Cudo Ventures

Security Assessment

October 21st, 2020

For :

Cudo Ventures

By :

Alex Papageorgiou @ CertiK

alex.papageorgiou@certik.org

Georgios Delkos @ CertiK

georgios.delkos@certik.io

Overview

Project Summary

Project Name	Cudo Ventures
Description	A typical ERC20 implementation with a strict whitelist feature and Ether withdrawing mechanism.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. a9c2c34870f46398d863d8ac60472bda2c37dc1b

Audit Summary

Delivery Date	**October 19th, 2020**
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	October 18th, 2020 - October 19th, 2020

Vulnerability Summary

Total Issues	**5**
Total Critical	0
Total Major	0
Total Medium	0
Total Minor	1
Total Informational	4

Executive Summary

This section will represent the summary of the whole audit process once it has concluded.

Files In Scope

ID	Contract	Location
CTN	CudosToken.sol	contracts/CudosToken.sol

Findings

ID	Title	Type	Severity	Resolved
CTN-01	Numerical Representation Refinement	Coding Style	Informational	✓
CTN-02	Mutability Specifier Missing	Gas Optimization	Informational	✓
CTN-03	Error Message Consistency	Coding Style	Informational	✓
CTN-04	`require` to `modifier`	Coding Style	Informational	✓
CTN-05	Inexistent Input Sanitization	Volatile Code	Minor	✓

CTN-01: Numerical Representation Refinement

Type	Severity	Location
Coding Style	Informational	CudosToken.sol L17

Description:

The linked number literal contains many zero digits and no separator.

Recommendation:

In Solidity, numbers can be split with the `_` character which is ignored by the compiler meaning that the linked `TEN_BILLION` variable can be better represented as `10_000_000_000`.

Alleviation:

The number representation format was changed according to our recommendation.

CTN-02: Mutability Specifier Missing

Type	Severity	Location
Gas Optimization	Informational	CudosToken.sol L26

Description:

The linked variable is assigned to once during the `constructor`'s execution and on each execution of the `updateAccessControls` function. We advise that the necessity of the `updateAccessControls` function is evaluated as, should it be omitted, the gas cost involving the `require` checks can be greatly optimized by applying the recommendation clause of this exhibit.

Recommendation:

If the `updateAccessControls` function is revised, we advise that the `immutable` mutability specifier is set at the variable's contract-level declaration to greatly optimize the gas cost of utilizing the variables.

Alleviation:

The `updateAccessControls` function remained as is and as such, this exhibit can be considered null.

CTN-03: Error Message Consistency

Type	Severity	Location
Coding Style	Informational	CudosToken.sol L37, L50

Description:

The error messages of the `CudosToken` smart contract utilize a convention whereby the function, if any, is also specified in the error message. The linked error messages do not conform to this convention.

Recommendation:

We advise that the same convention is applied across the codebase to aid in the debugging process.

Alleviation:

The error messages were properly updated to conform to the format mentioned in the description.

CTN-04: `require` to `modifier`

Type	Severity	Location
Coding Style	Informational	CudosToken.sol L37, L50, L60, L70

Description:

The linked `require` statements are repeated in the codebase with a different error message.

Recommendation:

These statements could be grouped to `modifier`s that accept the error message as input, easing the maintainability of the codebase.

Alleviation:

This exhibit was not applied to the codebase, however it is an informational exhibit and as such can be ignored.

CTN-05: Inexistent Input Sanitization

Type	Severity	Location
Volatile Code	Minor	CudosToken.sol L80-L83

Description:

The linked function can lock all transfers of the token by assigning the `accessControls` variable to zero as it is not sanitized in the function.

Recommendation:

We advise that some form of sanitization takes place, i.e. the new address also evaluates the result of the `hasAdminRole` function.

Alleviation:

The input address of the `updateAccessControls` function is properly sanitized to not be equal to the zero address.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that 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.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that 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.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an in-storage one.

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 and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.