Celo Contracts Audit - Release 4

MAY 24, 2021 | IN SECURITY AUDITS | BY OPENZEPPELIN SECURITY

Introduction

As another phase of auditing, the cLabs team has asked us to review and audit the recent changes to the core contracts of the Celo protocol.

Scope

The audited commit for this phase has the tag `celo-core-contracts-v5.pre-audit` and associated commit hash `f64b4c5b5228ecbf41e3e7cfdbb8c0e9a983eea2` within the Celo Monorepo.

Specifically, we audited the difference between this commit and commit `04c63970135f4df165eb2e56d89e898bcf53d46d`, with associated tag `celo-core-contracts-v4.post-audit`.

Within the diff between commit `f64b4c5b5228ecbf41e3e7cfdbb8c0e9a983eea2` and `04c63970135f4df165eb2e56d89e898bcf53d46d`, the changes to the following files were considered in-scope for this audit:

```
packages/protocol/contracts/common/Accounts.sol
packages/protocol/contracts/common/Create2.sol
packages/protocol/contracts/common/GoldToken.sol
packages/protocol/contracts/common/InitializableProxy.sol
packages/protocol/contracts/common/InitializableV2.sol
packages/protocol/contracts/common/MetaTransactionWallet.sol
packages/protocol/contracts/common/Proxy.sol
packages/protocol/contracts/common/ProxyCloneFactory.sol
packages/protocol/contracts/governance/Election.sol
packages/protocol/contracts/governance/Governance.sol
packages/protocol/contracts/governance/Proposals.sol
packages/protocol/contracts/governance/ReleaseGold.sol
packages/protocol/contracts/identity/IdentityProxy.sol
packages/protocol/contracts/identity/IdentityProxyHub.sol
packages/protocol/contracts/stability/ExchangeEUR.sol
packages/protocol/contracts/stability/Reserve.sol
packages/protocol/contracts/stability/SortedOracles.sol
packages/protocol/contracts/stability/StableToken.sol
packages/protocol/contracts/stability/StableTokenEUR.sol
```

This corresponds to all modified solidity files that are intended for production. Note that there may be some references to files that were not audited in this report.

Overview of the changes

The reviewed version changes the functionality of signer authorizations to be more general, allowing an account to hold multiple signer roles, smart contracts to be signers, and the creation of new roles. It also updates many references to the now deprecated "cGLD" to "CELO". It now includes a new version of `Initializable`, which sets the `initialized` flag in the constructor, thereby preventing anyone from calling the initializer on the implementation contract. Additionally, the ability of admins to "freeze" transfers of the cGLD/CELO token has been removed.

Update

All of the following vulnerabilities have been addressed or acknowledged by the Celo team. **The updated version for this phase has the tag `celo-core-contracts-v5.post-audit` and associated commit hash `9e04953da04bb784d7d002d81129ec9cc8ab427b` within the Celo Monorepo.**

Vulnerabilities

Below, we list all vulnerabilities found in this audit.

Critical severity

None.

High severity

[H01] Incorrect use of `require` in `revokeVotes`

The `revokeVotes` function of the `Governance` contract is intended to revoke votes on proposals in the Referendum stage. However, it actually `requires` all `dequeued` proposals to be in that stage, which is not necessary during normal operations. Whenever any element of `dequeued` is not within the `Referendum` stage, the `revokeVotes` function will revert. This effectively makes the `revokeVotes` function unusable.

Instead of reverting, consider performing the following operations if the proposal is within the `Referendum` stage. Otherwise, that iteration of the loop should be treated as a no-op.

Update: *Fixed in commits a10bdf3 and 750341b.*

Medium severity

[M01] Potential out of gas error in `revokeVotes`

Within the `revokeVotes` function, the `for` loop will execute `dequeued.length` number of times.

Since the size of `dequeued` increases over time, eventually the contract may reach a state where `revokeVotes` will consume more than the available gas limit. This will make `revokeVotes` unusable.

Consider modifying `revokeVotes` to bound the number of iterations which the `for` loop can perform. This can be achieved, for example, by allowing users to specify indices within `dequeued` to clear.

Update: *The Celo team decided not to address this. In their words:*

> On the `Governance` contract, there is a maximum of `concurrentProposals` dequeued (added to the `dequeued` array) per `dequeueFrequency` time interval. The lifetime of a proposal is limited to `stageDurations.Approval + stageDurations.Referendum + stageDurations.Execution`, before it will be expired and removed from `dequeued` array. Therefore, the length of `dequeued` is already strongly limited.
>
> We could allow indices to be provided but the intent of this method is to guarantee that `isVoting(msg.sender) == false`. Without checking all of the `dequeued` indices, we cannot assign `voter.mostRecentReferendumProposal = 0`, and the method won't accomplish its goal if misused. There is virtually no utility in a `revokeVotes` that only revokes some of the dequeued proposals.

Low severity

[L01] Discarded return value

The `makeCall` function of the `IdentityProxy` contract allows an authorized user to make an arbitrary call on behalf of the contract. However, the return value of the call is discarded, which unnecessarily limits the possible use cases. Consider bubbling the return value back to the caller so they can react to it.

Update: *Fixed in commits d417345 and cba251c.*

[L02] Extraneous Events

Many functions in the `Accounts` contract complete successfully and emit extraneous events when there is no operation to be performed. For example, it's possible to authorize a signer that already exists or remove a signer that doesn't exist. In such cases, the emitted event will not correspond to any executed action, which may intefere with offline processing. Consider validating that the contract state has changed before emitting events.

Update: *The Celo team decided not to address this. In their words:*

1. Extraneous events should be considered no-ops from the perspective of indexing the chain
2. The current behaviour is consistent with the other setters (e.g. `setName`, `setMetadataURL`, etc) in our `Accounts.sol` contract
3. Calls to authorise signers remain idempotent which is desirable and correct

[L03] Misleading comments

Within the codebase, some misleading comments or docstrings were identified. For example:

- Line 271 of `Accounts.sol` indicates that the signature is over `msg.sender`, when in actuality it should be a signature on the EIP 712 encoding of `AuthorizeSigner` struct with the `msg.sender` account and the specified role.
- Line 900 of `Accounts.sol` should better reflect the logic of the function. Consider changing the comment to "Returns `false` if authorized signer for another account. Returns `true` otherwise".

Consider updating the comments and docstrings to more accurately describe the purpose and effect of the codebase.

Update: *Fixed in commit 3b27e3a.*

[L04] Missing docstrings

The function `getRoleAuthorizationSigner` is missing natspec comments. This hinders reviewers' understanding of the code's intention, which is fundamental to correctly assess not only security, but also correctness. Additionally, docstrings improve readability and ease maintenance. They should explicitly explain the purpose or intention of the functions, the scenarios under which they can fail, the roles allowed to call them, the values returned and the events emitted.

Consider thoroughly documenting all functions (and their parameters) that are part of the contracts' public API. Functions implementing sensitive functionality, even if not public, should be clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: *Fixed in commits 234c647 and 9e04953.*

[L05] Missing Event

The `removeLegacySigner` function of the `Accounts` contract does not emit an event. Consider emitting events after all sensitive changes to facilitate tracking and notify off-chain clients about the contract's activity.

Update: *Fixed in commit 8317448.*

[L06] Proxy ownership fragility

After deploying an `InitializableProxy`, the `ProxyCloneFactory` transfers ownership of the proxy to itself. This potentially allows the proxy to invoke its own administration functions. Since the proxy's functionality is determined by its logic contract, whether this is possible or desirable depends on the particular use case and implementation, but that may not be obvious when constructing the logic contract. In particular, implementations that permit arbitrary function calls may accidentally introduce this behavior. Consider documenting this possibility in the `deploy` function comments so that proxy users are aware of the risk.

Update: *Fixed in commit 222644b.*

[L07] Unnecessary Access Control

The `setEip712DomainSeparator` function of the `Accounts` contract can only be called by the contract owner. However, it

is simply used to initialize a contract variable, and its functionality does not depend on the caller. For simplicity, consider removing the `onlyOwner` modifier.

Additionally, consider calling this function within the contract initializer. This will not affect any contracts that are upgraded to this version, but it would ensure that fresh deployments are initialized correctly.

Update: *Fixed in commit 2049082.*

[L08] Unnecessary operations in `revokeVotes`

The `revokeVotes` function is used to set a voters `referendumVotes` for a proposal to zero. However, it is executed against all active proposals, including ones where the voter hasn't voted. In such cases, all the operations associated with removing a voter's `referendumVotes` will still occur.

Consider only executing the main logic of the function when the voter has submitted non-zero `referendumVotes` for the relevant proposal, to avoid wasting gas.

Update: *Fixed in commit 056a5b4.*

Notes & Additional Information

[N01] TODOs in code

There are "TODO" comments in the code base that should be tracked in the project's issues backlog. See for example:

- `ProxyCloneFactory` line 24
- `ProxyCloneFactory` line 56
- `Reserve` line 214
- `Election` line 402
- `StableToken` line 458

During development, having well described "TODO" comments will make the process of tracking and solving them easier. Without that information, these comments might tend to rot and important information for the security of the system might be forgotten by the time it is released to production.

These TODO comments should at least have a brief description of the task pending to do, and a link to the corresponding issue in the project repository. Consider updating the TODO comments to add this information. For completeness and traceability, a signature and a timestamp can be added.

Update: *The Celo team showed us their project backlog, which contains links to the relevant issues on GitHub.*

[N02] Typographical errors

Several typographical errors were found in the codebase. Some examples are:

- Line 669 of `Governance.sol` should say "proposals" instead of "proposal".
- Line 670 of `Governance.sol` should say "were" instead of "was".
- Line 265 of `Accounts.sol` should say "to act as" instead of "to as".
- Line 918 of `Accounts.sol` references a non-existent `current` variable.
- Lines 927-928 of `Accounts.sol` contain empty comments which can be removed.

- Line 19 of `IdentityProxy.sol` should say "to call" instead of "the call".
- Line 24 and Line 29 of `ProxyCloneFactory.sol` should say "OpenZeppelin" instead of "open-zeppelin".

Consider correcting them.

Update: *Fixed in commit 414cfed.*

[N03] Redundant Inheritance

The `GoldToken` contract inherits from `Ownable`, but it does not include any functions with access control. Moreover, `Ownable` is already included in the inheritance chain through the `UsingRegistry` contract. Consider removing the unused and redundant inheritance of `Ownable`.

Update: *Fixed in commit 8b50238.*

[N04] Simplify identity heuristic calculation

The `passesIdentityHeuristic` function of the `IdentityProxyHub` contract evaluates multiple independent conditions, including a compound condition, and then evaluates whether or not they are all satisfied. For simplicity, consider returning immediately when any of the conditions fail. This would make the function easier to reason about, and would also increase efficiency.

Update: *Fixed in commit 3f405e0. This does not change the logic but it adds comments explaining the reasoning.*

Conclusions

No critical and 1 high severity issue was found. Some changes were proposed to follow best practices and reduce potential attack surface.