Celo Contracts Audit – Phase 7

OPENZEPPELIN SECURITY | MARCH 23, 2021

Security Audits

Introduction

After a seventh phase of auditing, the cLabs team asked us to review and audit the recent changes in the smart contracts and scripts of their protocol.

Scope

The audited commit for this phase was `1c391af75e37da1108f426b5ff14b4766538c79e` of the Celo Monorepo. This commit has associated tag celo-core-contracts-v4.pre-audit.

To ensure that the audit process was complete and no unaudited code was added to the contracts, we audited the difference between this commit and the previous phase's audited commit `64e618b9b856073305dd5748fc04fc772ff72714`, tagged `celo-core-contracts-v3.staging`, `celo-core-contracts-v3.rc0`, and `celo-core-contracts-v3.baklava`, that also contains fixes from the previous audit round.

In addition to the change to the contracts, changes to previously audited scripts have been taken into account.

The changed files, in which the diff between commit `1c391af75e37da1108f426b5ff14b4766538c79e` and `64e618b9b856073305dd5748fc04fc772ff72714` was audited, were:

```
packages/protocol/contracts/common/MetaTransactionWallet.sol
packages/protocol/lib/compatibility/report.ts
packages/protocol/scripts/build.ts
packages/protocol/lib/compatibility/utils.ts
packages/protocol/scripts/bash/make-release.sh
packages/protocol/scripts/bash/check-versions.sh
packages/protocol/scripts/bash/verify-release.sh
packages/protocol/scripts/bash/verify-deployed.sh
packages/protocol/scripts/bash/release-lib.sh
```

Anything not listed above was considered outside the scope for this audit. Note that while there may be some references to out-of-scope files in this report, these files should not be considered as audited.

Overview of the changes

For this phase of auditing, the cLabs team introduced a recovery mechanism to the `MetaTransactionWallet`, enhanced the backwards compatibility report used in the release-process, and made some bug fixes.

Vulnerabilities

Below, we list all vulnerabilities found in this audit phase of the Celo codebase.

Update

Most of the following issues have been either fixed or acknowledged by the Celo team. Our analysis of the mitigations is limited to the specific changes made to cover the issues, and disregards all other unrelated changes in the codebase.

Critical severity

None.

Medium severity

None.

Low severity

Function variables could corrupt global variables

The helper function `build_tag` of the `release-lib.sh` script sets the values of its `BRANCH`, `LOG_FILE`, and `BUILD_DIR` variables. In Bash, by default all variables are global. Also, Bash functions cannot explicity return strings. So in the scripts that call `build_tag`, their `BUILD_DIR` variable is understood to take on the value set within the `build_tag` function.

Currently, the `BRANCH` and `LOG_FILE` variables are not used in scripts after their call to `build_tag`. But in future development to scripts using `BRANCH` and `LOG_FILE` variables, calling `build_tag` could unintentionally corrupt their values.

Consider defining the `BRANCH` and `LOG_FILE` variables within `build_tag` to be `local` since they are not intended to be used outside the function body.

Update: *This has been fixed in commit c273843.*

Release notes inconsistent with release

The effects of PR's 6899, 6850, and 7344 were listed in the release notes as features of the release. But also included in this release is PR7309 which undoes the changes made by `6899`, `6850`, and the git history of `7344` is not even included in this branch. This means that the effects of PR's `6899`, `6850`, and `7344` do not appear in this release.

While the effects of these PR's are fairly innocuous, the release-process of the Celo protocol relies on a governance process where the information in the release notes could sway the electorate. If the effects of the PR's were less innocuous, such as a patch to a critical vulnerability, their absence in the release approved by the governance process could leave an attack vector open.

Consider paying special attention that future release notes correctly document the content of the release.

Update: *This has been fixed by removing description of the effects of PR's* `6899`, `6850`, *and* `7344` *from the release notes.*

Notes & Additional Information

Commented out code

Lines 87 and 128 of the `build.ts` script include commented out lines of code without giving developers enough context on why those lines have been discarded, thus providing them with little to no value at all.

As the purpose of these lines is unclear and may confuse future developers and external contributors, consider removing them from the codebase. If they are to provide alternate implementation options, consider extracting them to a separate document where a deeper and more thorough explanation could be included.

Update: *This has been fixed in commit 125e3a6.*

`guardian` not set on initialization

This release brought the addition of the owner settable `guardian` address to the `MetaTransactionWallet`, whom has the ability to recover the wallet by way of the `recoverWallet` function. This `guardian` address is set by the `setGuardian` function which has modifier `onlyOwner`. The `owner` is set by `deploy` function of the `MetaTransactionWalletDeployer`. If control of the `owner` account that is set during deployment is lost before the `setGuardian` function is called then the wallet will be unrecoverable.

To make the recoverability of the `MetaTransactionWallet` more robust, consider setting the `guardian` address in the `initialize` function.

Naming issue hinders code understanding and readability

To favor explicitness and readability, a script may benefit from better naming. Consider implementing the following:

- Changing `BUILD_DIR_1` to `OLD_BRANCH_BUILD_DIR`.
- Changing `BUILD_DIR_2` to `NEW_BRANCH_BUILD_DIR`.

Update: *This has been fixed in commit 947a176.*

Typographical error

In line 5 of `report.ts`, there is an unneeded comma in the `import` statement.

Update: *This has been fixed in commit 7132ca1.*

Conclusions

Two low severity issues were found. Some changes were proposed to follow best practices and reduce potential attack surface.

Update: *The low severity issues have been fixed. Many of the recommendations have been acknowledged or incorporated.*

Related Posts

Beefy
Zap Audit

BRUSHFAM
OpenBrush Contracts Library Security Review

Linea
Bridge Audit

Beefy Zap Audit

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

Security Audits

OpenBrush Contracts Library Security Review

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

Security Audits

Linea Bridge Audit

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

Security Audits