Audit Report

Astroport vxASTRO

v0.2

May 31, 2022

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 Astroport to perform a security audit of vxASTRO.

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 repositories:

https://github.com/astroport-fi/astroport-core

Commit hash: `4a0a4fc619a7549e1f347727d57e17522719f3fb`

https://github.com/astroport-fi/astroport-governance

Commit hash: `0b1a4282b062cc2f9e8fb3684f203e6fad1f9fd2`

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 covers the functionality associated with vxASTRO. vxASTRO allows holders to lock their xASTRO in the vxASTRO pool to receive vxASTRO points, amplify their governance power, receive an additional share of trading fees, and access other benefits such as boosted liquidity mining rewards.

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.

Summary of Findings

No	Description	Severity	Status
1	Astral assembly does not impose negative consequences for creation of low-quality or malicious proposals	Major	Acknowledged
2	Link validation allows for non fully formed domains to be whitelisted	Minor	Partially resolved
3	No minimum value for voting quorum in config validation	Minor	Resolved
4	Unbounded loop could run out of gas	Minor	Resolved
5	New owner proposal minimum and maximum expiry times	Minor	Partially resolved
6	Incorrect attribute emission during pool deactivation	Minor	Resolved
7	Updating the expiration period can lead to unexpected behavior of live proposals	Minor	Resolved
8	Update of factory contract owner may cause inconsistency	Minor	Resolved
9	Blacklist design presents centralization concerns	Informational	Acknowledged
10	Use of magic numbers	Informational	Resolved
11	Duplicate zero amount check is inefficient	Informational	Resolved
12	Contract should return error if boolean parameter is updated to the same value that is currently set	Informational	Resolved

Code Quality Criteria

Criteria	Status	Comment
Code complexity	**Medium-High**	-
Code readability and clarity	**Medium-High**	-
Level of documentation	**High**	-
Test coverage	**Medium-High**	-

Detailed Findings

1. Astral assembly does not impose negative consequences for creation of low-quality or malicious proposals

Severity: Major

The `end_proposal` function in `contracts/assembly/src/contract.rs:330` returns the proposal creator's funds regardless of the outcome of the proposal. This does not provide any disincentive for submission of malicious or spam proposals, which could allow malicious actors to grief the assembly contracts.

Recommendation

We recommend only returning proposal deposits to users if a specific threshold or condition is met, that is lower than the threshold required for a proposal to pass.

Status: Acknowledged

2. Link validation allows for non fully formed domains to be whitelisted

Severity: Minor

The `validate_links` function in `packages/astroport-governance/src/assembly.rs:387` performs validation on whitelisted links during instantiation and config updates. Currently, the validation criteria for links is related to the characters of the string. This is potentially problematic when viewed in conjunction with how proposal links are checked against the whitelist.

The proposal link validation is handled in `packages/astroport-governance/src/assembly.rs:280`. Currently, the whitelisted links are used as a prefix validation for the proposal link. This can be problematic if the whitelist contains entries that are not FDQNs (fully qualified domain names). Specifically, if the link does not contain a TLD (top level domain), it will allow an attacker to create a malicious domain that will still pass the whitelist validation. For example, if the whitelist contains `https://astroport`, attackers could register a domain that shares the same prefix such as `https://astroport-gov.com` or a domain that has a different TLD such as `https://astroport.xyz` to conduct an attack.

Recommendation

We recommend adding validation to the `validate_links` function that ensures that whitelisted links terminate with a top-level domain.

Status: Partially resolved

The Astroport team added an additional validation to ensure that the link provided contains a `..` This condition may also be met if the supplied link contains a subdomain. While the fix provided is sufficient, we advise the Astroport Team and governance to be cognizant of the edge case that may occur if a subdomain is in use.

Example:

Valid Whitelisted Domain: `https://gov.astroport`
Attack proposal Domain: `https://gov.astroport-attack.com`

3. No minimum value for voting quorum in config validation

Severity: Minor

In the assembly contracts in `contract/assembly/src/contract.rs`, the `instantiate` and `update_config` functions apply a maximum quorum size during config validation. However, no validation is performed to ensure that the minimum quorum size is not below a sensible threshold.

This could enable the quorum threshold to be set to a level whereby proposals could be passed without a significant proportion of the total token supply being involved in the proposal.

Recommendation

We recommend adding a minimum threshold for the `proposal_required_quorum` as is performed for the `proposal_required_threshold` during the validation of the config file.

Status: Resolved

4. Unbounded loop could run out of gas

Severity: Minor

The function `calc_claim_amount` found in `contract/escrow_fee_distributor/src/contract.rs:284` uses a loop and break conditions to calculate the claim amounts from the last claimed period to the current period.

In the case that claims are not processed for a long period of time this loop may run out of gas and become unexecutable. However, due to the unlikelihood of this occurring and the fact that it would only affect one user the severity is noted as minor.

Recommendation

We recommend adding an optional break condition to the loop that enables users to process claims in multiple transactions.

Status: Resolved

5. New owner proposal minimum and maximum expiry times

Severity: Minor

The function `propose_new_owner` in `packages/astroport/src/common.rs:34-67` enables the current owner of a contract to propose a new owner who is subsequently able to claim the contract ownership. The original owner specifies an amount of time, `expires_in`, during which the proposal is claimable.

However, there are no minimum or maximum requirements to the amount of time that the proposal can be active for. This could lead to a proposal that expires immediately preventing a proposed owner from claiming or alternatively a claim being active for excessives periods of time.

Recommendation

We recommend that validation of `expires_in` occurs in the function `propose_new_owner` to ensure that proposals are active for a reasonable minimum and maximum lengths of time.

Status: Partially Resolved

The Astroport team has added a maximum time validation.

6. Incorrect attribute emission during pool deactivation

Severity: Minor

The execution of the function `deactive_pool` in `contracts/tokenomics/generator/src/contract.rs` emits an event upon execution. However, the attribute action of that event is incorrectly defined as `setup_pool`. This could lead to event listeners or other queries from being unable to identify blocks that contain `deactive_pool` messages.

Recommendation

We recommend changing the attribute action in line `contracts/tokenomics/generator/src/contract.rs:922` from `setup_pool` to `deactivate_pool`.

Status: Resolved

7. Updating the expiration period can lead to unexpected behavior of live proposals

Severity: Minor

In the function `update_config` found in `contract/assembly/src/contract.rs:532` the expiration period can be updated. Doing so would also alter the expiration period of live proposals. This could have the consequence of live proposals ending much sooner than anticipated and lead to operational issues.

Recommendation

We recommend capturing the proposal expiration in a parameter when the proposal is actually created that would make the existing proposals unaffected if the expiration period changes.

Status: Resolved

8. Update of factory contract owner may cause inconsistency

Severity: Minor

The `execute_create_pair` function in `contracts/factory/src/contract.rs:352` instantiates a new pair contract, with the admin set to `config.owner`. This may be problematic as `config.owner` is an updatable value which may lead to a pair contract's admin being different from the factory contract's owner.

Note, that the scope of this issue is related to migrations of the pair contract. This means that the previous owner of the factory contract will remain the admin of the new pair contract and can perform contract migrations.

This situation is not inherently malicious or exploitable, but it does present a situation where it becomes difficult to manage the migration of multiple pair contracts with potentially many different owners.

An extreme scenario where the private keys of the previous owner are compromised presents room for human error since every pair would need to have the admin manually updated.

Recommendation

While the pair contract is out of the scope of this audit, we recommend analyzing the likelihood and impact of the concerns raised above when considering the admin of the pair contract. It may be worthwhile to implement a mechanism that allows sequential migration of the admin of all pair contracts while all contract functions are disabled.

Status: Resolved

9. Blacklist design presents centralization concerns

Severity: Informational

The `receive_cw20` function in `contracts/voting_escrow/src/contract.rs:447` implements a `blacklist_check` to control access to the `CreateLock`, `ExtendLockAmount`, and `DepositFor` functionality. While this does not pose a direct security threat, it allows the `owner` and `guardian_addr` to control which addresses have voting power. This raises concerns around centralization of the protocol.

Recommendation

We recommend considering removing the ability to blacklist addresses, which currently allows the `owner` or `guardian_addr` to nullify a voter's voting power.

Status: Acknowledged

The Astroport team has stated that backlisting is community governed. A valid proposal with a blacklisting message should successfully pass in order to exclude someone from vxASTRO.

10. Use of magic numbers

Severity: Informational

In the factory contract in `packages/astroport/src/factory.rs:61`, numbers are used without context for calculations – so-called "magic numbers". Usage of magic numbers is problematic since they decrease the maintainability of the codebase.

Recommendation

We recommend replacing magic numbers with constants.

Status: Resolved

11. Duplicate zero amount check is inefficient

Severity: Informational

In the `claim` function in `contracts/escrow_fee_distributor/src/contract.rs:182-216`, the claim amount is validated during the creation of the transfer message to ensure it is not zero. This check is repeated in the function `transfer_token_amount` in line `204`. This is inefficient.

Recommendation

We recommend removing the duplicate zero amount check in line `contracts/escrow_fee_distributor/src/contract.rs:202`.

Status: Resolved

12. Contract should return error if boolean parameter is updated to the same value that is currently set

Severity: Informational

The `update_config` function in `contracts/escrow_fee_distributor/src/contract.rs:377` allows the `owner` to update the contract's configuration parameters. To improve the user experience, the function should return an error if the value of `is_claim_disabled` is updated to the value that is currently set to inform the caller that the value has not been changed.

Recommendation

We recommend adding functionality to confirm that `is_claim_disabled` is changed, and if it is not changed to return an error.

Status: Resolved