AUDIT REPORT

DATE APRIL 7TH

FOR

ORAKURU

Solid Group Auditing Service
Telegram: @solid_1
Twitter: https://twitter.com/solid_group_1

Disclaimer

SolidGroup reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team.

SolidGroup Audits do not provide any warranty or guarantee **regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors. SolidGroup Audits should not be used in any way to make decisions around investment or involvement with any particular project**. **These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.**

SolidGroup Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. **SolidGroup’s position is that each company and individual are responsible for their own due diligence and continuous security.** SolidGroup in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Overview

Network: Binance Smart Chain
Website: https://orakuru.io
Twitter Group: https://t.me/Orakuru_ork
Telegram Group: https://t.me/Orakuru

Description

Orakuru is Binance Smart Chain’s community owned and operated Oracle.

Files In Scope

Contract Name	MD5
Orakuru.sol	b3a2e71a71f044275ef27bcca4796a87
TokenVesting.sol	7D20EAD3CC91976C1A4E4F3C0455EFD3

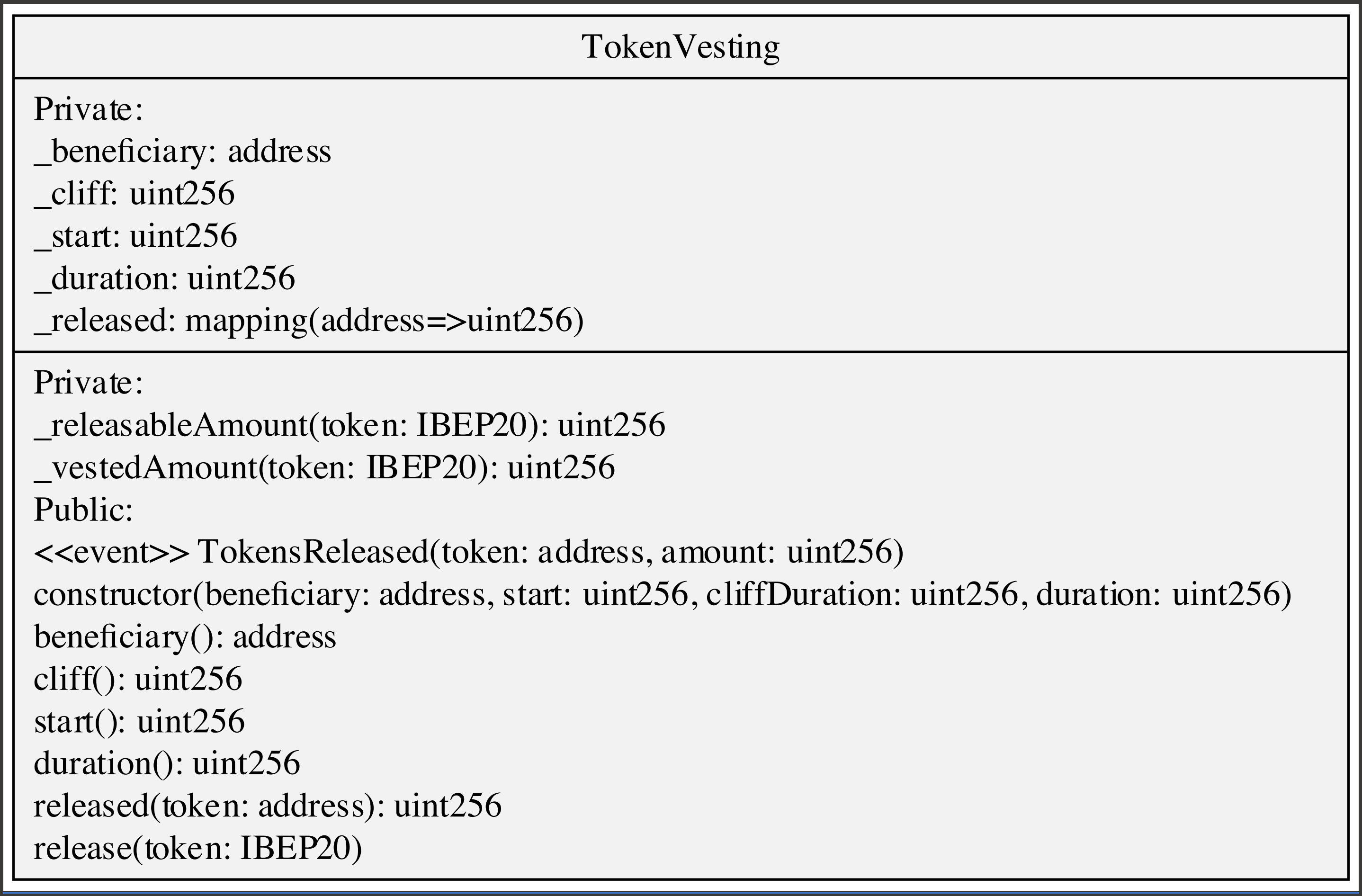
Vulnerability Summary

● Informational severity Issues	1
● Low severity issues	0
● Medium severity issues	0
● High severity issues	0

Total Issues
Informational
100%
🟩 Informational
🟨 Low
🟧 Medium
🟥 High

UML

TokenVesting.sol

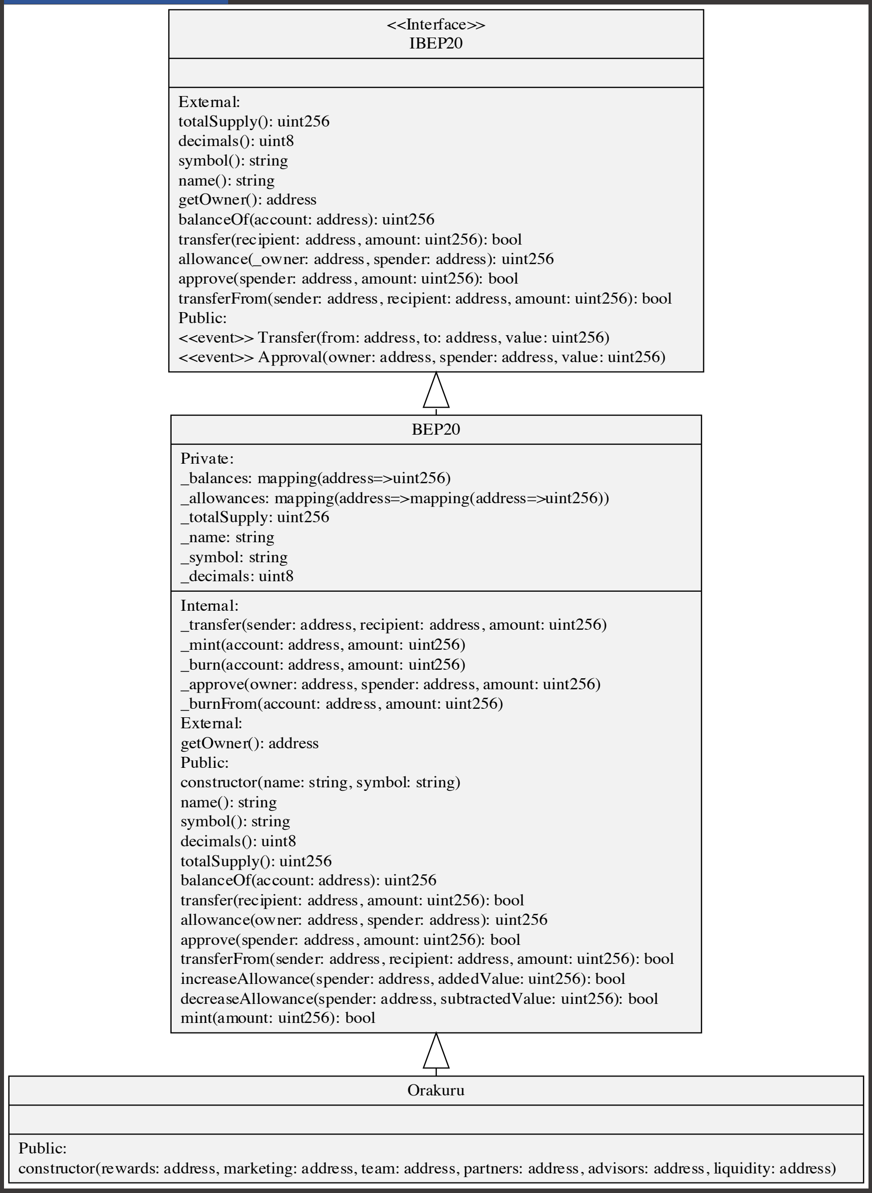


TokenVesting

Private:
_beneficiary: address
_cliff: uint256
_start: uint256
_duration: uint256
_released: mapping(address=>uint256)

Private:
_releasableAmount(token: IBEP20): uint256
_vestedAmount(token: IBEP20): uint256
Public:
<<event>> TokensReleased(token: address, amount: uint256)
constructor(beneficiary: address, start: uint256, cliffDuration: uint256, duration: uint256)
beneficiary(): address
cliff(): uint256
start(): uint256
duration(): uint256
released(token: address): uint256
release(token: IBEP20)

Orakuru.sol


<<Interface>>
IBEP20
External:
totalSupply(): uint256
decimals(): uint8
symbol(): string
name(): string
getOwner(): address
balanceOf(account: address): uint256
transfer(recipient: address, amount: uint256): bool
allowance(_owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(sender: address, recipient: address, amount: uint256): bool
Public:
<<event>> Transfer(from: address, to: address, value: uint256)
<<event>> Approval(owner: address, spender: address, value: uint256)
BEP20
Private:
_balances: mapping(address=>uint256)
_allowances: mapping(address=>mapping(address=>uint256))
_totalSupply: uint256
_name: string
_symbol: string
_decimals: uint8
Internal:
_transfer(sender: address, recipient: address, amount: uint256)
_mint(account: address, amount: uint256)
_burn(account: address, amount: uint256)
_approve(owner: address, spender: address, amount: uint256)
_burnFrom(account: address, amount: uint256)
External:
getOwner(): address
Public:
constructor(name: string, symbol: string)
name(): string
symbol(): string
decimals(): uint8
totalSupply(): uint256
balanceOf(account: address): uint256
transfer(recipient: address, amount: uint256): bool
allowance(owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(sender: address, recipient: address, amount: uint256): bool
increaseAllowance(spender: address, addedValue: uint256): bool
decreaseAllowance(spender: address, subtractedValue: uint256): bool
mint(amount: uint256): bool
Orakuru
Public:
constructor(rewards: address, marketing: address, team: address, partners: address, advisors: address, liquidity: address)

BEP-20’s Conformance

This test checks for BEP-20’s conformance.

- All the functions are present
- All the events are present
- Functions return the correct type
- Functions that must be view are view
- Events' parameters are correctly indexed
- The functions emit the events
- Derived contracts do not break the conformance

Function	present	type	Correct Return value	events	
totalSupply	✅	✅ view	✅		
balanceOf(address)	✅	✅ view	✅		
transfer(address,uint256)	✅	✅ external	✅	✅ Transfer	
transferFrom(address, address, uint256)	✅	✅ external	✅	✅ Transfer	
approve(address,uint256)	✅	✅ external	✅	✅ Approval	
allowance(address, address)	✅	✅ view	✅		
name	✅	✅ view	✅		
symbol	✅	✅ view	✅		

Check Events:

✅ Transfer

✅ Approve

The contract that was tested is the token’s contract: Orakuru.sol

Findings

Orakuru.sol

Description – The official token contract.

Summary

S

● Informational severity Issues	0
● Low severity issues	0
● Medium severity issues	0
● High severity issues	0

TokenVesting.sol

Description – A token holder contract that can release its token balance gradually (vesting period).

Issue #1:

Type	Severity	Location
Gas Optimization	● Informational	TokenVesting.sol

Description:

The public **`beneficiary`**, **`cliff`**, **`start`**, **`duration`** , **`released`** **,** and **`release`** functions should be declared as external.

Recommendation:

These functions are only called from outside of the contract, consider using the external attribute instead of public to save gas.

Summary

S

● Informational severity Issues	1
● Low severity issues	0
● Medium severity issues	0
● High severity issues	0