



AUDIT REPORT SecureWise

RICE TOKEN



Table of Contents

02	Disclaimer
03	Overview
04	Quick Result
05	Auditing Approach and Methodologies
06	Automated Analysis
11	Inheritance Graph
13	Contract Summary
14	Manual Review

Disclaimer

SecureWise provides the smart contract audit of solidity. Audit and report are for informational purposes only and not, nor should be considered, as an endorsement to engage with, invest in, participate, provide an incentive, or disapprove, criticise, discourage, or purport to provide an opinion on any particular project or team.

This audit report doesn't provide any warranty or guarantee regarding the nature of the technology analysed. These reports, in no way, provide investment advice, nor should be used as investment advice of any sort. Investors must always do their own research and manage their risk.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and SecureWise and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) SecureWise owe no duty of care towards you or any other person, nor does SecureWise make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SecureWise hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SecureWise hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SecureWise, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

Overview

Token Name: RiceToken(**RICE**)

Methodology: Automated Analysis, Manual Code Review

Language: Solidity

Contract Address: 0xA5d20d84deeEd3850a7538e993E3b2592bf997A0

ContractLink: <https://scan.coredao.org/address/0xA5d20d84deeEd3850a7538e993E3b2592bf997A0>

Network: Core

Supply: 5.000.000

Website: <https://core.riceprotocol.org/>

Twitter: <https://twitter.com/protocolrice>

Telegram: https://t.me/rice_pro

Report Date: March 3, 2023

Quick Result

SecureWise has applied the automated and manual analysis of Smart Contract and were reviewed for common contract vulnerabilities and centralized exploits



There is no Owner privileges

RiceToken(RICE) has succesfully **PASSED** the smart contract audit with **high score**.

Auditing Approach and Methodologies

SecureWise has performed starting with analyzing the code, issues, code quality, and libraries. Reviewed line-by-line by our team. Finding any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

Methodology

- Understanding the size, scope and functionality of your project's source code
- Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Testing and automated analysis of the Smart Contract to determine proper logic has been followed throughout the whole process
- Deploying the code on testnet using multiple live test
- Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.
- Checking whether all the libraries used in the code are on the latest version.

Goals

Smart Contract System is secure, resilient and working according to the specifications and without any vulnerabilities.



Risk Classification

High: Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.

Medium: Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fixed as soon as possible.

Low: Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.

Automated Analysis

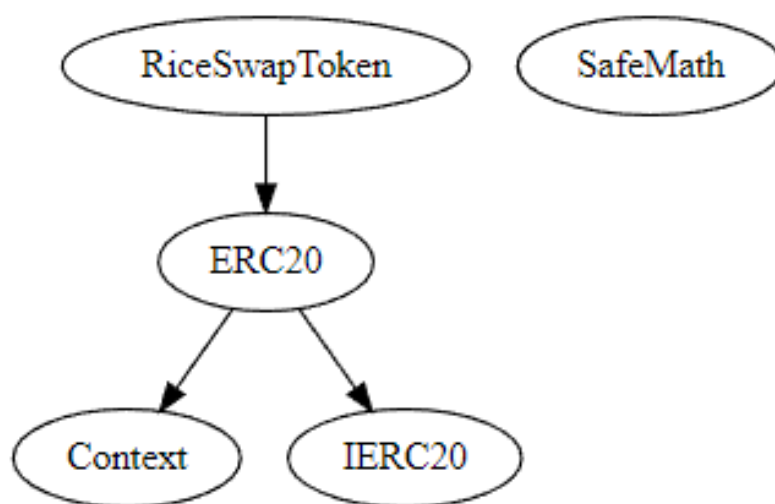
Symbol	Meaning
	Function can modify state
	Function is payable

Context	Implementation			
L	_msgSender	Internal 		
L	_msgData	Internal 		
IERC20	Interface			
L	totalSupply	External 		NO 
L	balanceOf	External 		NO 
L	transfer	External 		NO 
L	allowance	External 		NO 
L	approve	External 		NO 
L	transferFrom	External 		NO 
SafeMath	Library			
L	tryAdd	Internal 		
L	trySub	Internal 		
L	tryMul	Internal 		
L	tryDiv	Internal 		
L	tryMod	Internal 		
L	add	Internal 		
L	sub	Internal 		
L	mul	Internal 		
L	div	Internal 		
L	mod	Internal 		
L	sub	Internal 		
L	div	Internal 		
L	mod	Internal 		
ERC20	Implementation	Context, IERC20		
L		Public 		NO 
L	name	Public 		NO 
L	symbol	Public 		NO 
L	decimals	Public 		NO 

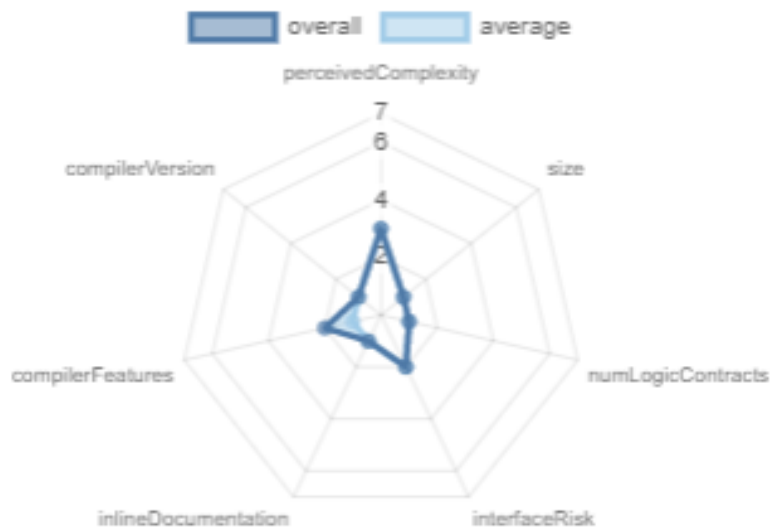
Automated Analysis

L	totalSupply	Public !		NO !
L	balanceOf	Public !		NO !
L	transfer	Public !	●	NO !
L	allowance	Public !		NO !
L	approve	Public !	●	NO !
L	transferFrom	Public !	●	NO !
L	increaseAllowance	Public !	●	NO !
L	decreaseAllowance	Public !	●	NO !
L	_transfer	Internal 🔒	●	
L	_mint	Internal 🔒	●	
L	_burn	Internal 🔒	●	
L	_approve	Internal 🔒	●	
L	_setupDecimals	Internal 🔒	●	
L	_beforeTokenTransfer	Internal 🔒	●	
RiceSwapToken	Implementation	ERC20		
L		Public !	●	ERC20

Inheritance Graph





Risk





Source Lines



Contract Summary

Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
4	1	618	567	176	389	114	
4	1	618	567	176	389	114	

Components

 Contracts	 Libraries	 Interfaces	 Abstract
2	1	1	1

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.





 Public	 Payable
17	0







External	Internal	Private	Pure	View
6	37	0	13	11

StateVariables

Total	 Public
6	0

Capabilities

Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
^0.7.0				

 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 Ecrecover	 New/Create/Create2

 TryCatch	 Σ Unchecked

Manual Review

Access Modifiers Vulnerabilities

```
transferFrom()  
decimals()  
decreaseAllowance()  
symbol()  
balanceOf()  
transfer()  
increaseAllowance()  
name()  
approve()  
allowance()
```

These functions are used as public instead of external.

Recommendation

Access control identifiers must be authenticated and set adequately to avoid possible vulnerabilities

Out date compiler version

```
pragma solidity ^0.7.0;
```

Compiler is set an outdated version.

Recommendation

Set and use new versions

Floating Pragma

```
pragma solidity ^0.7.0;
```

Recommendation

Lock the pragma version and also consider known bugs (<https://github.com/ethereum/solidity/releases>) for the compiler version that is chosen.



AUDIT REPORT SecureWise



<https://securewise.info/>



<https://t.me/securewisehub>



<https://github.com/securewise>