Eden

Security review

Version 1.0

Reviewed by
nmirchev8
deth

Table of Contents

1 About Egis Security

Egis Security is a team of experienced smart contract researchers, who strive to provide the best smart contract security services possible to DeFi protocols.

The team has a proven track record on public auditing platforms like Code4rena, Sherlock, and Cantina, earning top placements and rewards exceeding $170,000. They have identified over 150 high and medium-severity vulnerabilities in both public contests and private audits.

2 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

3.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

3.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

4 Executive summary

Overview

Project Name	Eden
Repository	Private
Commit hash	087fe03c959cdb93f2093b402b436bd62deb4bb2
Resolution	c54dc1191f2e3ef74638737a4cb209bcb91c8e4e
Documentation	https://eden-2.gitbook.io/eden
Methods	Manual review

Scope

eden-contracts/src/Airdrop.sol
eden-contracts/src/Migrator.sol
eden-contracts/src/Staking.sol

Issues Found

Critical risk	0
High risk	0
Medium risk	2
Low risk	1
Informational	0

5 Findings

5.1 Medium risk

5.1.1 On `distribute/distributeToSpecificPool` call, rewards should be updated before distributing new tokens

Severity: *Medium risk*

Context: Staking.sol#L437-L447

Description: When `distribute`/`distributeToSpecificPool` is called we first update the corresponding pools and then call `updateRewardsIfNecessary()`, which may result in counting the distributed tokens for the wrong pool, if the cycle has already ended.

Recommendation: First call `updateRewardsIfNecessary` and then distribute the funds, which are entering the contract.

Resolution: Fixed

5.1.2 Staking bonus mechanics differs from documentation

Severity: *Medium risk*

Context: Staking.sol#L302-L320

Description: Current staking bonus implementation is as follows:

- 40 - 90 days -> from 0 - 5%
- 90 - 365 -> from 5% - 10%
- 365 - 730 -> from 10% - 20%
- 730 - 1480 -> from 20% - 30%

But in the docs it is stated thah “5% minimum to 100% maximum Scaling in proportion to the Days you Stake.”

Recommendation: Implement linear intepulation from 5% to 100% based on the staking duration.

Resolution: Fixed

5.2 Low risk

5.2.1 Implement logic to call `lotusStaking.compoundRewards` in Migrator.sol

Severity: *Low risk*

Context: Migrator.sol#L19

Description: Consider implementing logic to call `lotusStaking.compoundRewards` in `Migrator.sol` so the accured rewards are swapped for lotus and staked again, which will increase the `totalShares` amount.

Resolution: Acknowledged