Audit Report

Eclipse Equinox Lockdrop Contract

v1.0

July 30, 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 Luna Launcher Protocol Inc dba “Eclipse Fi” to perform a security audit of the Eclipse Equinox Lockdrop contract.

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/EclipsePad/equinox-contracts
Commit	`0541fb7d6d9715add8eff5c25880bdd97209bc6b`
Scope	The scope was restricted to the following files: • `contracts/lockdrop` • `packages/equinox_msg/src/lockdrop.rs`
Fixes verified at commit	`6742137c4c20f25c303326210fba010abcc5244f` 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 Eclipse Equinox Lockdrop contract allows users to stake their tokens with different lockup durations and receive external lockdrop incentives and staking 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.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	Medium-High	Complex reward mechanisms are implemented for single-sided and LP token staking, which integrates with other contracts. Additional features such as early withdrawal penalties, enforced one-time withdrawals, bonus multiplier, and vested rewards also increase the complexity.
Code readability and clarity	Medium	-
Level of documentation	Medium-High	Documentation is available in `contracts/lockdrop/README.md`, the whitepaper, and a blog post.
Test coverage	Medium	`cargo tarpaulin` reports a coverage of `72.39%`.

Summary of Findings

No	Description	Severity	Status
1	Stakers can steal funds by providing duplicate assets	Critical	Resolved
2	Rewards are not recorded by the global index, causing a loss of rewards for stakers	Critical	Resolved
3	Incorrect penalty amount possible	Minor	Resolved
4	Contract ownership cannot be transferred	Minor	Partially Resolved
5	Missing configuration validation	Minor	Resolved
6	Incorrect and misleading events emitted	Minor	Resolved
7	Inconsistent lockdrop ending period definition	Minor	Resolved
8	Unnecessary code in `Cw20HookMsg::IncreaseIncentives`	Informational	Resolved
9	Usage of magic numbers decreases maintainability	Informational	Resolved
10	Unused code	Informational	Resolved
11	Contract can be migrated to the same version	Informational	Resolved

Detailed Findings

1. Stakers can steal funds by providing duplicate assets

Severity: Critical

In `contracts/lockdrop/src/entry/execute.rs:315-349`, the `try_claim_rewards` and `try_claim_all_rewards` functions implement the optional `assets` parameter if the staker wants to withdraw specific assets. The issue is that if duplicate assets are provided, stakers can withdraw more rewards than intended due to the loops in lines `1362`, `1493`, `1658`, and `1810`.

Consequently, rewards from other stakers will be paid out, triggering an insufficient balance error when other stakers unlock their staked `eclipASTRO` and liquidity pool tokens, causing a loss of funds.

Recommendation

We recommend modifying the `try_claim_rewards` and `try_claim_all_rewards` functions to dedupe the `assets` parameter to ensure duplicate assets are removed.

Status: Resolved

2. Rewards are not recorded by the global index, causing a loss of rewards for stakers

Severity: Critical

In `contracts/lockdrop/src/entry/execute.rs:1593-1604`, the `_claim_all_single_sided_rewards` function does not update the `SINGLE_STAKING_REWARD_WEIGHTS` state if the reward duration does not match the specified `durations`. This is problematic because when the rewards are claimed from the staking contract in lines `1571-1582`, the global index must be increased accordingly to distribute the rewards to the stakers. An example of a correct implementation can be found in `contracts/lockdrop/src/entry/execute.rs:1345-1352`.

Consequently, the withdrawn rewards are not distributed to the stakers, causing a loss of rewards.

Recommendation

We recommend updating the `SINGLE_STAKING_REWARD_WEIGHTS` state before checking whether the reward duration matches the specified `durations` parameter.

Status: Resolved

3. Incorrect penalty amount possible

Severity: Minor

In `contracts/lockdrop/src/entry/execute.rs:2046`, the `_unlock_lp_lockup` function computes the penalty amount to be half of the withdrawn amount. This is incorrect because the penalty amount should be computed with the `early_unlock_penalty_bps` value defined in `packages/equinox_msg/src/lockdrop.rs:231`.

If the contract instantiator configured a different `Some(msg.lock_configs)` value in `contracts/lockdrop/src/entry/instantiate.rs:47`, the penalty amount will be incorrect.

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

Recommendation

We recommend computing the penalty amount according to the lock configuration in `packages/equinox_msg/src/lockdrop.rs:211`.

Status: Resolved

4. Contract ownership cannot be transferred

Severity: Minor

The contract within the scope of this audit does not expose any entry points for the owner to transfer ownership. This is problematic because if the contract owner is compromised, there is no way to recover.

Recommendation

We recommend implementing a two-step ownership transfer, which allows the current owner to propose a new owner, and then the account that is proposed as the new owner calls a function to claim ownership and execute the config update.

Status: Partially Resolved

This issue is partially resolved because only a one-step ownership transfer has been implemented instead of the recommended two-step ownership transfer.

5. Missing configuration validation

Severity: Minor

The `lockdrop` contract does not perform validation on several of its configuration fields in the following locations:

- `instantiation` in `contracts/lockdrop/src/entry/instantiate.rs:18-70`
- `try_update_config` in `contracts/lockdrop/src/entry/execute.rs:38-54`
- `try_update_reward_distribution_config` in `contracts/lockdrop/src/entry/execute.rs:56-78`

The affected fields are the following:

- The addresses are not validated using `addr_validate` to ensure the addresses are in the correct format:
    - `dao_treasury_address`
    - `eclipastro_token`
    - `converter`
    - `single_sided_staking`
    - `lp_staking`
    - `astro_staking`
- The `eclip` and `bclip` assets may be native or CW20 tokens from the `AssetInfo` enum:
    - If they are native assets, the denom should be validated with `BankQuery::Supply` to ensure the supply is larger than zero, meaning the denom exists.
    - If they are CW20 token addresses, `addr_validate` should be called on the address.
- The `lock_configs` vector holds individual lock configurations that consist of `duration`, `multiplier`, and `early_unlock_penalty` fields:
    - The `lock_configs` vector should be validated to ensure no lock configurations use the same `duration` field. Entries with repeated durations make elements different than the first to be unreachable when performing value matching or duplicate entries in storage pieces to be modified several times when loading data based on the duration key by iterating through the configs.
- The `deposit_window` and `withdrawal_window` fields should be validated to contain a minimum duration. For example, the protocol will not work properly if the fields are misconfigured as zero.

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

Recommendation

We recommend thoroughly validating the affected items as described above.

Status: Resolved

6. Incorrect and misleading events emitted

Severity: Minor

The following instances illustrate incorrect or misleading events emitted:

- The `migrate` function in `contracts/lockdrop/src/contract.rs:132` emits the contract name as "*new_contract_name*". This is misleading because the contract name does not change throughout the migration due to the validation in `contracts/lockdrop/src/contract.rs:113-117`.

  Consider modifying the attribute key to "*contract_name*" to prevent confusion.

- In `contracts/lockdrop/src/entry/execute.rs:1220`, the `amount` attribute is emitted incorrectly as `eclipastro_amount_to_deposit`. This is incorrect because the `xASTRO` amount value will be reflected as the `eclipASTRO` deposit amount.

  Consider modifying the attribute value to emit `xastro_amount / Uint128::from(2u128)` following `contracts/lockdrop/src/entry/execute.rs:1046`.

- In `contracts/lockdrop/src/entry/execute.rs:1230`, the `amount` attribute is emitted incorrectly as `eclipastro_amount_to_deposit`. This is incorrect because the `xASTRO` amount value will be reflected as the `eclipASTRO` deposit amount.

  Consider modifying the attribute value to emit `xastro_amount_to_deposit`.

- In `contracts/lockdrop/src/entry/execute.rs:261`, the action emitted is to "convert `ASTRO` to `xASTRO`". This is incorrect because the action is converting `ASTRO` to `eclipASTRO`.

  Consider modifying the attribute value to "*convert ASTRO to eclipASTRO*".

- In `contracts/lockdrop/src/entry/execute.rs:50`, the attribute key is emitted as "*new_timelock_staking*". This is incorrect because the updated configuration is the DAO treasury address.

Consider modifying the attribute key to "*new_dao_treasury_address*".

Recommendation

We recommend applying the changes mentioned above.

Status: Resolved

7. Inconsistent lockdrop ending period definition

Severity: Minor

The `lockdrop` contract checks whether both the deposit and withdraw windows have ended in several places throughout the codebase, defined as "Lockdrop is finished". However, this definition has an "off by one" inconsistency among the different instances.

In `contracts/lockdrop/src/entry/execute.rs:300`, the `try_stake_to_vaults` function considers the lockdrop has finished if the current timestamp is larger than the time window (`cfg.init_timestamp + cfg.deposit_window + cfg.withdrawal_window`). If the current time equals the time window, the `LockdropNotFinished` error will be triggered, indicating that the lockdrop has not ended.

However, other instances consider this differently. Lines `237`, `784`, `823`, `1889`, and `1989` consider the lockdrop to have ended when the current time equals the time window.

Although we did not find a security vulnerability introduced by this inconsistency, inconsistency in the time window definition affects maintainability and could have unforeseen consequences in future versions.

Recommendation

We recommend clearly defining the ending period of the lockdrop. In addition, consider creating a standalone function that checks if the lockdrop has finished, reducing the likelihood of error when enforcing this condition in different functions.

Status: Resolved

8. Unnecessary code in `Cw20HookMsg::IncreaseIncentives`

Severity: Informational

In `contracts/lockdrop/src/entry/execute.rs:799-809`, the `Cw20HookMsg::IncreaseIncentives` message attempts to increase the incentive amount based on the native tokens sent in `info.funds` on top of

`Cw20ReceiveMsg.amount`. This is unnecessary because the `Cw20ReceiveMsg` message does not include native tokens when dispatching the `Send` message.

Recommendation

We recommend removing `contracts/lockdrop/src/entry/execute.rs:799-809` and instead use the `try_increase_incentives` function to increase native tokens incentive.

Status: Resolved

9. Usage of magic numbers decreases maintainability

Severity: Informational

Throughout the codebase, hard-coded number literals without context or a description are used. Using such "magic numbers" goes against best practices as they reduce code readability and maintenance as developers are unable to easily understand their use and may make inconsistent changes across the codebase.

Instances of magic numbers are listed below:

- `contracts/lockdrop/src/math.rs:38`
- `contracts/lockdrop/src/entry/query.rs:425`

Recommendation

We recommend defining magic numbers as constants with descriptive variable names and comments, where necessary.

Status: Resolved

10. Unused code

Severity: Informational

The `lockdrop` contract includes several instances of unused code:

- The `send_native_token_msg` function in `contracts/lockdrop/src/entry/execute.rs:2113-2118`.
- The following errors defined in `contracts/lockdrop/src/error.rs`: `AlreadyAllowed`, `AlreadySet`, `AlreadyUnlocked`, `FailedToParseReply`, `InsufficientAmountInContract`, `InvalidLockupAsset`, `LpStakingNotHappend`, `NoDeposit`, `NoEclipReward`, `OnlyEclipAllowed`, `OwnershipProposalExpired`, `RelockNotAllowed`, `StakingNotHappend`, `UnknownReplyId`, and `WaitToUnlock`.

- The `contracts/lockdrop/src/querier.rs` file is not used and not referenced in the `lib.rs` file.

Although we did not find a security vulnerability introduced by unused code, it reduces maintainability and readability.

Recommendation

We recommend removing any unused code.

Status: Resolved

11. Contract can be migrated to the same version

Severity: Informational

The `lockdrop` contract includes the ability to migrate to another version of the contract by calling the `migrate` function defined in `contracts/lockdrop/src/contract.rs:105`.

However, the `ensure_eq!` macro in `contracts/lockdrop/src/contract.rs:122` allows a migration where the new contract version can be equal to the current contract version, essentially migrating to the same code ID.

Recommendation

We recommend modifying the `ensure_eq!` implementation to `version > storage_version` to ensure the contract can only be migrated to a newer version. Additionally, consider removing the redundant if statement in `contracts/lockdrop/src/contract.rs:127`.

Status: Resolved