Celo Contracts Audit

Release 7 – Part 2

OPENZEPPELIN SECURITY | JUNE 16, 2022

Security Audits

Table of Contents

Summary

Type

Languages
Solidity

Total Issues
5 (5 resolved)

Scope

We reviewed all changes to production Solidity files in the following pull requests:

- PR #9252 up to commit 15f6b6d
- PR #9367 up to commit 0003e85
- PR #9369 up to commit 2d73adc

The following contracts were in scope:

- PR #9252
  - packages/protocol/contracts/stability/Exchange.sol
- PR #9367
  - packages/protocol/contracts/stability/Reserve.sol
- PR #9369
  - packages/protocol/contracts/governance/LockedGold.sol

Overview of changes

A summary of the changes in the pull requests:

- PR #9252 – Implements bounds checks on new spread values to ensure they are less than or equal to 1 in `setSpread`
- PR #9367 – Removes the requirement for an active oracle when adding new stablecoins with `addToken`
- PR #9369 – Implements a `getPendingWithdrawal` function in order to allow single withdrawal lookups

These changes, along with those from Part 1 of this audit, comprise cLab's `Release 7` for the celo-monorepo.

Low Severity

Missing oracle can cause unexpected behavior

Prior to pull request #9367, the `addToken` function in the `Reserve` contract checked that an oracle exists for the token being added, and the oracle returns a non-zero exchange rate. These checks ensured that every token in the `_tokens` array had a corresponding oracle.

With the oracle checks removed from `addToken` by this pull request, it is possible to enter a state where the token has been added but the oracle doesn't exist yet. However, the `getReserveRatio` function in the `Reserve` contract assumes that every token has a corresponding oracle that returns non-zero exchange rate values. If no oracle exists for a specific token, the converted price calculation will result in a divide-by-zero error.

The following contracts also contain code which can incorrectly divide by zero if queried with an oracle that is returning zero:

- `getGasPriceMinimum` of `GasPriceMinimum`
- `getTargetTotalEpochPaymentsInGold` of `EpochRewards`

Consider implementing additional logic that excludes tokens without oracles from the reserve ratio calculation, as well as including checks to ensure helpful errors are thrown rather than divide-by-zero errors.

Update: *Partially fixed. The* `getReserveRatio` *function was fixed in PR #9527. Both* `getGasPriceMinimum` *and* `getTargetTotalEpochPaymentsInGold` *remain unchanged. Celo's statement for this issue:*

> *Updating* `getGasPriceMinimum` *is not critical, as the only reasonable behavior when there is no oracle report is to revert (could be nicer to fail with a relevant require message, but this is an edge case). Updating* `getTargetTotalEpochPaymentsInGold` *is not necessary as it only ever converts from cUSD (not other cStables), which already has an oracle rate.*

Incorrect version number

Consider incrementing the patch number returned by `getVersionNumber` in order to adhere to the smart contract release process.

Update: *Fixed in PR #8334.*

Notes & Additional Information

Documentation mismatch

The online Celo docs state that before fully activating a new stable token, it is required to have at least one oracle report. The documentation points to line number 223 in the `Reserve` contract's `addToken` function as the enforcer of this requirement. Pull request #9367 removed that code from the `Reserve` contract, invalidating the online documentation.

Consider updating the online Celo documentation to accurately reflect the new behavior of the `addToken` function.

Update: *Fixed in PR #312 of the celo-org/docs repository.*

Inconsistent code

In pull request #9252, code was added in order to ensure newly set spread values are valid. The change updated the `setSpread` function in the `Exchange` contract.

Similar to the `Exchange` contract, the `GrandaMento` contract includes an implementation of `setSpread` which already has a bounds check, however the two implementations differ in terms of logic and error messages.

In favor of consistent code across the repository, consider updating the code to make both implementations match.

Update: *Fixed in PR #9459.*

Inconsistent test coverage

for `getPendingWithdrawal` was added in `lockedgold.ts`, but the corresponding test for `getPendingWithdrawals` was removed in the process, even though this function is still in use. Furthermore, there are not matching test cases for both functions.

To improve code coverage, consider restoring the test that was removed and providing equivalent test cases for both functions.

Update: *Fixed in PR #9460.*

Conclusions

No critical or high severity issues have been found. Recommendations and fixes have been proposed to improve code quality, minimize errors, and address uncommon but possible operating conditions which could result in error.

Related Posts

Beefy Zap Audit

BeefyZapRouter serves as a versatile intermediary designed to execute users' orders through routes...

OpenBrush Contracts Library Security Review

OpenBrush is an open-source smart contract library written in the Rust programming language and the...

Linea Bridge Audit

Linea is a ZK-rollup deployed on top of Ethereum. It is designed to be EVM-compatible and aims to...

OpenZeppelin