May 4, 2022

Smart Contract Assessment – Staked Celo - Phase 2

Prepared by FTI Consulting

FTI CONSULTING

Overview

This report has been prepared for cLabs in review of Staked Celo smart contracts to identify issues and vulnerabilities in the source code as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The assessment process pays particular attention to the following considerations:

- Testing smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring that the contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase.

The security assessment resulted in findings that ranged from high to low. FTI recommends addressing these findings to ensure security standards as well as enhancing general coding practices for better structure of source code as described in Detailed Findings.

FTI CONSULTING

Project Summary

Project Name	Staked Celo Contracts
Description	The assessment is comprised of code review of Staked Celo smart contracts repository. The original assessment included two unmerged pull requests. A reassessment was conducted after the pull requests were merged. The report is split into two parts containing: 1. Issues identified during the reassessment delivered May 4, 2022 2. Issues identified in the original assessment delivered April 21, 2022, with updated status from the reassessment
Platform	EVM Compatible Chains
Language	Solidity
Scope and Codebase	Original Assessment Delivered April 21, 2022: https://github.com/celo-org/staked-celo/tree/master/contracts https://github.com/celo-org/staked-celo/tree/soloseng/rebasing-wrapper-token/contracts https://github.com/celo-org/staked-celo/tree/m-chrzan/voting-limit/contracts Commit: 9a74c5e5784d93ad3128aeb858deaeab510ed130 Reassessment Delivered May 4, 2022: https://github.com/celo-org/staked-celo/tree/master/contracts Commit: 3adf699456e8e03bcea61d63e6595abe30dad0ba

FTI CONSULTING

Assessment Summary

Original Assessment Delivery Date	April 21, 2022
Reassessment Delivery Date	May 4, 2022
Assessment Methodology	Static Analysis, Manual Review

Reassessment delivered May 4, 2022:

Vulnerability Summary

Total Issues	**1**
High	0
Medium	0
Low	1

Overview of All Contracts Assessed

Number	Contract	Name	General Assessment
1	SC-01	StakedCelo.sol	No Changes Required

2	MGR-01	Manager.sol	No Changes Required
3	MGD-01	Managed.sol	No Changes Required
4	ACT-01	Account.sol	Coding Style
5	RC-01	RebasedCelo.sol	No Changes Required

Findings – Vulnerability Summary

ID	Title	Category	Severity	Status
ACT-01	Account.sol	Coding Style	Low	Requires Attention

Detailed Findings

Low

1. Account.sol

ID	Title	Category	Severity	Status
ACT-01	Account.sol	Coding Style	Low	Requires Attention

Local variable shadowing

Line **#195**
initialize(address _registry, address _manager, address _owner)

Issue

_owner shadows *_owner()* (OwnableUpgradeable **#22**)

Recommendation

Consider renaming the variable as *owner_* inside *initialize()* that shadows the state variables.

Original Assessment delivered April 21, 2022 with Updated Status:

Vulnerability Summary

Branch: staked-celo-master

Total Issues	**4**
Medium	1
Low	3

Branch: staked-celo-m-chrzan-voting-limit

Total Issues	**4**
Medium	1
Low	3

Branch: staked-celo-soloseng-rebasing-wrapper-token

Total Issues	**7**

High	1
Medium	1
Low	5

Overview of All Contracts Assessed

Number	Contract	Name	General Assessment
1	SC-01	StakedCelo.sol	Coding Style
2	MGR-01	Manager.sol	Coding Style, Gas Optimization, Logical Issue
3	MGD-01	Managed.sol	Gas Optimization, Coding Style
4	ACT-01	Account.sol	Coding Style, Gas Optimization
5	RC-01	[branch: staked-celo-soloseng-rebasing-wrapper-token] RebasedCelo.sol	Coding Style, Gas Optimization

Findings – Vulnerability Summary

Branch: staked-celo-master

ID	Title	Category	Severity	Status
MGR-01	Manager.sol	Gas Optimization, Coding style, Logical Issue	Medium	Resolved
SC-01	StakedCelo.sol	Coding style	Low	Resolved
MGD-01	Managed.sol	Coding Style, Gas Optimization	Low	Resolved
ACT-01	Account.sol	Coding Style, Gas Optimization	Low	Requires Attention

Branch: staked-celo-m-chrzan-voting-limit

ID	Title	Category	Severity	Status
MGR-01	Manager.sol	Gas Optimization, Coding style, Logical Issue	Medium	Resolved
SC-01	StakedCelo.sol	Coding style	Low	Resolved
MGD-01	Managed.sol	Coding Style, Gas Optimization	Low	Resolved
ACT-01	Account.sol	Coding Style, Gas Optimization	Low	Requires Attention

Branch: staked-celo-soloseng-rebasing-wrapper-token

ID	Title	Category	Severity	Status
RC-01	RebasedCelo.sol	Coding Style, Gas Optimization	High	Resolved
MGR-01	Manager.sol	Gas Optimization, Coding style, Logical Issue	Medium	Resolved
SC-01	StakedCelo.sol	Coding style	Low	Resolved

MGD-01	Managed.sol	Coding Style, Gas Optimization	Low	Resolved
ACT-01	Account.sol	Coding Style, Gas Optimization	Low	Requires Attention

Detailed Findings

Branch: staked-celo-master

Medium

1. Manager-01

ID	Title	Category	Severity	Status
MGR-01	Manager.sol	Coding Style	Medium	Resolved

Unused return

Lines **#186, #196, #223, #421**

Issue

The value returned by an external call is not stored in local or state variable in the below functions:

activateGroup(address) #180-198
deprecatedGroups.remove(group) #186
activeGroups.add(group) #196

deprecateGroup(address) #215-227
deprecatedGroups.add(group) #223

getDeprecatedGroupsWithdrawalDistribution(uint256) #398-436
deprecatedGroups.remove(deprecatedGroupsWithdrawn[i]) #421

add() and remove() returns a bool when called. This value is unused in the code.

Recommendation

Consider making use of the return values, here the returned value is boolean and it indicates if the add() or remove() call was successful or not.

Low

1. StakedCelo.sol

ID	Title	Category	Severity	Status
SC-01	StakedCelo.sol	Coding Style	Low	Resolved

Local variable shadowing

Line **#26**
initialize(address manager, address owner)

Issue

manger shadows *Managed.manager* (Managed.sol **#12**)
owner shadows *OwnableUpgradeable.owner()* (OwnableUpgradeable.sol **#30**)

Recommendation

Consider renaming variables inside *initialize()* that shadows the state variables.

2. Account.sol

ID	Title	Category	Severity	Status
ACT-01	Account.sol	Coding Style, Gas Optimizations	Low	Requires Attention

Local variable shadowing

Line **#198, #199, #200**
initialize(address registry, address manager, address owner)

Issue

registry shadows *registry* (UsingRegistryUpgradeable.sol **#47**)
manger shadows *manager* (Managed.sol **#12**)
owner shadows *owner()* (OwnableUpgradeable **#40-42**)

Recommendation

Consider renaming variables inside *initialize()* that shadows the state variables.

Reentrancy vulnerabilities

Line: **#302-365**, **#378-412**

withdraw() **#302-365**
activateAndVote() **#378-412**

Issue

activateAndVote() :
External call on **#388** and state variables written after the external call on **#402.**
withdraw() :

External calls on **#344, revokeVotes() #347-355,**
lockedGold.unlock() **#358**
State variables written after the external calls on **#360-362:**
pendingWithdrawals[beneficiary].push()

Recommendation

Consider saving the pending withdrawal amount before the external call.
Consider resetting the unlocked CELO amount for the group before the external call.

State variables that could be declared constant

Line: #**48**

Issue

The below variable can be declared as constant to save gas.
address public stakedCelo **#48**

Recommendation

Declare *address stakedCelo* as *constant*.

3. Manager.sol

ID	Title	Category	Severity	Status
MGR-01	Manager.sol	Coding style	Low	Resolved

Local variable shadowing

Line **#120**
initialize(address registry, address owner)

Issue

registry shadows *registry* (UsingRegistryUpgradeable.sol **#47**)
owner shadows *owner()* (OwnableUpgradeable **#40-42**)

Recommendation

Consider renaming variables inside *initialize()* that shadows the state variables.

4. Managed.sol

ID	Title	Category	Severity	Status
MGD-01	Managed.sol	Coding Style, Gas Optimization	Low	Resolved

Missing zero address validation

Line: **#49**

Issue

Lacking a zero-address check on *_manager*.

Recommendation

Consider verifying that *_manager* is a non-zero address.

Declare External

Line: **#49-52**
setManager(address) **#49-52**

Issue

Function *setManager()* is declared *public*, but is never called by the contract.

Recommendation

setManager() can be declared *external* to save gas.

Branch: staked-celo-m-chrzan-voting-limit

Medium

1. Manager.sol

ID	Title	Category	Severity	Status
MGR-01	Manager.sol	Logical Issue	Medium	Resolved

Unused return

Lines: **#186, #196, #223, #421**

Issue

The value returned by an external call is not stored in local or state variable

in the below functions:

activateGroup(address) #187-205
deprecatedGroups.remove(group) #193
activeGroups.add(group) #203

deprecateGroup(address) #222-234
deprecatedGroups.add(group) #230

getDeprecatedGroupsWithdrawalDistribution(uint256) #421-459
deprecatedGroups.remove(deprecatedGroupsWithdrawn[i]) #444

add() and remove() returns a bool when called. This value is unused in the code.

Recommendation

Consider making use of the return values, here the returned value is boolean and it indicates if the add() or remove() call was successful or not.

Low

1. StakedCelo.sol

ID	Title	Category	Severity	Status
SC-01	StakedCelo.sol	Coding Style	Low	Resolved

Unused return

Lines: **#26**
initialize(address manager, address owner)

Issue

manger shadows *Managed.manager* (Managed.sol **#12**)
owner shadows *OwnableUpgradeable.owner()* (OwnableUpgradeable.sol **#30**)

Recommendation

Consider renaming variables inside *initialize()* that shadows the state variables.

2. Account.sol

ID	Title	Category	Severity	Status
ACT-01	Account.sol	Coding Style, Gas Optimizations	Low	Requires Attention

Local variable shadowing

Lines **#198, #199, #200**
initialize(address registry, address manager, address owner)

Issue

registry shadows *registry* (UsingRegistryUpgradeable.sol **#47**)
manger shadows *manager* (Managed.sol **#12**)
owner shadows *owner()* (OwnableUpgradeable **#40-42**)

Recommendation

Consider renaming variables inside *initialize()* that shadows the state variables.

Reentrancy in Events

Line: **#302-365**, **#378-412**

withdraw() **#302-365**
activateAndVote() **#378-412**

Issue

activateAndVote() :
External call on **#388** and state variables written after the external call on **#402.**
withdraw() :
External calls on **#344, #347-355 revokeVotes(), #358 lockedGold.unlock().** State variables written after the external calls on **#360-362:**
pendingWithdrawals[beneficiary].push()

Recommendation

Consider saving the pending withdrawal amount before the external call.
Consider resetting the unlocked CELO amount for the group before the external call.

State variables that could be declared constant

Line: #**48**

Issue

The below variable can be declared as constant to save gas. **#48**
address public stakedCelo;

Recommendation

Declare *address stakedCelo* as *constant*.

3. Manager.sol

ID	Title	Category	Severity	Status
MGR-01	Manager.sol	Coding Style, Gas Optimizations	Low	Resolved

Local variable shadowing

Line **#127**
initialize(address registry, address owner)

Issue

registry shadows *registry* (UsingRegistryUpgradeable.sol **#47**)
owner shadows *owner()* (OwnableUpgradeable **#40-42**)

Recommendation

Consider renaming variables inside *initialize()* that shadows the state variables.

4. Managed.sol

ID	Title	Category	Severity	Status
MGD-01	Managed.sol	Coding Style, Gas Optimizations	Low	Resolved

Missing zero address validation

Line: **#49**

Issue

Lacking a zero-address check on *_manager.*

Recommendation

Consider verifying that *_manager* is a non-zero address.

Declare External

Line: **#49-52**

setManager(address) **#49-52**

Issue

Function *setManager()* is declared *public*, but is never called by the contract.

Recommendation

setManager() can be declared *external* to save gas.

branch: staked-celo-soloseng-rebasing-wrapper-token

High

1. RebasedCelo.sol

ID	Title	Category	Severity	Status

RC-01	Rebased.sol	Coding Style	High	Resolved

Unchecked transfer

Line: **#119, #142**

Issue

The return values of external *transferFrom* calls (**#119, #142**) are not checked.

Recommendation

Consider using SafeERC20 or make sure to check the boolean value returned by *transferFrom*() calls because they indicate whether the call was successful or not.

Medium

1. Manager.sol

ID	Title	Category	Severity	Status
MGR-01	Manager.sol	Logical Issue	Medium	Resolved

Unused return

Lines **#186, #196, #223, #421**

Issue

The value returned by an external call is not stored in local or state variable in the below functions:
activateGroup(address) #180-198

deprecatedGroups.remove(group) #186
activeGroups.add(group) #196

deprecateGroup(address) #215-227
deprecatedGroups.add(group) #223

getDeprecatedGroupsWithdrawalDistribution(uint256) #398-436
deprecatedGroups.remove(deprecatedGroupsWithdrawn[i]) #421

add() and remove() returns a bool when called. This value is unused in the code.

Recommendation

Consider making use of the return values, here the returned value is boolean and it indicates if the add() or remove() call was successful or not.

Low

1. StakedCelo.sol

ID	Title	Category	Severity	Status
SC-01	StakedCelo.sol	Coding Style	Low	Resolved

Local variable shadowing

Line **#26**
initialize(address manager, address owner)

Issue

manger shadows *Managed.manager* (Managed.sol **#12**)
owner shadows *OwnableUpgradeable.owner()* (OwnableUpgradeable.sol **#30**)

Recommendation

Consider renaming variables inside *initialize()* that shadows the state variables.

2. Account.sol

ID	Title	Category	Severity	Status
ACT-01	Account.sol	Coding Style	Low	Resolved

Local variable shadowing

Line **#198, #199, #200**
initialize(address registry, address manager, address owner)

Issue

registry shadows *registry* (UsingRegistryUpgradeable.sol **#47**)
manger shadows *manager* (Managed.sol **#12**)
owner shadows *owner()* (OwnableUpgradeable **#40-42**)

Recommendation

Consider renaming variables inside *initialize()* that shadows the state variables.

Reentrancy vulnerabilities

Issue

Line : **378-412**
Account.activateAndVote(address,address,address) **#378-412**
External calls:

election.activate(group) **#388**
State variables written after the call(s):
scheduledVotes[group].toVote = 0 **#402**

Line: 302-365
Account.withdraw(address,address,address,address,address,address,uint256)

External calls:
success = getGoldToken().transfer(beneficiary,immediateWithdrawalAmount) **#334**
revokeVotes(group,revokeAmount,lesserAfterPendingRevoke,greaterAfterPendingRevoke,
lesserAfterActiveRevoke,greaterAfterActiveRevoke,index) **#347-355**

!election.revokePending(group,toRevokeFromPending,lesserAfterPendingRevoke,greaterA
fterPendingRevoke,index) **#615-621**

!election.revokeActive(group,toRevokeFromActive,lesserAfterActiveRevoke,greaterAfterA
ctiveRevoke,index) #639-645

lockedGold.unlock(revokeAmount) (contracts/Account.sol#358)

State variables written after the call(s):
pendingWithdrawals[beneficiary].push(PendingWithdrawal(revokeAmount,block.timesta
mp + lockedGold.unlockingPeriod())) (contracts/Account.sol#360-362)

Recommendation

Consider saving the pending withdrawal amount before the external call.
Consider resetting the unlocked CELO amount for the group before the external call.

State variables that could be declared constant

Line: #**48**

Issue

The below variable can be declared as constant to save gas. **#48**
address public stakedCelo;

Recommendation

Declare *address stakedCelo* as *constant*.

3. Manager.sol

ID	Title	Category	Severity	Status
MGR-01	Manager.sol	Coding Style	Low	Resolved

Local variable shadowing

Line **#120**
initialize(address registry, address owner)

Issue

registry shadows *registry* (UsingRegistryUpgradeable.sol **#47**)
owner shadows *owner()* (OwnableUpgradeable **#40-42**)

Recommendation

Consider renaming variables inside *initialize()* that shadows the state variables.

4. Managed.sol

ID	Title	Category	Severity	Status
MGD-01	Managed.sol	Coding Style, Gas Optimization	Low	Resolved

Missing zero address validation

Line: **#49**

Issue

Lacking a zero-address check on *_manager.*

Recommendation

Consider verifying that *_manager* is a non-zero address.

Declare External

Line: **#49-52**
setManager(address) **#49-52**

Issue

Function *setManager()* is declared *public*, but is never called by the contract.

Recommendation

setManager() can be declared *external* to save gas.

5. RebasedCelo.sol

ID	Title	Category	Severity	Status
RC-01	Rebased.sol	Coding Style, Gas Optimization	Low	Resolved

Local variable shadowing

Line: **#95**

initialize(address _stakedCelo, address _account, address _manager,address _owner)

Issue

_owner shadows *OwnableUpgradeable._owner()* (OwnableUpgradeable.sol **#22**)

Recommendation

Consider renaming variables inside *initialize()* that shadows the state variables.

Reentrancy vulnerabilities

Line: **#108-121, #127-144**

Issue

RebasedCelo.deposit(uint256) **#108-121**
External call:
stakedCelo.transferFrom(msg.sender,address(this),stCeloAmount) **#119**
Event emitted after the call:
StCeloDeposited(msg.sender,stCeloAmount) **#120**
RebasedCelo.withdraw(uint256) **#127-144**
External call:
stakedCelo.transfer(msg.sender,stCeloAmount) **#142**
Event emitted after the call:
StCeloWithdrawn(msg.sender,stCeloAmount) **#143**

Event is emitted after the call. If reentrency occurs, events will be shown in an incorrect order, which might lead to issues for third parties.

Recommendation

Consider emitting the event *StCeloDeposited* before the external calls.

Declare External

Line: **#241-243**

stakedCeloBalance(address) **#241-243**

Issue

Function *stakedCeloBalance()* is declared *public*, but is never called by the contract.

Recommendation

stakedCeloBalance() can be declared *external* to save gas.

Appendix - Category Descriptions

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e., incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase is more legible and, as a result, easily maintainable.

Mathematical Operations

Mathematical operation findings related to mishandling of math formulas, such as overflows, incorrect operations etc.

Disclaimer

This report is based on the documents and materials provided to FTI as of the time of this report. The report is for informational purposes only and should not be used as the basis for any determinations or decisions by the recipient. This report is not an endorsement or disapproval of any particular project or team nor is it an indication of the economic value of any product or asset created by any team or project. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice. FTI shall not bear any responsibility for the consequences of the relevant decisions adopted by the recipient.

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed and should not be considered a perfect representation of the risks threatening the analyzed system. Source code reviews are a “point-in-time” analysis and it is possible that the code and overall smart contract functionality could have changed since the review in this report was conducted.

In consideration of FTI allowing the recipient access to the report and, the recipient agrees that it does not acquire any rights as a result of such access that it would not otherwise have had and acknowledges that FTI does not assume any duties or obligations to the recipient in connection with such access.

The recipient agrees to release FTI and its personnel from any claim by the recipient that arises as a result of FTI permitting the recipient access to the report. Without our written consent, the report shall not be disclosed or provided to any other party or used for any other purposes.