Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	Raku Doge
Website:	https://rakucoin.com
Platform:	Ethereum
Language:	Solidity
Date:	July 12th, 2021

Table of contents

Introduction

We were contracted by the Raku Doge team to perform the Security audit of the RAKUDOGE Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on July 12th, 2021.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

Raku Doge is Raku Coin's Reward Eco-System token with a low supply to be used with Raku Vault for harvest or staking. Raku doge will have its own unique Tokenomics with a low supply that reward Holders for holding on to the Ecosystem Reward Token. Raku Coin Holders will be able to harvest or stake for Raku Doge based on a Supply/Demand Pool System.

Audit scope

Name	**Code Review and Security Analysis Report for Raku Doge (RAKUDOGE) Token Smart Contract**
Platform	**Ethereum / Solidity**
File	RakuCoin.sol
Smart Contract Online Code	https://etherscan.io/address/0x3AE21a74df8FE75a6731cb5656ee9b7A5e6B9685#code
File MD5 Hash	0A44B7C40E6EE32204E68DB364E620BF
Audit Date	July 12th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Name: Raku Doge	**YES, This is valid.**
Symbol: RAKUDOGE	**YES, This is valid.**
Decimal: 18	**YES, This is valid.**
Buy Tax Fee: 3%	**YES, This is valid.The smart contract owner can change these fees anytime**
Buy Stake Tax Fee : 4%	
Buy Marketing Pool Fee: 3%	
Sell Tax Fee: 5%	
Sell Stake Tax Fee: 7%	
Sell Marketing Pool Fee: 3%	
● swapTokensForEth: The lockTheSwap can generate the uniswap pair path of token to weth.	**YES, This is valid. The smart contract owner controls these functions, so the owner must handle the private key of the owner's wallet very securely. Because if the private key is compromised, then it will create problems.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contract is **Secured**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.

Insecure
Poor secured
Secure
Well-secured
You are here

We found 0 critical, 0 high, 0 medium and 3 low and some very low level technical issues.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Moderated
	Other code specification issues	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 1 smart contract. This smart contract also contains Libraries, Smart contracts inherits and Interfaces. This is a compact and well written contract.

The libraries in the Raku Doge Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Raku Doge Token.

The Raku Doge Token team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are **not well** commented on smart contracts.

Documentation

We were given Raku Doge Token smart contract code in the form of an Etherscan web link. The hashes of that code are mentioned above in the table.

As mentioned above, some code parts are **not well** commented. So it is difficult to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://rakucoin.com which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

(1) Interface

(a) IERC20

(b) IUniswapV2Factory

(c) IUniswapV2Pair

(d) IUniswapV2Router01

(e) Uniswap V2 Router 02

(2) Inherited contracts

(a) Context

(b) Ownable

(c) IERC20

(3) Usages

(a) using SafeMath for uint256;

(b) using Address for address;

(4) Events

(a) event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);

(b) event SwapEnabledUpdated(bool enabled);

(5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	lockTheSwap	modifier	Passed	No Issue
2	openTrading	external	Critical operation lacks event log	Refer Audit Findings
3	setWalletlimit	external	access only Owner	No Issue
4	addDestLimit	external	access only Owner	No Issue
5	removeDestLimit	external	access only Owner	No Issue
6	isBlackListed	write	Passed	No Issue
7	name	read	Passed	No Issue
8	symbol	read	Passed	No Issue
9	decimals	read	Passed	No Issue
10	totalSupply	read	Passed	No Issue
11	balanceOf	read	Passed	No Issue
12	transfer	write	Passed	No Issue

13	allowance	read	Passed	No Issue
14	approve	write	Passed	No Issue
15	transferFrom	write	Passed	No Issue
16	increaseAllowance	write	Passed	No Issue
17	decreaseAllowance	write	Passed	No Issue
18	isExcluded	read	Passed	No Issue
19	setExcludeFromFee	external	access only Owner	No Issue
20	totalFees	read	Passed	No Issue
21	RemoveSniper	external	Critical operation lacks event log	Refer Audit Findings
22	amnestySniper	external	Infinite loops possibility at multiple places	Refer Audit Findings
23	deliver	write	external instead of public	Refer Audit Findings
24	reflectionFromToken	read	Passed	No Issue
25	tokenFromReflection	read	Passed	No Issue
26	excludeAccount	external	Critical operation lacks event log	Refer Audit Findings
27	includeAccount	external	Critical operation lacks event log	Refer Audit Findings
28	removeAllFee	write	Critical operation lacks event log	Refer Audit Findings
29	restoreAllFee	write	Critical operation lacks event log	Refer Audit Findings
30	isExcludedFromFee	read	Passed	No Issue
31	_approve	write	Passed	No Issue
32	_transfer	write	Passed	No Issue
33	swapTokensForEth	write	Passed	No Issue
34	manualSwap	external	access only Owner	No Issue
35	manualSend	external	access only Owner	No Issue
36	setSwapEnabled	external	access only Owner	No Issue
37	_tokenTransfer	write	Passed	No Issue
38	_transferStandard	write	Passed	No Issue
39	_transferToExcluded	write	Passed	No Issue
40	_transferFromExcluded	write	Passed	No Issue
41	_transferBothExcluded	write	Passed	No Issue
42	_takeMarketingPool	write	Passed	No Issue
43	_takeStakePool	write	Passed	No Issue
44	_reflectFee	write	Passed	No Issue
45	_getValues	read	Passed	No Issue
46	_getValues2	read	Passed	No Issue
47	_getTValues	write	Passed	No Issue
48	_addonvalues	write	Passed	No Issue
49	_getRValues	write	Passed	No Issue
50	_getRate	write	Passed	No Issue
51	_getCurrentSupply	read	Passed	No Issue
52	_getTaxFee	read	Passed	No Issue

53	_getMaxTxAmount	read	Passed	No Issue
54	_getETHBalance	read	Passed	No Issue
55	_setTaxFee	external	access only Owner	No Issue
56	_setbuyTaxFee	external	access only Owner	No Issue
57	_setsellTaxFee	external	access only Owner	No Issue
58	_setStakeFee	external	access only Owner	No Issue
59	_setbuystakeFee	external	access only Owner	No Issue
60	_setsellstakeFee	external	access only Owner	No Issue
61	_setMarketingPoolFee	external	access only Owner	No Issue
62	_setbuyMarketingPoolFee	external	access only Owner	No Issue
63	_setsellMarketingPoolFee	external	access only Owner	No Issue
64	_setMarketingPoolWallet	external	access only Owner	No Issue
65	_setStakePoolAddress	external	access only Owner	No Issue
66	_setMaxTxAmount	external	access only Owner	No Issue
67	owner	read	Passed	No Issue
68	onlyOwner	modifier	Passed	No Issue
69	renounceOwnership	write	Possible to gain ownership	Refer Audit Findings
70	transferOwnership	write	access only Owner	No Issue
71	geUnlockTime	read	Passed	No Issue
72	lock	write	Possible to gain ownership	Refer Audit Findings
73	unlock	write	Possible to gain ownership	Refer Audit Findings
74	_msgSender	internal	Passed	No Issue
75	_msgData	internal	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

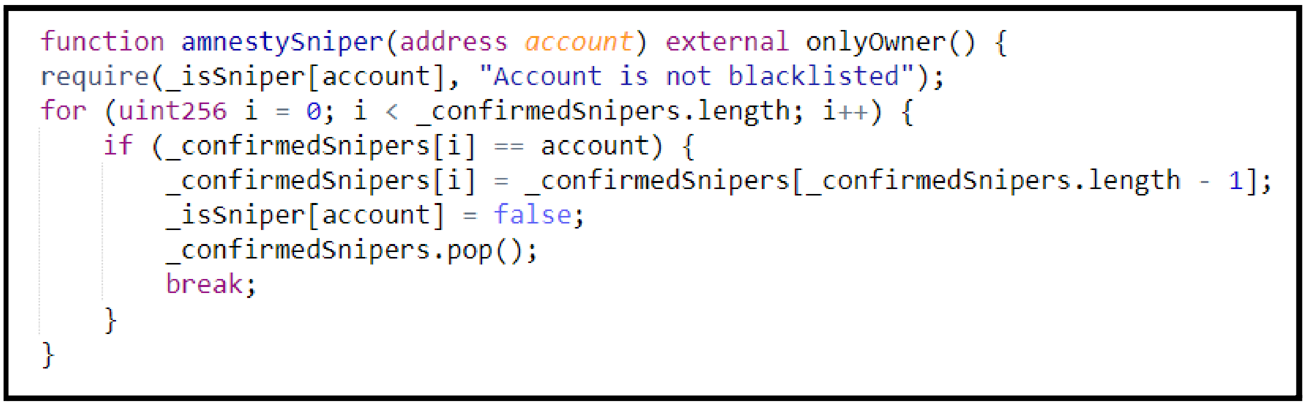
(1) Critical operation lacks event log:

Missing Events log for some functions:

- excludeAccount
- includeAccount
- removeAllFee

- restoreAllFee
- deliver
- openTrading
- RemoveSniper
- amnestySniper
- functions for setting values.

(2) Infinite loops possibility at multiple places:



```
function amnestySniper(address account) external onlyOwner() {
require(_isSniper[account], "Account is not blacklisted");
for (uint256 i = 0; i < _confirmedSnipers.length; i++) {
    if (_confirmedSnipers[i] == account) {
        _confirmedSnipers[i] = _confirmedSnipers[_confirmedSnipers.length - 1];
        _isSniper[account] = false;
        _confirmedSnipers.pop();
        break;
    }
}
```

There is an amnestySniper() function in the smart contracts, where _confirmedSnipers.length variable is used directly in the loops. It is recommended to put some kind of limits.

Resolution: So it does not go wild and create any scenario where it can hit the block gas limit.

(3) Possible to gain ownership:

Possible to gain ownership after renouncing the contract ownership. Owner can renounce ownership and make contract without owner but he can regain ownership by following the steps below:

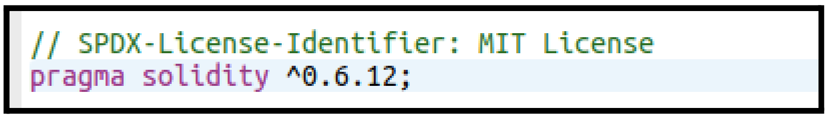
1. Owner calls the lock function in contract to set the current owner as _previousOwner.
2. Owner calls unlock to unlock the contract and set _owner = _previousOwner.
3. Owner called renounceOwnership to leave the contract without the owner.
4. Owner calls unlock to regain ownership.

.

Resolution: We suggest removing these lock/unlock functions as this seems not serving a great purpose. Otherwise, always renounce ownership first before calling the lock function.

Very Low / Discussion / Best practices:

(1) Use the latest solidity version:

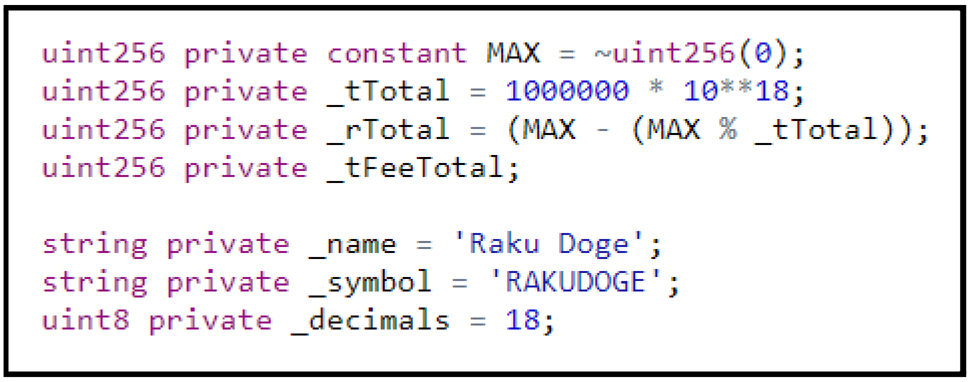


```
// SPDX-License-Identifier: MIT License
pragma solidity ^0.6.12;
```

Using the latest solidity will prevent any compiler level bugs.

Resolution: Please use 0.8.6 which is the latest version.

(2) Make variables constant



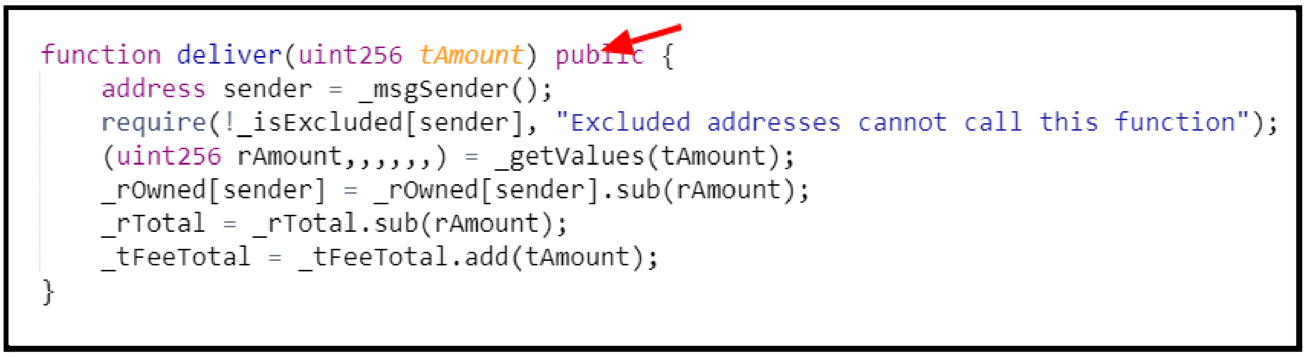
```
uint256 private constant MAX = ~uint256(0);
uint256 private _tTotal = 1000000 * 10**18;
uint256 private _rTotal = (MAX - (MAX % _tTotal));
uint256 private _tFeeTotal;

string private _name = 'Raku Doge';
string private _symbol = 'RAKUDOGE';
uint8 private _decimals = 18;
```

_name, _symbol, _decimals and _tTotal. These variables will be unchanged. So, please make it constant. It will save some gas.

Resolution: Declare those variables as constant. Just put a constant keyword. And define constants in the constructor.

(3) external instead of public:



```
function deliver(uint256 tAmount) public {
    address sender = _msgSender();
    require(!_isExcluded[sender], "Excluded addresses cannot call this function");
    (uint256 rAmount,,,,,,) = _getValues(tAmount);
    _rOwned[sender] = _rOwned[sender].sub(rAmount);
    _rTotal = _rTotal.sub(rAmount);
    _tFeeTotal = _tFeeTotal.add(tAmount);
}
```

If any function is not called from inside the smart contract, then it is better to declare it as external instead of public. As it saves some gas as well.

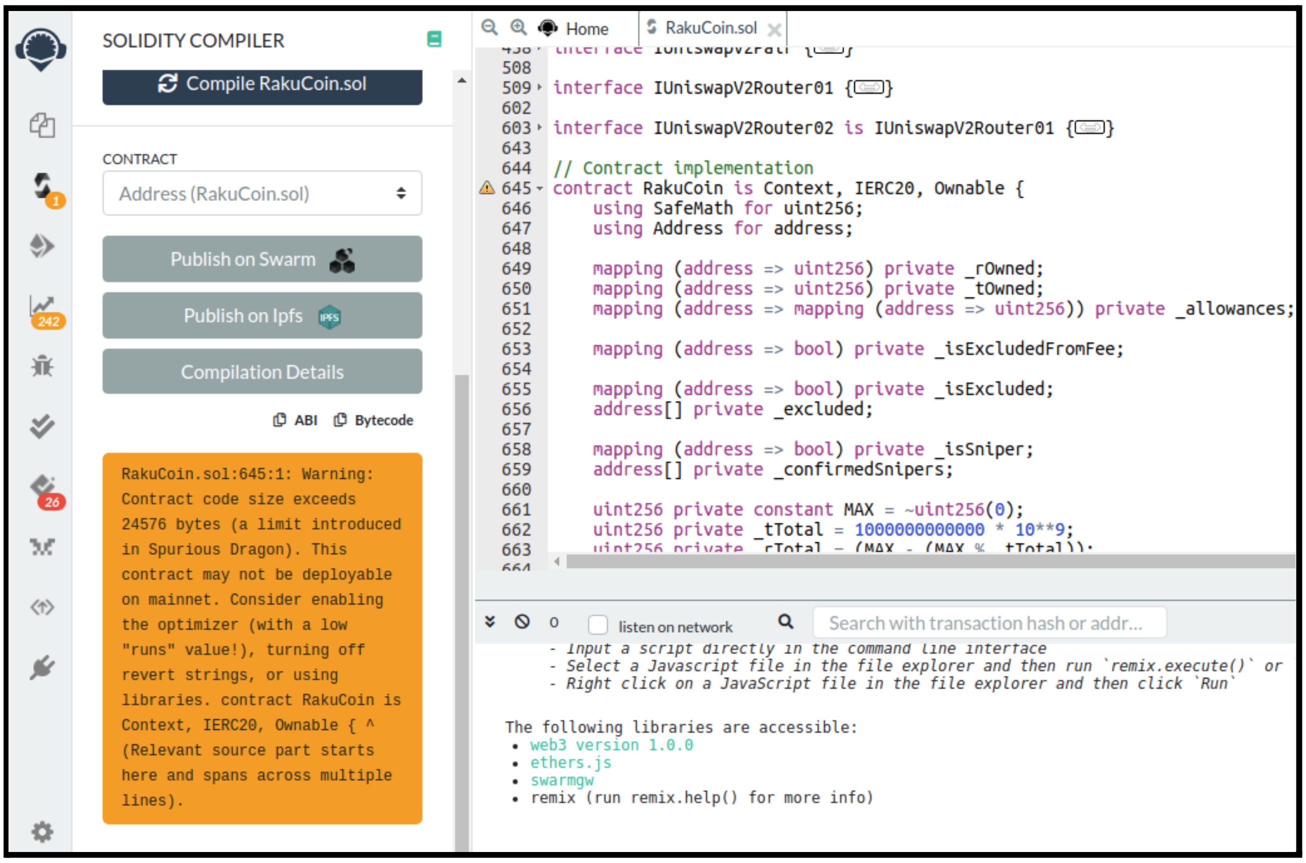
Resolution: make all public to external, which are not called from inside the contract.

(4) HardCoded Router address:

Router's address is hardcoded in code.

Resolution: Owner has to double confirm before deploying.

(5) MLR contract size limit:



```
RakuCoin.sol:645:1: Warning:
Contract code size exceeds
24576 bytes (a limit introduced
in Spurious Dragon). This
contract may not be deployable
on mainnet. Consider enabling
the optimizer (with a low
"runs" value!), turning off
revert strings, or using
libraries. contract RakuCoin is
Context, IERC20, Ownable { ^
(Relevant source part starts
here and spans across multiple
lines).
```

```
interface IUniswapV2Router01 {}

interface IUniswapV2Router02 is IUniswapV2Router01 {}

// Contract implementation
contract RakuCoin is Context, IERC20, Ownable {
    using SafeMath for uint256;
    using Address for address;

    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private _allowances;

    mapping (address => bool) private _isExcludedFromFee;

    mapping (address => bool) private _isExcluded;
    address[] private _excluded;

    mapping (address => bool) private _isSniper;
    address[] private _confirmedSnipers;

    uint256 private constant MAX = ~uint256(0);
    uint256 private _tTotal = 1000000000000 * 10**9;
```

```
 - Select a Javascript file in the file explorer and then run `remix.execute()` or
 - Right click on a JavaScript file in the file explorer and then click `Run`

The following libraries are accessible:
 • web3 version 1.0.0
 • ethers.js
 • swarmgw
 • remix (run remix.help() for more info)
```

Warning: Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon).

Resolution: Divide into multiple contracts and inherited.

Centralization

This smart contract has some functions which can be executed by Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- openTrading: The Owner can set tradingOpen status,swapEnabled status.
- setWalletlimit: The Owner can set the wallet limit.
- addDestLimit: The Owner can set uniswapOnly status true.
- removeDestLimit: The Owner can set uniswapOnly status false.
- setExcludeFromFee: The Owner can set excluded fee from wallet account.
- totalFees: The Owner can return total fees.
- RemoveSniper: The Owner can remove the sniper from account.
- amnestySniper: The Owner can check if the account is already blacklisted or not.
- excludeAccount: The Owner can check if the wallet account is excluded or not.
- includeAccount: The Owner can check if the wallet account is included or not.
- manualSwap: The Owner can be able to manually swap and send in case the token is highly valued and 5M becomes too much.
- manualSend: The Owner can be able to manually send wallet balance.
- setSwapEnabled: The Owner can set swap enabled status.
- _setTaxFee: The Owner can set a tax fee.
- _setbuyTaxFee: The Owner can set the buy(purchase) tax fee.
- _setsellTaxFee: The Owner can set the sales tax fee.
- _setStakeFee: The Owner can set a stake fee.
- _setbuystakeFee: The Owner can set a buy stake fee.
- _setsellstakeFee: The Owner can set a sell stake fee.
- _setMarketingPoolFee: The Owner can set marketing pool fee.
- _setbuyMarketingPoolFee: The Owner can set a buy marketing pool fee.
- _setsellMarketingPoolFee: The Owner can set a sell marketing pool fee.
- _setMarketingPoolWallet: The Owner can set a marketing pool wallet address.
- _setStakePoolAddress: The Owner can set a stake pool wallet address
- _setMaxTxAmount: The Owner can set a maximum tax amount.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those are fixed/acknowledged in the smart contracts. **So it is good to go for the production.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **" Secured".**

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:
In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:
Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

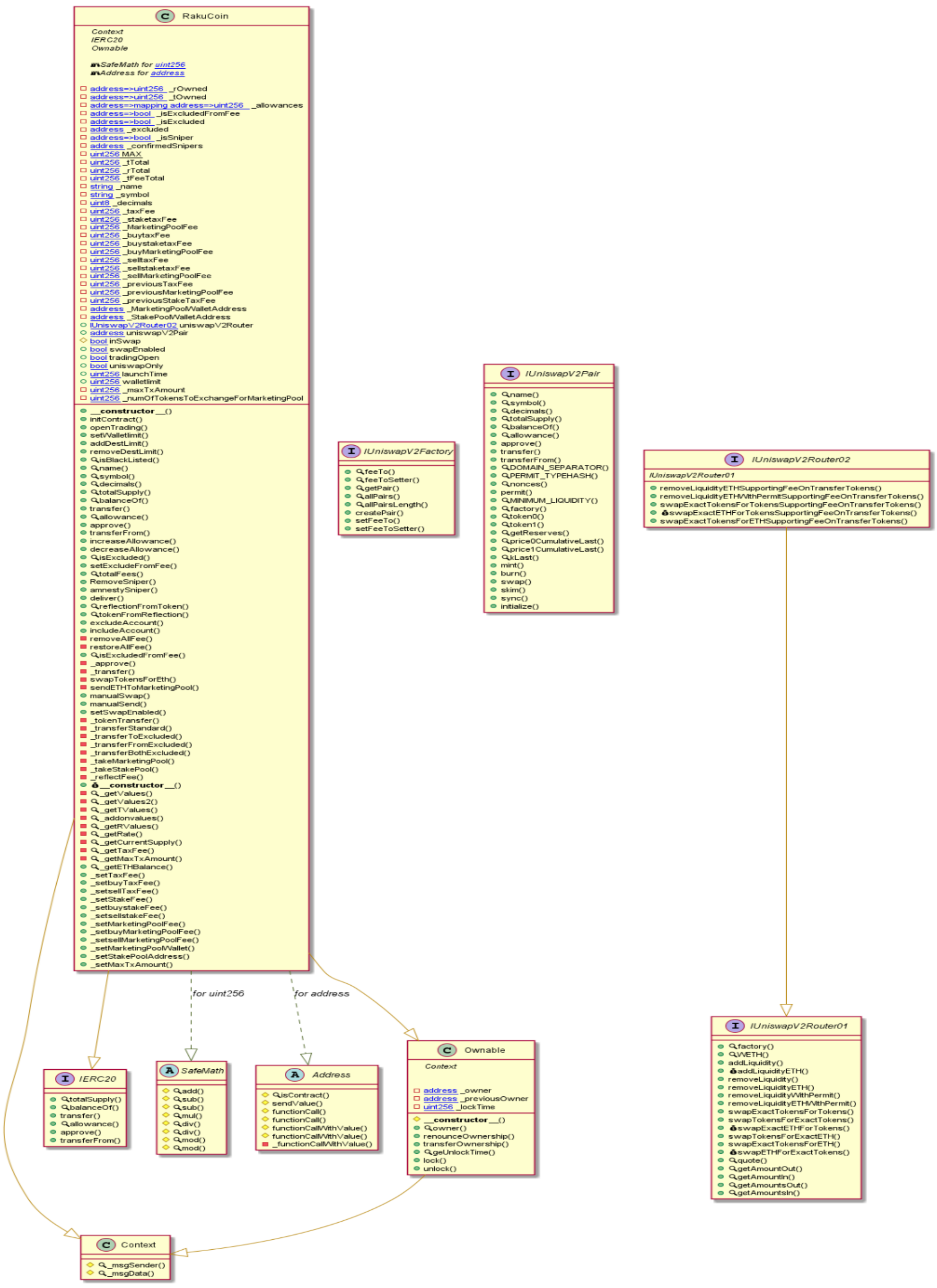
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - Raku Doge Token



Slither Results Log

Slither log >> RakuCoin.sol

```
INFO:Detectors:
RakuCoin.sendETHToMarketingPool(uint256) (RakuCoin.sol#1196-1199) sends eth to arbitrary user
	Dangerous calls:
	- _MarketingPoolWalletAddress.transfer(amount) (RakuCoin.sol#1197)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinatio
ns
INFO:Detectors:
Reentrancy in RakuCoin._transfer(address,address,uint256) (RakuCoin.sol#1065-1176):
	External calls:
	- swapTokensForEth(contractTokenBalance) (RakuCoin.sol#1137)
		-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
his),block.timestamp) (RakuCoin.sol#1187-1193)
	External calls sending eth:
	- sendETHToMarketingPool(address(this).balance) (RakuCoin.sol#1141)
		- _MarketingPoolWalletAddress.transfer(amount) (RakuCoin.sol#1197)
	State variables written after the call(s):
	- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
		- _rOwned[address(_StakePoolWalletAddress)] =
_rOwned[address(_StakePoolWalletAddress)].add(rStakePool) (RakuCoin.sol#1293)
		- _rOwned[address(this)] = _rOwned[address(this)].add(rMarketingPool) (RakuCoin.sol#1285)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (RakuCoin.sol#1249)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (RakuCoin.sol#1239)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (RakuCoin.sol#1273)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (RakuCoin.sol#1240)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (RakuCoin.sol#1262)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (RakuCoin.sol#1263)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (RakuCoin.sol#1251)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (RakuCoin.sol#1275)
	- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
		- _rTotal = _rTotal.sub(rFee) (RakuCoin.sol#1299)
	- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
		- _tOwned[address(_StakePoolWalletAddress)] =
_tOwned[address(_StakePoolWalletAddress)].add(tStakePool) (RakuCoin.sol#1295)
		- _tOwned[address(this)] = _tOwned[address(this)].add(tMarketingPool) (RakuCoin.sol#1287)
		- _tOwned[sender] = _tOwned[sender].sub(tAmount) (RakuCoin.sol#1272)
		- _tOwned[sender] = _tOwned[sender].sub(tAmount) (RakuCoin.sol#1261)
		- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (RakuCoin.sol#1250)
		- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (RakuCoin.sol#1274)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
RakuCoin._setTaxFee(uint256) (RakuCoin.sol#1376-1380) contains a tautology or contradiction:
	- require(bool,string)(taxFee >= 0 && taxFee <= 10,taxFee should be in 0 - 10) (RakuCoin.sol#1377)
RakuCoin._setbuyTaxFee(uint256) (RakuCoin.sol#1382-1385) contains a tautology or contradiction:
	- require(bool,string)(buytaxFee >= 0 && buytaxFee <= 10,taxFee should be in 0 - 10)
(RakuCoin.sol#1383)
RakuCoin._setsellTaxFee(uint256) (RakuCoin.sol#1387-1390) contains a tautology or contradiction:
	- require(bool,string)(selltaxFee >= 0 && selltaxFee <= 10,taxFee should be in 0 - 10)
(RakuCoin.sol#1388)
RakuCoin._setStakeFee(uint256) (RakuCoin.sol#1392-1396) contains a tautology or contradiction:
	- require(bool,string)(stakeFee >= 0 && stakeFee <= 11,stakeFee should be in 0 - 11)
(RakuCoin.sol#1393)
RakuCoin._setbuystakeFee(uint256) (RakuCoin.sol#1397-1400) contains a tautology or contradiction:
	- require(bool,string)(buystakeFee >= 0 && buystakeFee <= 11,stakeFee should be in 0 - 11)
(RakuCoin.sol#1398)
RakuCoin._setsellstakeFee(uint256) (RakuCoin.sol#1401-1404) contains a tautology or contradiction:
	- require(bool,string)(sellstakeFee >= 0 && sellstakeFee <= 11,stakeFee should be in 0 - 11)
(RakuCoin.sol#1402)
RakuCoin._setMarketingPoolFee(uint256) (RakuCoin.sol#1406-1410) contains a tautology or contradiction:
	- require(bool,string)(MarketingPoolFee >= 0 && MarketingPoolFee <= 11,MarketingPoolFee should be
in 0 - 11) (RakuCoin.sol#1407)
RakuCoin._setbuyMarketingPoolFee(uint256) (RakuCoin.sol#1412-1415) contains a tautology or
contradiction:
	- require(bool,string)(buyMarketingPoolFee >= 0 && buyMarketingPoolFee <= 11,MarketingPoolFee
should be in 0 - 11) (RakuCoin.sol#1413)
RakuCoin._setsellMarketingPoolFee(uint256) (RakuCoin.sol#1417-1420) contains a tautology or
contradiction:
	- require(bool,string)(sellMarketingPoolFee >= 0 && sellMarketingPoolFee <= 11,MarketingPoolFee
should be in 0 - 11) (RakuCoin.sol#1418)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
RakuCoin.allowance(address,address).owner (RakuCoin.sol#930) shadows:
	- Ownable.owner() (RakuCoin.sol#387-389) (function)
RakuCoin._approve(address,address,uint256).owner (RakuCoin.sol#1057) shadows:
	- Ownable.owner() (RakuCoin.sol#387-389) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
RakuCoin.constructor(address,address).MarketingPoolWalletAddress (RakuCoin.sol#712) lacks a
zero-check on :
		- _MarketingPoolWalletAddress = MarketingPoolWalletAddress (RakuCoin.sol#713)
RakuCoin.constructor(address,address).StakePoolWalletAddress (RakuCoin.sol#712) lacks a zero-check on
:
		- _StakePoolWalletAddress = StakePoolWalletAddress (RakuCoin.sol#714)
RakuCoin._setMarketingPoolWallet(address).MarketingPoolWalletAddress (RakuCoin.sol#1422) lacks a
zero-check on :
		- _MarketingPoolWalletAddress = MarketingPoolWalletAddress (RakuCoin.sol#1423)
RakuCoin._setStakePoolAddress(address).StakePoolAddress (RakuCoin.sol#1426) lacks a zero-check on :
		- _StakePoolWalletAddress = StakePoolAddress (RakuCoin.sol#1427)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in RakuCoin._transfer(address,address,uint256) (RakuCoin.sol#1065-1176):
	External calls:
	- swapTokensForEth(contractTokenBalance) (RakuCoin.sol#1137)
		-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
his),block.timestamp) (RakuCoin.sol#1187-1193)
	External calls sending eth:
	- sendETHToMarketingPool(address(this).balance) (RakuCoin.sol#1141)
		- _MarketingPoolWalletAddress.transfer(amount) (RakuCoin.sol#1197)
	State variables written after the call(s):
	- removeAllFee() (RakuCoin.sol#1155)
		- _MarketingPoolFee = 0 (RakuCoin.sol#1043)
	- _MarketingPoolFee = _buyMarketingPoolFee (RakuCoin.sol#1158)
	- removeAllFee() (RakuCoin.sol#1163)
		- _MarketingPoolFee = 0 (RakuCoin.sol#1043)
	- _MarketingPoolFee = _sellMarketingPoolFee (RakuCoin.sol#1166)
	- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
		- _MarketingPoolFee = _previousMarketingPoolFee (RakuCoin.sol#1049)
		- _MarketingPoolFee = 0 (RakuCoin.sol#1043)
	- removeAllFee() (RakuCoin.sol#1155)
		- _previousMarketingPoolFee = _MarketingPoolFee (RakuCoin.sol#1039)
	- removeAllFee() (RakuCoin.sol#1163)
		- _previousMarketingPoolFee = _MarketingPoolFee (RakuCoin.sol#1039)
	- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
		- _previousMarketingPoolFee = _MarketingPoolFee (RakuCoin.sol#1039)
	- removeAllFee() (RakuCoin.sol#1155)
		- _previousStakeTaxFee = _staketaxFee (RakuCoin.sol#1040)
	- removeAllFee() (RakuCoin.sol#1163)
```

```
		- _previousStakeTaxFee = _staketaxFee (RakuCoin.sol#1040)
	- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
		- _previousStakeTaxFee = _staketaxFee (RakuCoin.sol#1040)
	- removeAllFee() (RakuCoin.sol#1155)
		- _previousTaxFee = _taxFee (RakuCoin.sol#1038)
	- removeAllFee() (RakuCoin.sol#1163)
		- _previousTaxFee = _taxFee (RakuCoin.sol#1038)
	- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
		- _previousTaxFee = _taxFee (RakuCoin.sol#1038)
	- removeAllFee() (RakuCoin.sol#1155)
		- _staketaxFee = 0 (RakuCoin.sol#1044)
	- _staketaxFee = _buystaketaxFee (RakuCoin.sol#1157)
	- removeAllFee() (RakuCoin.sol#1163)
		- _staketaxFee = 0 (RakuCoin.sol#1044)
	- _staketaxFee = _sellstaketaxFee (RakuCoin.sol#1165)
	- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
		- _staketaxFee = _previousStakeTaxFee (RakuCoin.sol#1050)
		- _staketaxFee = 0 (RakuCoin.sol#1044)
	- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
		- _tFeeTotal = _tFeeTotal.add(tFee) (RakuCoin.sol#1300)
	- removeAllFee() (RakuCoin.sol#1155)
		- _taxFee = 0 (RakuCoin.sol#1042)
	- _taxFee = _buytaxFee (RakuCoin.sol#1156)
	- removeAllFee() (RakuCoin.sol#1163)
		- _taxFee = 0 (RakuCoin.sol#1042)
	- _taxFee = _selltaxFee (RakuCoin.sol#1164)
	- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
		- _taxFee = _previousTaxFee (RakuCoin.sol#1048)
		- _taxFee = 0 (RakuCoin.sol#1042)
Reentrancy in RakuCoin.constructor(address,address) (RakuCoin.sol#712-731):
	External calls:
	- uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())
(RakuCoin.sol#720-721)
	State variables written after the call(s):
	- _isExcludedFromFee[owner()] = true (RakuCoin.sol#727)
	- _isExcludedFromFee[address(this)] = true (RakuCoin.sol#728)
	- uniswapV2Router = _uniswapV2Router (RakuCoin.sol#724)
Reentrancy in RakuCoin.transferFrom(address,address,uint256) (RakuCoin.sol#939-943):
	External calls:
	- _transfer(sender,recipient,amount) (RakuCoin.sol#940)
		-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
his),block.timestamp) (RakuCoin.sol#1187-1193)
	External calls sending eth:
	- _transfer(sender,recipient,amount) (RakuCoin.sol#940)
		- _MarketingPoolWalletAddress.transfer(amount) (RakuCoin.sol#1197)
	State variables written after the call(s):
	- _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer
amount exceeds allowance)) (RakuCoin.sol#941)
		- _allowances[owner][spender] = amount (RakuCoin.sol#1061)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in RakuCoin._transfer(address,address,uint256) (RakuCoin.sol#1065-1176):
	External calls:
	- swapTokensForEth(contractTokenBalance) (RakuCoin.sol#1137)
		-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
his),block.timestamp) (RakuCoin.sol#1187-1193)
	External calls sending eth:
	- sendETHToMarketingPool(address(this).balance) (RakuCoin.sol#1141)
		- _MarketingPoolWalletAddress.transfer(amount) (RakuCoin.sol#1197)
	Event emitted after the call(s):
	- Transfer(sender,recipient,tTransferAmount) (RakuCoin.sol#1244)
		- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
	- Transfer(sender,recipient,tTransferAmount) (RakuCoin.sol#1255)
		- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
	- Transfer(sender,recipient,tTransferAmount) (RakuCoin.sol#1267)
		- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
	- Transfer(sender,recipient,tTransferAmount) (RakuCoin.sol#1279)
		- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
Reentrancy in RakuCoin.constructor(address,address) (RakuCoin.sol#712-731):
	External calls:
	- uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())
(RakuCoin.sol#720-721)
	Event emitted after the call(s):
	- Transfer(address(0),_msgSender(),_tTotal) (RakuCoin.sol#730)
Reentrancy in RakuCoin.transferFrom(address,address,uint256) (RakuCoin.sol#939-943):
	External calls:
	- _transfer(sender,recipient,amount) (RakuCoin.sol#940)
		-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
his),block.timestamp) (RakuCoin.sol#1187-1193)
	External calls sending eth:
	- _transfer(sender,recipient,amount) (RakuCoin.sol#940)
		- _MarketingPoolWalletAddress.transfer(amount) (RakuCoin.sol#1197)
	Event emitted after the call(s):
	- Approval(owner,spender,amount) (RakuCoin.sol#1062)
		- _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20:
transfer amount exceeds allowance)) (RakuCoin.sol#941)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (RakuCoin.sol#434-439) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(now > _lockTime,Contract is locked until 7 days) (RakuCoin.sol#436)
RakuCoin._transfer(address,address,uint256) (RakuCoin.sol#1065-1176) uses timestamp for comparisons
	Dangerous comparisons:
	- block.timestamp < launchTime + 5 (RakuCoin.sol#1109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (RakuCoin.sol#251-260) uses assembly
	- INLINE ASM (RakuCoin.sol#258)
Address._functionCallWithValue(address,bytes,uint256,string) (RakuCoin.sol#344-365) uses assembly
	- INLINE ASM (RakuCoin.sol#357-360)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (RakuCoin.sol#344-365) is never used and
should be removed
Address.functionCall(address,bytes) (RakuCoin.sol#304-306) is never used and should be removed
Address.functionCall(address,bytes,string) (RakuCoin.sol#314-316) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (RakuCoin.sol#329-331) is never used and should be
removed
Address.functionCallWithValue(address,bytes,uint256,string) (RakuCoin.sol#339-342) is never used and
should be removed
Address.isContract(address) (RakuCoin.sol#251-260) is never used and should be removed
Address.sendValue(address,uint256) (RakuCoin.sol#278-284) is never used and should be removed
Context._msgData() (RakuCoin.sol#13-16) is never used and should be removed
RakuCoin._getMaxTxAmount() (RakuCoin.sol#1368-1370) is never used and should be removed
RakuCoin._getTaxFee() (RakuCoin.sol#1364-1366) is never used and should be removed
SafeMath.mod(uint256,uint256) (RakuCoin.sol#211-213) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (RakuCoin.sol#227-230) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
RakuCoin._rTotal (RakuCoin.sol#663) is set pre-construction with a non-constant function or state variable:
	- (MAX - (MAX % _tTotal))
RakuCoin._previousTaxFee (RakuCoin.sol#682) is set pre-construction with a non-constant function or state
variable:
	- _taxFee
```

```
RakuCoin._previousMarketingPoolFee (RakuCoin.sol#683) is set pre-construction with a non-constant
function or state variable:
        - _MarketingPoolFee
RakuCoin._previousStakeTaxFee (RakuCoin.sol#684) is set pre-construction with a non-constant function or
state variable:
        - _staketaxFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (RakuCoin.sol#278-284):
        - (success) = recipient.call{value: amount}() (RakuCoin.sol#282)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (RakuCoin.sol#344-365):
        - (success,returndata) = target.call{value: weiValue}(data) (RakuCoin.sol#348)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (RakuCoin.sol#473) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (RakuCoin.sol#474) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (RakuCoin.sol#491) is not in mixedCase
Function IUniswapV2Router01.WETH() (RakuCoin.sol#511) is not in mixedCase
Parameter RakuCoin.setWalletlimit(uint256)._walletlimit (RakuCoin.sol#887) is not in mixedCase
Function RakuCoin.RemoveSniper(address) (RakuCoin.sol#967-972) is not in mixedCase
Function RakuCoin._getETHBalance() (RakuCoin.sol#1372-1374) is not in mixedCase
Function RakuCoin._setTaxFee(uint256) (RakuCoin.sol#1376-1380) is not in mixedCase
Function RakuCoin._setbuyTaxFee(uint256) (RakuCoin.sol#1382-1385) is not in mixedCase
Function RakuCoin._setsellTaxFee(uint256) (RakuCoin.sol#1387-1390) is not in mixedCase
Function RakuCoin._setStakeFee(uint256) (RakuCoin.sol#1392-1396) is not in mixedCase
Function RakuCoin._setbuystakeFee(uint256) (RakuCoin.sol#1397-1400) is not in mixedCase
Function RakuCoin._setsellstakeFee(uint256) (RakuCoin.sol#1401-1404) is not in mixedCase
Function RakuCoin._setMarketingPoolFee(uint256) (RakuCoin.sol#1406-1410) is not in mixedCase
Parameter RakuCoin._setMarketingPoolFee(uint256).MarketingPoolFee (RakuCoin.sol#1406) is not in
mixedCase
Function RakuCoin._setbuyMarketingPoolFee(uint256) (RakuCoin.sol#1412-1415) is not in mixedCase
Function RakuCoin._setsellMarketingPoolFee(uint256) (RakuCoin.sol#1417-1420) is not in mixedCase
Function RakuCoin._setMarketingPoolWallet(address) (RakuCoin.sol#1422-1424) is not in mixedCase
Parameter RakuCoin._setMarketingPoolWallet(address).MarketingPoolWalletAddress (RakuCoin.sol#1422)
is not in mixedCase
Function RakuCoin._setStakePoolAddress(address) (RakuCoin.sol#1426-1428) is not in mixedCase
Parameter RakuCoin._setStakePoolAddress(address).StakePoolAddress (RakuCoin.sol#1426) is not in
mixedCase
Function RakuCoin._setMaxTxAmount(uint256) (RakuCoin.sol#1430-1432) is not in mixedCase
Variable RakuCoin._MarketingPoolFee (RakuCoin.sol#674) is not in mixedCase
Variable RakuCoin._MarketingPoolWalletAddress (RakuCoin.sol#686) is not in mixedCase
Variable RakuCoin._StakePoolWalletAddress (RakuCoin.sol#687) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (RakuCoin.sol#14)" inContext (RakuCoin.sol#8-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Reentrancy in RakuCoin._transfer(address,address,uint256) (RakuCoin.sol#1065-1176):
        External calls:
        - sendETHToMarketingPool(address(this).balance) (RakuCoin.sol#1141)
                - _MarketingPoolWalletAddress.transfer(amount) (RakuCoin.sol#1197)
        State variables written after the call(s):
        - removeAllFee() (RakuCoin.sol#1155)
                - _MarketingPoolFee = 0 (RakuCoin.sol#1043)
        - _MarketingPoolFee = _buyMarketingPoolFee (RakuCoin.sol#1158)
        - removeAllFee() (RakuCoin.sol#1163)
                - _MarketingPoolFee = 0 (RakuCoin.sol#1043)
        - _MarketingPoolFee = _sellMarketingPoolFee (RakuCoin.sol#1166)
        - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
                - _MarketingPoolFee = _previousMarketingPoolFee (RakuCoin.sol#1049)
                - _MarketingPoolFee = 0 (RakuCoin.sol#1043)
        - removeAllFee() (RakuCoin.sol#1155)
                - _previousMarketingPoolFee = _MarketingPoolFee (RakuCoin.sol#1039)
        - removeAllFee() (RakuCoin.sol#1163)
```

```
                - _previousMarketingPoolFee = _MarketingPoolFee (RakuCoin.sol#1039)
        - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
                - _previousMarketingPoolFee = _MarketingPoolFee (RakuCoin.sol#1039)
        - removeAllFee() (RakuCoin.sol#1155)
                - _previousStakeTaxFee = _staketaxFee (RakuCoin.sol#1040)
        - removeAllFee() (RakuCoin.sol#1163)
                - _previousStakeTaxFee = _staketaxFee (RakuCoin.sol#1040)
        - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
                - _previousStakeTaxFee = _staketaxFee (RakuCoin.sol#1040)
        - removeAllFee() (RakuCoin.sol#1155)
                - _previousTaxFee = _taxFee (RakuCoin.sol#1038)
        - removeAllFee() (RakuCoin.sol#1163)
                - _previousTaxFee = _taxFee (RakuCoin.sol#1038)
        - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
                - _previousTaxFee = _taxFee (RakuCoin.sol#1038)
        - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
                - _rOwned[address(_StakePoolWalletAddress)] =
    _rOwned[address(_StakePoolWalletAddress)].add(rStakePool) (RakuCoin.sol#1293)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rMarketingPool) (RakuCoin.sol#1285)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (RakuCoin.sol#1249)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (RakuCoin.sol#1239)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (RakuCoin.sol#1273)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (RakuCoin.sol#1240)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (RakuCoin.sol#1262)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (RakuCoin.sol#1263)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (RakuCoin.sol#1251)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (RakuCoin.sol#1275)
        - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
                - _rTotal = _rTotal.sub(rFee) (RakuCoin.sol#1299)
        - removeAllFee() (RakuCoin.sol#1155)
                - _staketaxFee = 0 (RakuCoin.sol#1044)
        - _staketaxFee = _buystaketaxFee (RakuCoin.sol#1157)
        - removeAllFee() (RakuCoin.sol#1163)
                - _staketaxFee = 0 (RakuCoin.sol#1044)
        - _staketaxFee = _sellstaketaxFee (RakuCoin.sol#1165)
        - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
                - _staketaxFee = _previousStakeTaxFee (RakuCoin.sol#1050)
                - _staketaxFee = 0 (RakuCoin.sol#1044)
        - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
                - _tFeeTotal = _tFeeTotal.add(tFee) (RakuCoin.sol#1300)
        - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
                - _tOwned[address(_StakePoolWalletAddress)] =
    _tOwned[address(_StakePoolWalletAddress)].add(tStakePool) (RakuCoin.sol#1295)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tMarketingPool) (RakuCoin.sol#1287)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (RakuCoin.sol#1272)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (RakuCoin.sol#1261)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (RakuCoin.sol#1250)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (RakuCoin.sol#1274)
        - removeAllFee() (RakuCoin.sol#1155)
                - _taxFee = 0 (RakuCoin.sol#1042)
        - _taxFee = _buytaxFee (RakuCoin.sol#1156)
        - removeAllFee() (RakuCoin.sol#1163)
                - _taxFee = 0 (RakuCoin.sol#1042)
        - _taxFee = _selltaxFee (RakuCoin.sol#1164)
        - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
                - _taxFee = _previousTaxFee (RakuCoin.sol#1048)
                - _taxFee = 0 (RakuCoin.sol#1042)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (RakuCoin.sol#1244)
                - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
        - Transfer(sender,recipient,tTransferAmount) (RakuCoin.sol#1267)
                - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
        - Transfer(sender,recipient,tTransferAmount) (RakuCoin.sol#1255)
                - _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
        - Transfer(sender,recipient,tTransferAmount) (RakuCoin.sol#1279)
```

```
		- _tokenTransfer(sender,recipient,amount,takeFee) (RakuCoin.sol#1173)
Reentrancy in RakuCoin.transferFrom(address,address,uint256) (RakuCoin.sol#939-943):
	External calls:
	- _transfer(sender,recipient,amount) (RakuCoin.sol#940)
		- _MarketingPoolWalletAddress.transfer(amount) (RakuCoin.sol#1197)
	State variables written after the call(s):
	- _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer
amount exceeds allowance)) (RakuCoin.sol#941)
		- _allowances[owner][spender] = amount (RakuCoin.sol#1061)
	Event emitted after the call(s):
	- Approval(owner,spender,amount) (RakuCoin.sol#1062)
		- _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20:
transfer amount exceeds allowance)) (RakuCoin.sol#941)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amoun
tADesired (RakuCoin.sol#516) is too similar to
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amoun
tBDesired (RakuCoin.sol#517)
Variable RakuCoin._takeMarketingPool(uint256).rMarketingPool (RakuCoin.sol#1284) is too similar to
RakuCoin._transferFromExcluded(address,address,uint256).tMarketingPool (RakuCoin.sol#1259)
Variable RakuCoin._takeMarketingPool(uint256).rMarketingPool (RakuCoin.sol#1284) is too similar to
RakuCoin._transferStandard(address,address,uint256).tMarketingPool (RakuCoin.sol#1238)
Variable RakuCoin._takeMarketingPool(uint256).rMarketingPool (RakuCoin.sol#1284) is too similar to
RakuCoin._takeMarketingPool(uint256).tMarketingPool (RakuCoin.sol#1282)
Variable RakuCoin._takeMarketingPool(uint256).rMarketingPool (RakuCoin.sol#1284) is too similar to
RakuCoin._transferBothExcluded(address,address,uint256).tMarketingPool (RakuCoin.sol#1271)
Variable RakuCoin._takeMarketingPool(uint256).rMarketingPool (RakuCoin.sol#1284) is too similar to
RakuCoin._transferToExcluded(address,address,uint256).tMarketingPool (RakuCoin.sol#1248)
Variable RakuCoin._takeMarketingPool(uint256).rMarketingPool (RakuCoin.sol#1284) is too similar to
RakuCoin._getTValues(uint256,uint256,uint256,uint256).tMarketingPool (RakuCoin.sol#1327)
Variable RakuCoin._takeMarketingPool(uint256).rMarketingPool (RakuCoin.sol#1284) is too similar to
RakuCoin._getValues(uint256).tMarketingPool (RakuCoin.sol#1307)
Variable RakuCoin._getRValues(uint256,uint256,uint256).rTransferAmount (RakuCoin.sol#1343) is too
similar to RakuCoin._transferToExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1248)
Variable RakuCoin._transferFromExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1259)
is too similar to RakuCoin._transferFromExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1259)
Variable RakuCoin._getValues2(uint256,uint256).rTransferAmount (RakuCoin.sol#1318) is too similar to
RakuCoin._addonvalues(uint256,uint256,uint256,uint256).tTransferAmount (RakuCoin.sol#1335)
Variable RakuCoin._getRValues(uint256,uint256,uint256).rTransferAmount (RakuCoin.sol#1343) is too
similar to RakuCoin._transferFromExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1259)
Variable RakuCoin._transferFromExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1259)
is too similar to RakuCoin._transferToExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1248)
Variable RakuCoin._getRValues(uint256,uint256,uint256).rTransferAmount (RakuCoin.sol#1343) is too
similar to RakuCoin._transferBothExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1271)
Variable RakuCoin._getValues2(uint256,uint256).rTransferAmount (RakuCoin.sol#1318) is too similar to
RakuCoin._transferToExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1248)
Variable RakuCoin._getValues(uint256).rTransferAmount (RakuCoin.sol#1309) is too similar to
RakuCoin._transferFromExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1259)
Variable RakuCoin._getRValues(uint256,uint256,uint256).rTransferAmount (RakuCoin.sol#1343) is too
similar to RakuCoin._getValues(uint256).tTransferAmount (RakuCoin.sol#1307)
Variable RakuCoin._getRValues(uint256,uint256,uint256).rTransferAmount (RakuCoin.sol#1343) is too
similar to RakuCoin._transferStandard(address,address,uint256).tTransferAmount (RakuCoin.sol#1238)
Variable RakuCoin._transferToExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1248) is
too similar to RakuCoin._transferToExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1248)
Variable RakuCoin._getValues(uint256).rTransferAmount (RakuCoin.sol#1309) is too similar to
RakuCoin._transferStandard(address,address,uint256).tTransferAmount (RakuCoin.sol#1238)
Variable RakuCoin._getValues(uint256).rTransferAmount (RakuCoin.sol#1309) is too similar to
RakuCoin._transferBothExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1271)
```

```
Variable RakuCoin._transferBothExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1271)
is too similar to RakuCoin._transferBothExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1271)
Variable RakuCoin._getValues(uint256).rTransferAmount (RakuCoin.sol#1309) is too similar to
RakuCoin._transferToExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1248)
Variable RakuCoin._transferBothExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1271)
is too similar to RakuCoin._transferToExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1248)
Variable RakuCoin._transferFromExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1259)
is too similar to RakuCoin._getValues(uint256).tTransferAmount (RakuCoin.sol#1307)
Variable RakuCoin._transferFromExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1259)
is too similar to RakuCoin._transferBothExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1271)
Variable RakuCoin._getValues(uint256).rTransferAmount (RakuCoin.sol#1309) is too similar to
RakuCoin._addonvalues(uint256,uint256,uint256,uint256).tTransferAmount (RakuCoin.sol#1335)
Variable RakuCoin._transferBothExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1271)
is too similar to RakuCoin._addonvalues(uint256,uint256,uint256,uint256).tTransferAmount
(RakuCoin.sol#1335)
Variable RakuCoin._getValues2(uint256,uint256).rTransferAmount (RakuCoin.sol#1318) is too similar to
RakuCoin._transferBothExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1271)
Variable RakuCoin._transferFromExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1259)
is too similar to RakuCoin._transferStandard(address,address,uint256).tTransferAmount
(RakuCoin.sol#1238)
Variable RakuCoin._getValues(uint256).rTransferAmount (RakuCoin.sol#1309) is too similar to
RakuCoin._getValues(uint256).tTransferAmount (RakuCoin.sol#1307)
Variable RakuCoin._getRValues(uint256,uint256,uint256).rTransferAmount (RakuCoin.sol#1343) is too
similar to RakuCoin._addonvalues(uint256,uint256,uint256,uint256).tTransferAmount (RakuCoin.sol#1335)
Variable RakuCoin._transferFromExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1259)
is too similar to RakuCoin._addonvalues(uint256,uint256,uint256,uint256).tTransferAmount
(RakuCoin.sol#1335)
Variable RakuCoin._transferToExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1248) is
too similar to RakuCoin._transferStandard(address,address,uint256).tTransferAmount (RakuCoin.sol#1238)
Variable RakuCoin._transferToExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1248) is
too similar to RakuCoin._transferFromExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1259)
Variable RakuCoin._getValues2(uint256,uint256).rTransferAmount (RakuCoin.sol#1318) is too similar to
RakuCoin._getValues(uint256).tTransferAmount (RakuCoin.sol#1307)
Variable RakuCoin._transferStandard(address,address,uint256).rTransferAmount (RakuCoin.sol#1238) is
too similar to RakuCoin._getValues(uint256).tTransferAmount (RakuCoin.sol#1307)
Variable RakuCoin._transferBothExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1271)
is too similar to RakuCoin._getValues(uint256).tTransferAmount (RakuCoin.sol#1307)
Variable RakuCoin._transferBothExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1271)
is too similar to RakuCoin._transferFromExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1259)
Variable RakuCoin._transferStandard(address,address,uint256).rTransferAmount (RakuCoin.sol#1238) is
too similar to RakuCoin._transferFromExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1259)
Variable RakuCoin.reflectionFromToken(uint256,bool).rTransferAmount (RakuCoin.sol#1001) is too similar to
RakuCoin._transferToExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1248)
Variable RakuCoin._getValues2(uint256,uint256).rTransferAmount (RakuCoin.sol#1318) is too similar to
RakuCoin._transferFromExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1259)
Variable RakuCoin._transferToExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1248) is
too similar to RakuCoin._getValues(uint256).tTransferAmount (RakuCoin.sol#1307)
Variable RakuCoin._getValues2(uint256,uint256).rTransferAmount (RakuCoin.sol#1318) is too similar to
RakuCoin._transferStandard(address,address,uint256).tTransferAmount (RakuCoin.sol#1238)
Variable RakuCoin.reflectionFromToken(uint256,bool).rTransferAmount (RakuCoin.sol#1001) is too similar to
RakuCoin._addonvalues(uint256,uint256,uint256,uint256).tTransferAmount (RakuCoin.sol#1335)
Variable RakuCoin.reflectionFromToken(uint256,bool).rTransferAmount (RakuCoin.sol#1001) is too similar to
RakuCoin._transferFromExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1259)
Variable RakuCoin._transferStandard(address,address,uint256).rTransferAmount (RakuCoin.sol#1238) is
too similar to RakuCoin._transferToExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1248)
Variable RakuCoin.reflectionFromToken(uint256,bool).rTransferAmount (RakuCoin.sol#1001) is too similar to
RakuCoin._getValues(uint256).tTransferAmount (RakuCoin.sol#1307)
```

```
Variable RakuCoin.reflectionFromToken(uint256,bool).rTransferAmount (RakuCoin.sol#1001) is too similar to
RakuCoin._transferStandard(address,address,uint256).tTransferAmount (RakuCoin.sol#1238)
Variable RakuCoin._transferStandard(address,address,uint256).rTransferAmount (RakuCoin.sol#1238) is
too similar to RakuCoin._addonvalues(uint256,uint256,uint256,uint256).tTransferAmount
(RakuCoin.sol#1335)
Variable RakuCoin.reflectionFromToken(uint256,bool).rTransferAmount (RakuCoin.sol#1001) is too similar to
RakuCoin._transferBothExcluded(address,address,uint256).tTransferAmount (RakuCoin.sol#1271)
Variable RakuCoin._transferBothExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1271)
is too similar to RakuCoin._transferStandard(address,address,uint256).tTransferAmount
(RakuCoin.sol#1238)
Variable RakuCoin._transferToExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1248) is
too similar to RakuCoin._transferBothExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1271)
Variable RakuCoin._transferStandard(address,address,uint256).rTransferAmount (RakuCoin.sol#1238) is
too similar to RakuCoin._transferStandard(address,address,uint256).tTransferAmount (RakuCoin.sol#1238)
Variable RakuCoin._transferStandard(address,address,uint256).rTransferAmount (RakuCoin.sol#1238) is
too similar to RakuCoin._transferBothExcluded(address,address,uint256).tTransferAmount
(RakuCoin.sol#1271)
Variable RakuCoin._transferToExcluded(address,address,uint256).rTransferAmount (RakuCoin.sol#1248) is
too similar to RakuCoin._addonvalues(uint256,uint256,uint256,uint256).tTransferAmount
(RakuCoin.sol#1335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _isSniper[address(0x000000000000084e91743124a982076C59f10084)] = true (RakuCoin.sol#778)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x000000000000084e91743124a982076C59f10084))
(RakuCoin.sol#779)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _isSniper[address(0x000000005804B22091aa9830E50459A15E7C9241)] = true (RakuCoin.sol#787)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x000000005804B22091aa9830E50459A15E7C9241))
(RakuCoin.sol#788)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _isSniper[address(0x00000000000007673393729D5618DC555FD13f9aA)] = true (RakuCoin.sol#796)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x00000000000007673393729D5618DC555FD13f9aA))
(RakuCoin.sol#797)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _isSniper[address(0x00000000000003441d59DdE9A90BFfb1CD3fABf1)] = true (RakuCoin.sol#799)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x00000000000003441d59DdE9A90BFfb1CD3fABf1))
(RakuCoin.sol#800)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _isSniper[address(0x000000917de6037d52b1F0a306eeCD208405f7cd)] = true (RakuCoin.sol#805)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x000000917de6037d52b1F0a306eeCD208405f7cd))
(RakuCoin.sol#806)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _isSniper[address(0x00000000003b3cc22aF3aE1EAc0440BcEe416B40)] = true (RakuCoin.sol#853)
RakuCoin.initContract() (RakuCoin.sol#733-877) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x00000000003b3cc22aF3aE1EAc0440BcEe416B40))
(RakuCoin.sol#854)
RakuCoin.slitherConstructorVariables() (RakuCoin.sol#645-1434) uses literals with too many digits:
        - _tTotal = 1000000 * 10 ** 18 (RakuCoin.sol#662)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
RakuCoin._decimals (RakuCoin.sol#668) should be constant
RakuCoin._name (RakuCoin.sol#666) should be constant
RakuCoin._numOfTokensToExchangeForMarketingPool (RakuCoin.sol#701) should be constant
RakuCoin._symbol (RakuCoin.sol#667) should be constant
RakuCoin._tTotal (RakuCoin.sol#662) should be constant
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
```

```
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (RakuCoin.sol#406-409)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (RakuCoin.sol#415-419)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (RakuCoin.sol#421-423)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (RakuCoin.sol#426-431)
unlock() should be declared external:
        - Ownable.unlock() (RakuCoin.sol#434-439)
isBlackListed(address) should be declared external:
        - RakuCoin.isBlackListed(address) (RakuCoin.sol#900-902)
name() should be declared external:
        - RakuCoin.name() (RakuCoin.sol#904-906)
symbol() should be declared external:
        - RakuCoin.symbol() (RakuCoin.sol#908-910)
decimals() should be declared external:
        - RakuCoin.decimals() (RakuCoin.sol#912-914)
totalSupply() should be declared external:
        - RakuCoin.totalSupply() (RakuCoin.sol#916-918)
transfer(address,uint256) should be declared external:
        - RakuCoin.transfer(address,uint256) (RakuCoin.sol#925-928)
allowance(address,address) should be declared external:
        - RakuCoin.allowance(address,address) (RakuCoin.sol#930-932)
approve(address,uint256) should be declared external:
        - RakuCoin.approve(address,uint256) (RakuCoin.sol#934-937)
transferFrom(address,address,uint256) should be declared external:
        - RakuCoin.transferFrom(address,address,uint256) (RakuCoin.sol#939-943)
increaseAllowance(address,uint256) should be declared external:
        - RakuCoin.increaseAllowance(address,uint256) (RakuCoin.sol#945-948)
decreaseAllowance(address,uint256) should be declared external:
        - RakuCoin.decreaseAllowance(address,uint256) (RakuCoin.sol#950-953)
isExcluded(address) should be declared external:
        - RakuCoin.isExcluded(address) (RakuCoin.sol#955-957)
totalFees() should be declared external:
        - RakuCoin.totalFees() (RakuCoin.sol#963-965)
deliver(uint256) should be declared external:
        - RakuCoin.deliver(uint256) (RakuCoin.sol#986-993)
reflectionFromToken(uint256,bool) should be declared external:
        - RakuCoin.reflectionFromToken(uint256,bool) (RakuCoin.sol#995-1004)
isExcludedFromFee(address) should be declared external:
        - RakuCoin.isExcludedFromFee(address) (RakuCoin.sol#1053-1055)
_getETHBalance() should be declared external:
        - RakuCoin._getETHBalance() (RakuCoin.sol#1372-1374)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:RakuCoin.sol analyzed (10 contracts with 75 detectors), 169 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```