Audit Report

Astroport Maker and Vesting Contract Updates

v1.0

April 4, 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 updates to Astroport's Maker and Vesting smart contracts.

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

Repository	https://github.com/astroport-fi/astroport-core
Commit	`1f50cabf6738f6ad57b6ed7b1d56f1276fe6d526`
Scope	The changes to the contract in `contracts/tokenomics/maker` and `packages/astroport/maker.rs` were in scope.

Repository	https://github.com/astroport-fi/astroport-core
Commit	`042b0768951422099f5d77224c320978cbfa92cc`
Scope	The changes to the contract in `contracts/tokenomics/vesting` and `packages/astroport/vesting.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

New functionality was added to the maker contract that allows the distribution of part of the collected fees to the second receiver. The vesting contract received a new `WithdrawFromActiveSchedule` message that allows withdrawing funds from active vesting schedules.

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	Most functions are well documented with clear and concise comments.
Level of documentation	Medium-High	Detailed documentation is available at https://docs.astroport.fi/docs/develop/smart-contracts/tokenomics/maker and https://docs.astroport.fi/docs/develop/smart-contracts/tokenomics/vesting.
Test coverage	Low-Medium	`maker` code coverage is 50.84%. `vesting` code coverage is 34.94%.

Summary of Findings

No	Description	Severity	Status
1	Schedule's start and end point timestamps are not validated to be in the future	Minor	Partially Resolved
2	Unbounded iteration through `schedules` can permanently inhibit the execution of transactions	Minor	Resolved
3	Misconfigured zero max spread causes swaps to fail	Minor	Resolved
4	Governance fee percentage is not set to 100 if the staking contract address is not set	Minor	Resolved
5	No events are emitted upon successful contract instantiation	Informational	Resolved
6	Incomplete parameter documentation	Informational	Resolved
7	Outstanding TODO comment	Informational	Resolved
8	Zero amount withdrawal will always fail	Informational	Resolved
9	Redundant check in `calc_schedule_unlocked_amount`	Informational	Resolved

Detailed Findings

1. Schedule's start and end point timestamps are not validated to be in the future

Severity: Minor

The `assert_vesting_schedules` function defined in `contracts/tokenomics/vesting/src/contract.rs:240-254` ensures that the scheduled end point is past the start point.

However, there is no check in place that enforces the start point or the endpoint to be in the future. This implies that there may be vesting schedules that instantly vest, which will probably only happen unintentionally.

We classify this issue as minor because only the contract owner can create vesting schedules.

Recommendation

We recommend validating that the timestamps of both the start and end points are, or at least the end point is, in the future.

Status: Partially Resolved

The client states that they will allow the schedule's start time to be in the past because it would be difficult to align it with the time of the Assembly proposals execution.

2. Unbounded iteration through `schedules` can permanently inhibit the execution of transactions

Severity: Minor

In `contracts/tokenomics/vesting/src/contract.rs:324` and `contracts/tokenomics/vesting/src/contract.rs:386`, unbounded loops are used to iterate through all the registered `schedules`.

Consequently, if the cardinality of registered `schedules` is significant, the execution could run out of gas and revert the transaction. Additionally, since there is no way to remove completed `schedules`, this could permanently inhibit the execution of transactions.

We classify this issue as minor because only the contract owner can create vesting schedules.

Recommendation

We recommend enforcing a maximum limit of `schedules` per user and implementing the removal of completed schedules.

Status: Resolved

3. Misconfigured zero max spread causes swaps to fail

Severity: Minor

In `contracts/tokenomics/maker/src/contract.rs:59`, the `max_spread` value is not validated to be greater than zero. If the `max_spread` value is misconfigured as zero, all swaps will fail due to a `MaxSlippageAssertion` contract error.

This issue is also present during the configuration update phase in line `698`.

We classify this issue as minor because only the contract owner can cause it.

Recommendation

We recommend validating that the max spread is greater than zero during the contract instantiation and configuration update phase.

Status: Resolved

4. Governance fee percentage is not set to 100 if the staking contract address is not set

Severity: Minor

The documentation in `packages/astroport/src/maker.rs:17` states that if `staking_contract` is set to `None`, the `governance_percent` value should be equal to `100`.

However, this behavior is not enforced during contract instantiation and the `UpdateConfig` message handling.

Recommendation

We recommend enforcing the `governance_percent` to `100` if the staking contract address equals `None`.

Status: Resolved

5. No events are emitted upon successful contract instantiation

Severity: Informational

In `contracts/tokenomics/maker/src/contract.rs:97`, no custom events or attributes are emitted upon successful contract instantiation. This prevents off-chain listeners from indexing parameters configured by the contract instantiator.

Recommendation

We recommend emitting relevant events upon successful contract instantiation.

Status: Resolved

6. Incomplete parameter documentation

Severity: Informational

In several instances of the codebase, function comments do not include documentation of all parameters:

- `contracts/tokenomics/maker/src/contract.rs:106-112` and `614-624`
  - `second_receiver_params` is not included.
- `contracts/tokenomics/vesting/src/contract.rs:61-75`
  - The `RegisterVestingAccounts`, `WithdrawFromActiveSchedule`, `ProposeNewOwner`, `DropOwnershipProposal`, and `ClaimOwnership` messages are not documented.
- `contracts/tokenomics/vesting/src/contract.rs:369`
  - The `account`, `receiver`, and `amount` parameters are not documented.
- `contracts/tokenomics/maker/src/utils.rs:132-143`
  - The `bridge_token` and `factory_contract` parameters are not documented.

Recommendation

We recommend completing the documentation for the missing parameters.

Status: Resolved

7. Outstanding TODO comment

Severity: Informational

In `contracts/tokenomics/maker/src/utils.rs:21`, a TODO comment is present that questions whether the swap simulation should adjust according to the token's precision. This indicates that the codebase might not be ready for production.

Recommendation

We recommend resolving or removing the TODO.

Status: Resolved

8. Zero amount withdrawal will always fail

Severity: Informational

In `contracts/tokenomics/vesting/src/contract.rs:375`, the `withdraw_from_active_schedule` function does not validate that the amount to withdraw is not zero. As Cosmos SDK prevents zero-amount native token transfers, specifying zero withdrawal amounts will fail. The resulting transfer error might confuse users.

Recommendation

We recommend returning a custom error when the withdrawal amount is zero.

Status: Resolved

9. Redundant check in `calc_schedule_unlocked_amount`

Severity: Informational

The `calc_schedule_unlocked_amount` function in `contracts/tokenomics/vesting/src/contract.rs:349` checks whether the `time_period` is not zero. This check is redundant as new schedules are validated through the `assert_vesting_schedules` function in line `246`. This function enforces `end_point.time` to be strictly greater than `start_point.time`, implying that `time_period` is always greater than zero.

Recommendation

We recommend removing the unnecessary check in line `349`.

Status: Resolved