code4rena

Celo

Smart Contract Security Assessment

Version 1.0

Audit dates: Jul 25 — Jul 29, 2024

Audited by: MiloTruck
HollaDieWaldFee100

Contents

1. Introduction

1.1 About Zenith

Zenith is an offering by Code4rena that provides consultative audits from the very best security researchers in the space. We focus on crafting a tailored security team specifically for the needs of your codebase.

Learn more about us at https://code4rena.com/zenith.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2. Executive Summary

2.1 About Celo

Our mission is to build a regenerative digital economy that creates conditions of prosperity for all.

2.2 Scope

Repository	celo-org/staked-celo
Commit Hash	a792d673a514f43bb2d7af75fb07043c582f25db

2.3 Audit Timeline

DATE	EVENT
Jul 25, 2024	Audit start
Jul 29, 2024	Audit end
Oct 29, 2024	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	0
Informational	1
Total Issues	1

3. Findings Summary

ID	DESCRIPTION	STATUS
I-1	Inconsistent vote accounting between strategies after upgrade to `changeStrategy()`	Acknowledged

4. Findings

4.1 Informational

A total of 1 informational findings were identified.

[I-1] Inconsistent vote accounting between strategies after upgrade to `changeStrategy()`

Severity: Informational	Status: Acknowledged

Context:

- Manager.sol#L369-L373

Description:

PR #204 - introduces a change to `changeStrategy()`, where the amount of stCelo transferred from the old strategy to the new strategy now includes the caller's locked vote balance:

```
- uint256 stCeloAmount = stakedCelo.balanceOf(msg.sender);
+ uint256 stCeloAmount = stakedCelo.balanceOf(msg.sender) +
+     stakedCelo.lockedVoteBalanceOf(msg.sender);
  if (stCeloAmount != 0) {
      _transfer(strategies[msg.sender], newStrategy, stCeloAmount);
  }
```

However, if a user calls `changeStrategy()` before the upgrade, their accounting between strategies becomes inconsistent after the upgrade to the `Manager` contract is performed.

Consider the following scenario:

- Assume a user has a portion of their stCelo locked in votes in strategy A.
- A user calls `changeStrategy()` to switch to strategy B before the upgrade.
- The upgrade is performed.
- The user attempts to call `changeStrategy()` to switch to strategy C.

When the user calls `changeStrategy()` post-upgrade, `_transfer()` is called to move `stCeloAmount` from strategy B to strategy C. However, a portion of `stCeloAmount` is still in strategy A as locked votes, so the transfer is not possible and will revert.

Recommendation:

The team has implemented a follow-up PR that addresses inconsistent vote accounting.

Celo: It is not a problem that accounting become non-consistent after shipping a fix, because we could fix it with the balance with a follow up governance proposal calling `updateGroupStCelo()` in PR #205.

Zenith: Acknowledged as a non-issue.

While the balance discrepancy for users caused by the previous bug has been addressed, the team should consider the case where a user is intentionally malicious and tries to mess up the accounting. For example, they could front-run or back-run the upgrade and/or the subsequent governance proposal.