Audit Report

Astroport Periphery

v1.0

December 23, 2021

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 Global Partners LLP to perform a security audit of the Astroport Periphery 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 behaviour.
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 repository:

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

Commit hash: `d13df3bdbd2bce99862c63aa7564f8a2e6ebc2b7`

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

Astroport is an automated, decentralised exchange protocol on the Terra blockchain. The scope of this report pertains to the periphery elements of the project including its airdrop, lockdrop and auction contracts. The airdrop contract facilitates an Astro airdrop for active Terra users, lockdrop allows for liquidity to be migrated from Terraswap to Astroport, and the auction contract facilitates Astro-UST Astroport pool initialization during the protocol launch.

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 improve 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 a security audit and vice versa.

Summary of Findings

No	Description	Severity	Status
1	Locked funds will be inaccessible if lockdrop incentives have not been set by the end of the withdrawal window	Major	Resolved
2	Unlocking funds may fail if a user has too many lockup positions	Major	Resolved
3	In the event of a compromised or lost owner key funds will be locked in the contracts forever	Minor	Acknowledged
4	Liquidity migration might be prevented by target contracts	Minor	Acknowledged
5	Issues in generator or downstream reward contracts may lead to users being unable to unlock LP tokens	Minor	Acknowledged
6	Auction contract may enable claims while withdrawal window of lockdrop contract is still open which could lead to an incorrect state	Minor	Resolved
7	Staked Astroport LP will become unclaimable if generator contract is updated	Minor	Resolved
8	Lack of validation on incentives share value can lead to locked funds becoming inaccessible.	Minor	Resolved
9	Updating the Astro token address may lead to an inconsistent state	Minor	Resolved
10	Users that claimed airdrops will miss out on additional airdrops	Minor	Acknowledged
11	Decoupled receipt and setting of Astro airdrops could lead to failing airdrop claims, delegations and withdrawals	Minor	Resolved
12	Mismatch of a generator's base reward and Astro token can lead to failures of reward distribution	Minor	Acknowledged
13	`calculate_weight` will panic if `weekly_divider` is set to zero	Minor	Resolved
14	`calculate_weight` will panic if `duration` is set to zero	Minor	Resolved

15	Auction contract's init timestamp parameter not validated during instantiation	Minor	Resolved
16	Overflow checks not set for release profile in most packages	Informational	Resolved
17	`terraswap_migrated_amount` is set but never read from	Informational	Acknowledged
18	Expire field of CW20 increase allowance message is not set	Informational	Resolved
19	Determining hashing order for Merkle proof verification using hex encoding is inefficient	Informational	Acknowledged
20	Unused functionality	Informational	Resolved

Code Quality Criteria

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

Detailed Findings

1. Locked funds will be inaccessible if lockdrop incentives have not been set by the end of the withdrawal window

Severity: Major

During `DelegateAstroToAuction` and `ClaimRewardsAndOptionallyUnlock` message handlers that can be executed after the withdrawal window, a user's `total_astro_rewards` is calculated in `contracts/lockdrop/src/contract.rs:827` and `910`. That calculation is done in the `update_user_lockup_positions_and_calc_rewards` function, which contains an assertion that `config.lockdrop_incentives` is set and will panic otherwise in `1735`. Lockdrop incentives can only be set by sending the `IncreaseAstroIncentives` message, which will fail after the end of the withdrawal window though. By the time this panic occurs, there is no way to recover from it, which would render any locked funds inaccessible in the lockdrop contract.

The same issue exists in the `query_lockup_info` function in line `1592`.

Likewise, the auction contract contains the same mechanism – during the `ClaimRewards` message handler, a user's `auction_incentive_amount` is calculated after LP shares have been minted in `contracts/auction/src/contract.rs:618`. The `update_user_incentives_and_lp_share` function that performs the calculation contains an assertion that the `astro_incentive_amount` is set and will return an error otherwise in `717`. These incentives can only be set by sending the `IncreaseAstroIncentives` message, which will fail after the LP shares have been minted. By the time this panic occurs, there is no way to recover from it, which implies that any contributed Astro and UST tokens are locked inaccessible in the contract.

We only classify this issue as major since it can only be caused by oversight of the contract owner.

Recommendation

We recommend replacing the `expect` in `contracts/lockdrop/src/contract.rs:1735` and the `ok_or_else` in `contracts/auction/src/contract.rs:717` with an `unwrap_or_default` to prevent panics. Alternatively, the type of `lockdrop_incentives` and `astro_incentive_amount` could be changed from `Option<Uint128>` to `Uint128`.

Status: Resolved

2. Unlocking funds may fail if a user has too many lockup positions

Severity: Major

The `update_user_lockup_positions_and_calc_rewards` function contains unbounded iterations over all lockup positions by a user in `contracts/lockdrop/src/contract.rs:1703` and `1707`. These iterations could run out of gas if a user submits many lockups for different pools and durations. Since the `update_user_lockup_positions_and_calc_rewards` function will be called when a user sends a `DelegateAstroToAuction` or `ClaimRewardsAndOptionallyUnlock` message, the user will be unable to unlock funds. There is currently no way to recover from this issue.

The same issue exists in the `query_user_info` handler in lines `1453` and `1457`.

We only classify this issue as major since it is unlikely that users submit enough lockup positions to cause out of gas issues.

Recommendation

We recommend restricting the number of lockup positions per user. Alternatively, the contract architecture could be changed to store aggregated values rather than computing them through unbounded iterations.

Status: Resolved

3. In the event of a compromised or lost owner key funds will be locked in the contracts forever

Severity: Minor

The only way to retrieve delegated funds from the auction contract after the withdrawal window is closed is through an `InitPool` message sent from the contract owner. In the event that the owner key is compromised or control over it is lost, delegated funds will be stuck in the auction contract forever. If an attacker gains access to the key, they could extort compensation for releasing the funds.

Likewise, the auction contract's `InitPool` message also sends the `EnableClaims` message to the lockdrop contract. That message enables claims by users in `contracts/lockdrop/src/contract.rs:436-442`. As before, a compromised or lost owner key implies that funds will be locked in the lockdrop contract forever.

The same mechanism enables claims in the airdrop contract, in `contracts/airdrop/src/contract.rs:197`. The issue is less problematic here, since only airdropped funds are in the contract, but still, claims will be impossible if the owner key is compromised or lost

Even though a compromised owner account could have other severe consequences, this issue has been added due to the high value that will probably be locked in these contracts. We still classify this issue only as minor since proper management of the owner account is assumed. The operational security of Astroport, which includes proper key management, has not been in scope of this audit though.

Recommendation

We recommend changing the boolean `lp_shares_minted`/`are_claims_allowed`/`are_claims_enabled` values currently used to unlock funds to time locks that allow claims automatically at some time after the withdrawal window/airdrop end has passed.

Status: Acknowledged

The Astroport team intends to take appropriate precautions, e. g. by using a multi-sig.

4. Liquidity migration might be prevented by target contracts

Severity: Minor

The `MigrateLiquidity` message of the lockdrop contract triggers a withdrawal of funds from TerraSwap in `contracts/lockdrop/src/contract.rs:505`. In theory, the target contract's message handler could be upgraded such that the call would always revert. In that case, funds would be stuck in the lockdrop contract, with no way for liquidity migration and no way for users to withdraw their funds.

Recommendation

We recommend making the lockdrop contract upgradeable to fix any incompatibilities. We also recommend implementing a way to withdraw locked TerraSwap LP tokens should no migration be possible until some time in the future.

Status: Acknowledged

The Astroport team intends to make contracts upgradeable.

5. Issues in generator or downstream reward contracts may lead to users being unable to unlock LP tokens

Severity: Minor

In the lockdrop contract's `ClaimRewardsAndOptionallyUnlock` message handler in `contracts/lockdrop/src/contract.rs:948-954` the generator contract is queried to check pending token/reward balances. Any misconfiguration/issue with the generator or proxy rewards contracts (which could be caused by a downstream issue in another protocol

such as Anchor or Mirror) might make this query fail, which would prevent a user from ever unlocking their LP tokens.

We classify this issue as minor since a misconfiguration is unlikely.

Recommendation

We recommend adding a fallback unlock message to the lockdrop contract that only unlocks LP tokens without any reward collection. Since such an unlock call does not interact with any external contracts, it would serve as a safety measure.

Status: Acknowledged

The Astroport team intends to make contracts upgradeable.

6. Auction contract may enable claims while withdrawal window of lockdrop contract is still open which could lead to an incorrect state

Severity: Minor

The auction contract can enable claims by sending the `EnableClaims` message at any time, even before the withdrawal window is closed. If it is still open, a user calling the `handle_claim_rewards_and_unlock_for_lockup` function will lead to `total_astro_rewards` being set at `contracts/lockdrop/src/contract.rs:910`, which might not use final values. The owner could still update a pool, which might impact `state.total_incentives_share` and would change the result of the `total_astro_rewards` calculation.

Recommendation

We recommend adding a condition to `handle_enable_claims` to ensure that the withdrawal window is closed.

Status: Resolved

7. Staked Astroport LP will become unclaimable if generator contract is updated

Severity: Minor

The lockdrop contract's `handle_update_config` function in `contracts/lockdrop/src/contract.rs:270-273`, currently enables the generator contract address to be updated. Changing the generator contract address would make any staked Astroport LP at the time unclaimable after the change. This would effectively reduce all user's balances.

The same issue exists in the auction contract's `handle_update_config` function in `contracts/auction/src/contract.rs:200-204`.

Recommendation

We recommend removing the ability to update the generator contract address once it has been set. The `handle_update_config` function should check if the generator exists, and if it doesn't exist, it should allow for it to be set. For the auction contract, an alternative would be to prevent updates to the `generator_contract` after `is_lp_staked` has been set to `true`.

Status: Resolved

8. Lack of validation on incentives share value can lead to locked funds becoming inaccessible

Severity: Minor

When users try to delegate Astro rewards or claim rewards and optionally unlock funds, their Astro rewards are calculated in the `update_user_lockup_positions_and_calc_rewards` function. That function internally calls `calculate_astro_incentives_for_lockup`, which will panic in `contracts/lockdrop/src/contract.rs:1666` if `total_incentives_share` is zero. This cannot be recovered from, since pools can neither be added nor updated after the withdrawal window is closed. Consequently, any funds in the lockdrop contract will become inaccessible.

We only classify this issue as minor since it is caused by a misconfiguration by the owner.

Recommendation

We recommend asserting in both the `handle_initialize_pool` and `handle_update_pool` functions that `total_incentives_share` is not zero.

Status: Resolved

9. Updating the Astro token address may lead to an inconsistent state

Severity: Minor

The lockdrop's `handle_update_config` function currently enables the Astro token address `astro_token` to be updated in `contracts/lockdrop/src/contract.rs:253-256`. Changing the Astro token address after users have called `DelegateAstroToAuction` may lead to an inconsistent

state between the stored values in the lockdrop contract and the balances in the Astro token contract.

Recommendation

We recommend removing the ability to update the Astro token address once it has been set. The `handle_update_config` function should check if `astro_addr` exists, and if it doesn't exist, it should allow for it to be set.

Status: Resolved

10. Users that claimed airdrops will miss out on additional airdrops

Severity: Minor

The current implementation of airdrop claims allows only one claim per user due to the condition in `contracts/airdrop/src/contract.rs:255`. At the same time, the contract owner has the ability to update Merkle roots in `160-162`. Taken together, if a user was assigned additional airdrops, they would not be able to claim those additional airdrops. Even without an update, the current design does not allow a user to claim multiple airdrop leaves from the same Merkle tree.

Recommendation

We recommend changing the check whether a user already claimed an airdrop to compare the amount they already claimed with the amount they try to claim. That will allow users that claimed in the past to claim higher amounts if the Merkle tree is updated.

Status: Acknowledged

The Astroport team does not intend to distribute multiple airdrops through the current airdrop contract.

11. Decoupled receipt and setting of Astro airdrops could lead to failing airdrop claims, delegations and withdrawals

Severity: Minor

The current implementation of the airdrop contract decouples receipt of Astro tokens and the setting of the amount to be distributed. That could cause the contract's Astro balance to be insufficient for Astro transfers to succeed. Examples are:

- The `handle_claim` function in `contracts/airdrop/src/contract.rs:267`. If the Astro balance is too low, claiming the airdrop will fail.
- The `handle_delegate_astro_to_bootstrap_auction` function in line `322`. If the Astro balance is too low, delegating airdrops will fail.

- The `handle_withdraw_airdrop_rewards` function in line `377`. If the Astro balance is too low, withdrawing the airdrop will fail.

Recommendation

We recommend using a CW20 receive hook to couple receipt of Astro tokens and setting the amount to be airdropped together. That pattern is used in both the lockdrop and auction contracts in `contracts/lockdrop/src/contract.rs:143` and `contracts/auction/src/contract.rs:113`.

Status: Resolved

12. Mismatch of a generator's base reward and Astro token can lead to failures of reward distribution

Severity: Minor

In the lockdrop contract's `update_pool_on_dual_rewards_claim` function, the generated astro per share is aggregated in `contracts/lockdrop/src/contract.rs:1067`. That function is querying the balance of `rwi.base_reward_token`, implicitly assuming that that token equals the Astro token. If that is not the case, different tokens would be summed up into one state variable, which would eventually lead to issues when distributing rewards.

The same issue exists in `contracts/lockdrop/src/contract.rs:969` as well as in the auction contract's `update_state_on_reward_claim` function in `contracts/auction/src/contract.rs:934` and in `664`.

We classify this issue as minor since a mismatch of the tokens can only be caused by a misconfiguration.

Recommendation

We recommend asserting that `rwi.base_reward_token` equals the Astro token in the instances mentioned above and skipping reward distribution otherwise.

Status: Acknowledged

13. `calculate_weight` will panic if `weekly_divider` is set to zero

Severity: Minor

The `calculate_weight` function will panic in `contracts/lockdrop/src/contract.rs:1680` if `weekly_divider` is equal to zero.

Recommendation

We recommend implementing validation to the `instantiate` function to ensure that `weekly_divider` is greater than zero.

Status: Resolved

14. `calculate_weight` will panic if `duration` is set to zero

Severity: Minor

The `calculate_weight` function will panic in `contracts/lockdrop/src/contract.rs:1679` if `duration` is equal to zero.

Recommendation

We recommend implementing validation to the `instantiate` function to ensure that `min_lock_duration` is greater than zero.

Status: Resolved

15. Auction contract's init timestamp parameter not validated during instantiation

Severity: Minor

While the lockdrop contract `contracts/lockdrop/src/contract.rs:39-49` does perform a validation step to ensure that the `init_timestamp` is in the future, the auction contract does not.

Recommendation

We recommend adding validation to `init_timestamp` in the same way that validation is performed in `lockdrop/contract.rs:39-44`.

Status: Resolved

16. Overflow checks not set for release profile in most packages

Severity: Informational

The following `Cargo.toml` files do not enable overflow-checks for the release profile:

- `contracts/airdrop/Cargo.toml`
- `contracts/auction/Cargo.toml`
- `contracts/lockdrop/Cargo.toml`

- `packages/astroport_periphery/Cargo.toml`

Recommendation

Even though this check is implicitly applied to all packages from the workspace's `Cargo.toml`, we recommend also explicitly enabling overflow checks in every individual package. That helps prevent unintended consequences when the codebase is refactored in the future.

Status: Resolved

17. `terraswap_migrated_amount` is set but never read from

Severity: Informational

`terraswap_migrated_amount` is set in `contracts/lockdrop/src/contract.rs:558` but it is never read from.

Recommendation

We recommend removing this field from `MigrationInfo`. It would be more efficient to emit this value as an attribute rather than storing it.

Status: Acknowledged

The Astroport team states that `terraswap_migrated_amount` is stored for informational purposes and is read during pool info queries.

18.Expire field of CW20 increase allowance message is not set

Severity: Informational

In several places in the codebase, a `Cw20ExecuteMsg::IncreaseAllowance` message is send with the `expires` field set to `None`. It is best practice to limit any allowances as much as possible. The following are occurrences of this issue:

- `contracts/lockdrop/src/contract.rs:612`
- `contracts/lockdrop/src/contract.rs:1350`
- `packages/astroport_periphery/src/helpers.rs:101`

Recommendation

We recommend setting the expiration to the next block.

Status: Resolved

19. Determining hashing order for Merkle proof verification using hex encoding is inefficient

Severity: Informational

In `contracts/airdrop/src/crypto.rs:27-28` the hashing order for Merkle proof verification is determined by hex encoding the hashes and then comparing the hex strings. That is inefficient and wastes gas.

Recommendation

We recommend implementing the Ord trait instead (see https://doc.rust-lang.org/std/cmp/trait.Ord.html#how-can-i-implement-ord) and comparing the bytes directly.

Status: Acknowledged

20. Unused functionality

Severity: Informational

Some of the functions in the codebase are unused:

- The `option_string_to_addr` function in `packages/astroport_periphery/src/helpers.rs:51`
- The `get_denom_amount_from_coins` function in `packages/astroport_periphery/src/helpers.rs:63`
- The `zero_address` function in `packages/astroport_periphery/src/helpers.rs:107`

Recommendation

We recommend removing unused functionality.

Status: Resolved