracts/factory/src/contract.rs:556` is responsible for returning the `TrackerConfig` object with parameters such as `code_id` and `token_factory_addr`.

During the `factory` contract instantiation, however, `TRACKER_CONFIG` is optional, so there is a chance that it will have no value. In this case, the call to `QueryMsg::TrackerConfig` will revert, which is unexpected.

Recommendation

We suggest verifying that `TRACKER_CONFIG` exists before returning a value in `query_tracker_config`. If not, a descriptive error message should be returned informing the caller of the reason for the problem.

Status: Resolved

7. Comments are not reflected in code

Severity: Informational

The `mint_liquidity_token_message` functions across the codebase have extensive documentation in the form of comments.

In recent code changes, the references to the `Generator` contract were changed to the `Incentive` contract.

Changes occur in the following implementations:

- `contracts/pair_stable/src/utils.rs:164`
- `contracts/pair/src/contract.rs:477`
- `packages/astroport_pcl_common/src/utils.rs:56`

However, the rest of the code and variables refer directly to the nomenclature associated with "generator", which is misleading. This decreases the maintainability of the codebase.

Recommendation

We suggest adapting the code to the new nomenclature and standardizing the technical documentation and the corresponding implementation.

Status: Resolved