Celo Contracts Audit – Release 6

OPENZEPPELIN SECURITY | FEBRUARY 23, 2022

Security Audits

In another round of auditing, the cLabs team asked OpenZeppelin to review and audit recent changes to the core contracts of the Celo protocol.

Scope

This audit was a diff audit, meaning that the scope of the audit was the difference across files in certain pull requests. The pull requests audited in this phase were 8300, 8349, 8360, 8475, 8750, 8751, 8831, and 8854. Note that the audit covered production Solidity files as well as upgrade deployment scripts. A more detailed scope is below.

Within the various pull requests, the following files were considered in-scope for this audit:

Within `PR #8300`

- `packages/protocol/contracts/stability/GrandaMento.sol`

Within `PR #8349`

- `packages/protocol/contracts/common/Accounts.sol`
- `packages/protocol/contracts/common/MetaTransactionWallet.sol`
- `packages/protocol/contracts/common/Signatures.sol`
- `packages/protocol/contracts/common/linkedlists/AddressLinkedList.sol`

`edListWithMedian.sol`

- `packages/protocol/contracts/common/linkedlists/IntegerSortedLinkedList.sol`
- `packages/protocol/contracts/governance/Election.sol`
- `packages/protocol/contracts/governance/Governance.sol`
- `packages/protocol/contracts/governance/Proposals.sol`
- `packages/protocol/contracts/identity/Attestations.sol`
- `packages/protocol/contracts/identity/Escrow.sol`
- `packages/protocol/contracts/stability/SortedOracles.sol`
- `packages/protocol/scripts/bash/release-on-devchain.sh`
- `packages/protocol/scripts/truffle/make-release.ts`

Within `PR #8360`

- `packages/protocol/contracts/common/Accounts.sol`

Within `PR #8475`

- `packages/protocol/contracts/governance/Validators.sol`
- `packages/protocol/contracts/stability/GrandaMento.sol`

Within `PR #8750`

- `packages/protocol/contracts/governance/ReleaseGold.sol`

Within `PR #8751`

- `packages/protocol/contracts/governance/ReleaseGold.sol`

Within `PR #8831`

- `packages/protocol/contracts/stability/ExchangeBRL.sol`
- `packages/protocol/contracts/stability/StableTokenBRL.sol`
- `packages/protocol/contracts/stability/proxies/ExchangeBRLProxy.sol`

Within `PR #8854`

- `packages/protocol/contracts/governance/LockedGold.sol`

If a file is not listed above, it should be considered out of scope for this audit. However, some notes were made on auxiliary files.

Overview of the changes

The pull requests are planned to be included in Release 6 of Celo's Core Contracts. A brief description is given of each pull request:

- PR #8300 changes the directory of the GrandaMento file for CI purposes.
- PR #8349 removes the `getVersionNumber` function from all libraries.
- PR #8360 introduces the `offchainStorageRoots` mapping on the Accounts contract that can be used to register new storage roots.
- PR #8475 implements per-proposal `vetoPeriodSeconds` and limits it to a hardcoded value.
- PR #8750 adds a `genericTransfer` function for rescuing ERC-20 tokens from ReleaseGold contracts.
- PR #8751 removes the requirement for funding when initializing ReleaseGold and adds an `isFunded` view function.
- PR #8831 adds and updates various files for the `StableTokenBRL` release. Note that PR #8831 was not merged at the time of this audit and we reviewed up to commit 63f75e3. The cLabs team confirmed that some parts of the implementation were still in development.
- PR #8854 adds a requirement that the reporter of a locked gold `slash` must also be a registered account.

Vulnerabilities

Below, we list all the vulnerabilities found in this audit.

Critical severity

None.

Medium severity

None.

Low severity

[L01] Missing error message in require statement

PR #8360 contains a require statement in `removeStorageRoot` that does not include an error message. Consider including specific and informative error messages in all require statements.

Update: *Fixed in PR9010 at commit `093ac05c3b66b8381ae85bfd4eef6b9eacb1c536`.*

[L02] `genericTransfer` can fail without error

In PR #8750, the `genericTransfer` function of the `ReleaseGold` contract calls the ERC20 member `transfer` function on the `erc20` token, but does not check the returned `bool` of the `transfer`.

This means that with certain ERC20 tokens, the `transfer` could fail and the user would receive no error messages with information on the failing conditions.

Consider updating the token `transfer` to instead use OpenZeppelin's `SafeERC20` `safeTransfer` function, which will provide proper logging in the event of a transfer failure.

Update: *Fixed in PR9025 at commit `6e5a9b26b3d4b34f7370f465ea8ee4fc28183cc3`.*

[L03] Validators does not increment PATCH version

In PR #8349, the `getVersionNumber` function is removed from `AddressLinkedList.sol`.

`Validators.sol` does not increment the PATCH version even though it is using `AddressLinkedList.sol`. Consider incrementing the PATCH version of the Validators contract to reflect the change.

[L04] Contract registry has no entry for `BRL`

In PR #8831, the `UsingRegistryV2` contract provides various getters returning addresses for contracts comprising the Celo protocol. For each featured protocol contract, a constant pointer is defined and used to inform the corresponding getter. While this mechanism isn't too complex, duplicating this process in code for a particular contract can lead to errors.

The problem is that the EUR token has corresponding getters for its registry and exchange, while the BRL token does not. So retrieval of corresponding BRL contract address would require code that may be error prone and not follow the standard defined by the `UsingRegistryV2` contract.

Consider defining within the `UsingRegistryV2` contract getters corresponding to the BRL token.

Update: *Fixed as of commit `0afc15a6dac109ada329e12c0d40fec7dcb9f40c`. cLab's statement for this issue:*

> *"we were indeed missing the getters, but the contract UsingRegistryV2 is not actually used anywhere and the contract to update should have been UsingRegistry."*

Notes & Additional Information

[N01] `addStorageRoot` can add duplicate `url`'s

In PR #8360, the `addStorageRoot` function of the `Accounts` contract adds `url`'s to an `offchainStorageRoots` array for the `msg.sender`'s `account`.

This function does not check whether the `url` already exists in the array so there can be duplicate entries added.

This does not pose a security risk. However, it can result in unnecessary gas expenditure. Furthermore the resulting redundant data returned by the `getOffchainStorageRoots` function can confuse offchain clients and add computational complexity to the processing of this data.

> *These duplicates can be checked for on the client side easily, both as a check before adding a duplicate or as a filter on the returned storage roots. To avoid complicating and inflating the cost of the storage root logic, we will not make a change to address this at this time.*

[N02] `ExchangeBRL.sol` lacks Solidity version pragma

In PR #8831, the `ExchangeBRL.sol` file does not define a Solidity version pragma.

In this case, it does not pose a security threat since the `ExchangeBRL` contract has trivial logic and its base contract does have its Solidity version pragma set. But it is considered good practice to always define the Solidity version pragma, since it can be a security concern when the contract contains non-trivial operations which can have different effects across Solidity versions.

Consider defining Solidity version pragmas for all Solidity source files in this project.

Update: *Fixed as of commit `8e0caecc0554d5a3f6fb25b6d024f29ff7562029`.*

[N03] Not using `SafeMath`

PR #8360 contains a `getOffchainStorageRoots` function which concatenates the url bytes of an account using the help of a local `totalLength` variable. The `totalLength` is the aggregated length of all the `offchainStorageRoots` elements for a specified account.

Consider using OpenZeppelin's `SafeMath` to perform the addition to `totalLength` to prevent overflow and maintain consistency with the method in `batchGetMetadataURL`.

Update: *Fixed in PR9026 as of commit `6af9bf0371550f723731080929f9d8946d78d524`.*

[N04] Incorrect token initialization data

In PR #8831, the `migrationsConfig.js` file is updated to include initialization data for the deployment of the `StableTokenBRL`.

This incorrect token initialization will not affect the inner logic and accounting of the token itself. However, this mislabelling can cause errors in indexing this token in exchanges, and will likely confuse users which can greatly affect its utility and adoption.

Consider updating the `migrationsConfig.js` to have appropriate `tokenName` and `tokenSymbol` set specific to the `StableTokenBRL`.

Update: *Fixed as of commit `92441e39e3b5cbb29bb8af6ed914b90a874d9afe`. cLabs comments on the issue:*

> *As a clarification, wanted to point out that file is only used for testing proposals and not for mainnet (not even testnets).*

Conclusions

No critical or high severity issues were found. Some changes were proposed to follow best practices and reduce potential attack surface.

Related Posts

Beefy Zap Audit

OpenBrush Contracts Library Security Review

Linea Bridge Audit

Security Audits

Security Audits

Security Audits