Audit Report

Astroport vxASTRO Updates

v1.0

August 20, 2024

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 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.

This audit has been performed by

Oak Security GmbH

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

Introduction

Purpose of This Report

Oak Security GmbH has been engaged by Astroport Protocol Foundation to perform a security audit of vxASTRO changes in Governance and Core modules.

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 targets:

Repository	https://github.com/astroport-fi/astroport-governance
Commit	`0236871cbdd58db92cbceaaa973f8728d3eceb2f`
Scope	Audit of the changes compared to commit `73f7f7e682ff79fd13d76d64c504fc28dc867685`.
Fixes verified at commit	`af6ea1d827a1d6038fe9c6db92c711395a87f7a0` 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.

Repository	https://github.com/astroport-fi/astroport-core
Commit	`0773b0855de7fe05d2efb2fa68e9ade830100ddc`
Scope	Audit of the changes compared to commit `beb8b774e4c17d06c4cc7b39893de7b53c9bbfb8`.
Fixes verified at commit	No fixes have been verified since we did not find any issues in this repository.

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 report covers new contracts and changes made mostly in the governance repository related to the creation of the Emission Controller component, allowing vxASTRO holders to vote on ASTRO emissions. Its duty is to register the new governance proposals, submitted in the Assembly contract, on all outposts. Once registered, vxASTRO outpost stakers can vote on them. The Controller receives IBC messages and applies votes in the Assembly contract.

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	-
Level of documentation	Medium-High	Diagrams and flow are available in every contract's `assets` directories and `README` files.
Test coverage	Medium-High	`cargo tarpaulin` reports a summarized test coverage of 84.64%: - 81.26% for `core` - 92.52% for `governance`

Summary of Findings

No	Description	Severity	Status
1	ASTRO pools can be voted for emissions	Major	Resolved
2	The `QueryWhitelist` query might run out of gas	Minor	Resolved
3	Previous owner retains contract migration privileges	Minor	Partially resolved
4	Multiple storage states are not exposed through smart queries	Informational	Partially resolved
5	Inconsistent documentation	Informational	Resolved

Detailed Findings

1. ASTRO pools can be voted for emissions

Severity: Major

In `contracts/emissions_controller/src/execute.rs:211-221`, the `whitelist_pool` function returns an error if the pool is one of the outpost's ASTRO pools. This means that if an ASTRO pool is configured in the outposts, the pool cannot be voted on because they already receive flat emissions, as indicated by the comment in line `211`.

However, this validation can be bypassed if a user whitelists the pool before the contract owner updates the pool via the `update_outpost` function. This causes the pool to remain whitelisted, and users can vote on it to receive emissions.

Consequently, the validation in `contracts/emissions_controller/src/execute.rs:211-221` is not enforced, allowing the pool to receive voted and flat emissions, which is not the protocol's intended behavior.

Recommendation

We recommend modifying the `update_outpost` function to remove the ASTRO pool from the `POOLS_WHITELIST` and `VOTED_POOLS` states to prevent users from voting on them.

Status: Resolved

2. The `QueryWhitelist` query might run out of gas

Severity: Minor

In `contracts/emissions_controller/src/query.rs:101` the `QueryWhiteList` returns all whitelisted pools. Since pool creation is permissionless, there is no limit to the number of existing pools. Therefore, the whitelist may become very long, either due to a malicious actor or due to organic growth. In such a case, the query may not be able to complete due to exceeding the gas limit. Although this does not impact any other existing protocol logic, aside from this query itself, it can have security implications for other protocols using this query.

Recommendation

We recommend implementing a pagination mechanism on this query.

Status: Resolved

3. Previous owner retains contract migration privileges

Severity: Minor

In `contracts/emissions_controller/src/instantiate.rs:98` and `contracts/emissions_controller_outpost/src/instantiate.rs:57`, the voting escrow's contract migration admin is set to the emission controller's contract owner. This is problematic because the emission controller's contract owner may be transferred via the `ProposeNewOwner` and `ClaimOwnership` messages, but the previous owner retains contract migration permission. If the previous owner is compromised or malicious, the voting escrow contract can be migrated into a malicious code ID to steal users' funds.

We classify this issue as minor because it can only be caused by the contract owner, which is a privileged account.

Recommendation

We recommend setting the contract migration admin to the assembly governance contract.

Status: Partially resolved

4. Multiple storage states are not exposed through smart queries

Severity: Informational

In multiple contracts, the `query` entry points do not expose some storage state values through smart queries. This forces third-party contracts and nodes to perform a raw query to read the stored value, which is error-prone and decreases user experience.

Occurrences:

- `OWNERSHIP_PROPOSAL` in `contracts/emissions_controller/src/state.rs`
- `OWNERSHIP_PROPOSAL`, `REGISTERED_PROPOSALS` and `PROPOSAL_VOTERS` in `contracts/emissions_controller_outpost/src/state.rs`

Recommendation

We recommend implementing smart queries that expose the storage states mentioned above.

Status: Partially resolved

5. Inconsistent documentation

Severity: Informational

In `contracts/assembly/src/contract.rs:86` a comment states that a `Cw20ReceiveMsg` should be processed. However, the contract code does not contain any logic to support such operation.

Additionally, `CastVoteOutpost ExecuteMsg` is not present in the comments between lines `82` and `104`, describing all available functionalities.

Recommendation

We recommend adding the missing feature or adjusting the comments to reflect the existing implementation.

Status: Resolved