Audit Report

1. DELIRIUM MASTERCHEF
2. DELIRIUM TOKEN
3. DELIRIUM TIMELOCK

Table of Contents

Introduction

This interim report has been prepared for the Delirium Token, Delirium Masterchef and Delirium Timelock smart contracts. The purpose of this report is to provide insights with the aim of optimizing current smart contracts. Due to the overlap in code, it was decided to reduce the interim report to 1 summary document. The procedure for arriving at the following conclusions is made up of the following:

- Testing the code against known and rare attack patterns
- Assessing the layout of the various code components to test best practice
- Scanning and stress testing of the contract functions, including low-level calls and edge cases. Cross
- Thorough line-by-line inspection by certified Solidity Developer.
- Masterchef isn't under timelock yet, We have been informed that this will happen after the presale ends and will update this in the near future.

The investigation resulted in a number of minimal findings, with mainly informative considerations. The "Findings" section contains an overview of the findings and associated recommendations or additional information. In a separate file are the explanations of the function calls, inheritance and call-graph in .DOT format.

Overview

Project summary

Contract Names	**Delirium Token | Delirium Masterchef | Delirium TimeLock**
Network	Polygon
Language	Solidity
Codebase	https://polygonscan.com/address/0xd3976E92a48821DD1122Ae5e8265b14595aF34d2

Audit summary

Delivery date	10-09-2021
Audit Methodology	Static Analysis, Manual Analysis

Risk summary

Risk Level	Total	Reported	Disproved	Solved	Recognized	Objection
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	0	0	0	0	0	0
Minor	4	0	0	0	0	0
Informative	14	0	0	0	0	0
Discussion	1	0	0	0	0	0

Findings

Public functions that can be declared External.

Category	Risk Level	Amount	Status
Gas Optimization	Informative	15	Reported

Low-Level Calls.

Category	Risk Level	Amount	Status
Sensitive to errors	Minor	4	Reported

Static variable that can be made constant.

Category	Risk Level	Amount	Status
Gas Optimization	Informative	1	Reported

Equation with constant Boolean

Category	Risk Level	Amount	Status
Gas Optimization	Informative	1	Reported

Unused return value.

	Risk Level	Amount	Status
Lost Computation	Discussion	1	Reported

Findings summary

Public functions that can be declared External (Master Chef)

renounceOwnership() Could be declared as external:

- Ownable.renounceOwnership() (masterchef.sol#72-74)

transferOwnership(address) Could be declared as external:

- Ownable.transferOwnership(address) (masterchef.sol#80-83)

name() Could be declared as external:

- ERC20.name() (masterchef.sol#604-606)

symbol() Could be declared as external:

- ERC20.symbol() (masterchef.sol#612-614)

decimals() Could be declared as external:

- ERC20.decimals() (masterchef.sol#629-631)

totalSupply() Could be declared as external:

- ERC20.totalSupply() (masterchef.sol#636-638)

balanceOf(address) Could be declared as external:

- ERC20.balanceOf(address) (masterchef.sol#643-645)

transfer(address,uint256) Could be declared as external:

- ERC20.transfer(address,uint256) (masterchef.sol#655-658)

allowance(address,address) Could be declared as external:

- ERC20.allowance(address,address) (masterchef.sol#663-665)

approve(address,uint256) Could be declared as external:

- ERC20.approve(address,uint256) (masterchef.sol#674-677)

transferFrom(address,address,uint256) Could be declared as external:

- ERC20.transferFrom(address,address,uint256) (masterchef.sol#692-706)

increaseAllowance(address,uint256) Could be declared as external:

- ERC20.increaseAllowance(address,uint256) (masterchef.sol#720-723)

decreaseAllowance(address,uint256) Could be declared as external:

- ERC20.decreaseAllowance(address,uint256) (masterchef.sol#739-747)

mint(address,uint256) Could be declared as external:

- DeliriumToken.mint(address,uint256) (masterchef.sol#908-910)

setEmissionRate(uint256) Could be declared as external:

- MasterChefV2.setEmissionRate(uint256) (masterchef.sol#1203-1210)

Public functions that can be declared External (Token)

renounceOwnership() can be written as external:

- Ownable.renounceOwnership() (tokensingle.sol#72-74)

transferOwnership(address) can be written as external:

- Ownable.transferOwnership(address) (tokensingle.sol#80-83)

name() can be written as external:

- ERC20.name() (tokensingle.sol#247-249)

symbol() can be written as external:

- ERC20.symbol() (tokensingle.sol#255-257)

decimals() can be written as external:

- ERC20.decimals() (tokensingle.sol#272-274)

totalSupply() can be written as external:

- ERC20.totalSupply() (tokensingle.sol#279-281)

balanceOf(address) can be written as external:

- ERC20.balanceOf(address) (tokensingle.sol#286-288)

transfer(address,uint256) can be written as external:

- ERC20.transfer(address,uint256) (tokensingle.sol#298-301)

allowance(address,address) can be written as external:

- ERC20.allowance(address,address) (tokensingle.sol#306-308)

approve(address,uint256) can be written as external:

- ERC20.approve(address,uint256) (tokensingle.sol#317-320)

transferFrom(address,address,uint256) can be written as external:

- ERC20.transferFrom(address,address,uint256) (tokensingle.sol#335-349)

increaseAllowance(address,uint256) can be written as external:

- ERC20.increaseAllowance(address,uint256) (tokensingle.sol#363-366)

decreaseAllowance(address,uint256) can be written as external:

- ERC20.decreaseAllowance(address,uint256) (tokensingle.sol#382-390)

mint(address,uint256) can be written as external:

Low-level calls

Low level call in Address.sendValue(address,uint256) (chef.sol#142-147):

- (success) = recipient.call{value: amount}() (chef.sol#145)

Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (chef.sol#210-221):

- (success,returndata) = target.call{value: value}(data) (chef.sol#219)

Low level call in Address.functionStaticCall(address,bytes,string) (chef.sol#239-248):

- (success,returndata) = target.staticcall(data) (chef.sol#246)

Low level call in Address.functionDelegateCall(address,bytes,string) (chef.sol#266-275):

- (success,returndata) = target.delegatecall(data) (chef.sol#273)

Reduce gas by defining state variable as constant

MasterChefV2.deliriumMaximumSupply (chef.sol#950) can be written as a constant.

Comparison with Boolean

MasterChefV2.nonDuplicated(IERC20) (chef.sol#999-1002) makes a comparison with a Boolean value:

-require(bool,string)(poolExistence[_lpToken] == false,nonDuplicated: duplicated) (chef.sol#1000)

Ignored Return Value

MasterChefV2.add(uint256,IERC20,uint16,bool) (chef.sol#1005-1027) ignores the return value of _lpToken.balanceOf(address(this)) (chef.sol#1007)