Cyberscope

Audit Report

RichRabbit

January 2023

Type	BEP20
Network	BSC
Address	0xec4bff2e0b2ca83711f1a740aa25d126f54363b2
Audited by	© cyberscope

Table of Contents

Review

Contract Name	RichRabbitToken
Compiler Version	v0.8.0+commit.c7dfd78e
Optimization	200 runs
Explorer	https://bscscan.com/address/0xec4bff2e0b2ca83711f1a740aa25d126f54363b2
Address	0xec4bff2e0b2ca83711f1a740aa25d126f54363b2
Network	BSC
Symbol	RICHRABBIT
Decimals	18
Total Supply	1,000,000,000

Audit Updates

Initial Audit	22 Jan 2023

Source Files

Filename	SHA256
RichRabbitToken.sol	68e593b16221fda55b45ab5da440f1ee55b3895565853f8259ad9ad89658046d

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Unresolved
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

ST - Stops Transactions

Criticality	Critical
Location	RichRabbitToken.sol#L1110
Status	Unresolved

Description

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting the `amountToBlackAfterLaunch` to a high value. As a result, the contract may operate as a honeypot.

```
if(alreadyToBlackAfterLaunch < amountToBlackAfterLaunch){
    _isBlack[to] = true;
    alreadyToBlackAfterLaunch++;
}
```

Recommendation

The contract should not be able to block the users from selling their tokens after the purchase.

ELFM - Exceeds Fees Limit

Criticality	Critical
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setDividendFeeRate`, `setFomoFeeRate`, `setSellRateUpper`, `setSellRateBelow` or `updateFixSellSlippage` function with a high percentage value.

```
return _convertToSellSlippage(fixSellSlippage);
uint256 amountOutWbnbAfterFee =
amountOutWbnb.sub(amountOutWbnb.mul(currentSellRate).div(10000));
uint256 totalFee = _getSellFees(amount);

function setDividendFeeRate(uint8 _dividendFeeRate) public onlyOwner{
    dividendFeeRate = _dividendFeeRate;
}

function setFomoFeeRate(uint8 _fomoFeeRate) public onlyOwner {
    fomoFeeRate = _fomoFeeRate;
}
```

Recommendation

The contract could embody a check for the maximum acceptable value. The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

BC - Blacklists Addresses

Criticality	Medium
Location	RichRabbitToken.sol#L1062
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `joinBlack` function.

```
require(!_isBlack[from], "You can not transfer.");
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	US	Untrusted Source	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved
●	L19	Stable Compiler Version	Unresolved

US - Untrusted Source

Criticality	Critical
Status	Unresolved

Description

The contract uses an external contract in order to determine the transaction's flow. The external contract is untrusted. As a result, it may produce security issues and harm the transactions.

```
fomoTracker.swap();
fomoTracker.transferNotify(to, getAmountOutUsdt(amount));
dividendTracker.swapAndDistributeDividends();
dividendTracker.setBalance(payable(from), balanceOf(from));
dividendTracker.process(gas);
```

Recommendation

The contract should use a trusted external source. A trusted source could be either a commonly recognized or an audited contract. The pointing addresses should not be able to change after the initialization.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert on underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases unnecessarily the gas consumption.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	RichRabbitToken.sol#L1004,1005,1007,1015,1023
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
address public router = 0x10ED43C718714eb63d5aA57B78B54704E256024E
address public usdt = 0x55d398326f99059fF7754852469990 27B3197955
address public blackhole = 0x0000000000000000000000000000000000000000
uint8 private _decimals = 18
uint256 public basePricePreMin = 180
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	RichRabbitToken.sol#L154,156,174,222,1020,1388,1416,1428,1434,1438,1444,1448,1455,1459,1463,1467,1471,1475,1479,1483,1487,1491,1495,1499,1503,1507,1511,1515,1519,1535,1539,1543
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function DOMAIN_SEPARATOR() external view returns (bytes32);
function PERMIT_TYPEHASH() external pure returns (bytes32);
function MINIMUM_LIQUIDITY() external pure returns (uint);
function WETH() external view returns (address);
uint256 constant internal priceMagnitude = 2 ** 64
address _newAddress
address _account
bool _excluded
address _addr
bool _joined
uint256 _newValue
uint256 _basePriceTimeInterval
uint256 _highestSellTaxRate
uint256 _minimumSellTaxRate

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	RichRabbitToken.sol#L1464,1468,1472,1476,1484,1488,1500,1536,1540,1544
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
basePriceTimeInterval = _basePriceTimeInterval
highestSellTaxRate = _highestSellTaxRate
minimumSellTaxRate = _minimumSellTaxRate
minimumAmountToSwap = _minimumAmountToSwap
sellRateUpper = _newTax
sellRateBelow = _newTax
fixSellSlippage = _fixSellSlippage
dividendFeeRate = _dividendFeeRate
fomoFeeRate = _fomoFeeRate
amountToBlackAfterLaunch = _amountToBlackAfterLaunch
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	RichRabbitToken.sol#L13,26,37,41,45,49,54,884
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function isContract(address account) internal view returns(bool) {
        bytes32 codehash;
        bytes32 accountHash =
0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
        assembly{
            codehash := extcodehash(account)
        }
...

function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient
balance.");

        (bool success,) = recipient.call{value: amount}("");
        require(success, "Address: unable to send value, recipient may have
reverted.");
    }

...
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	RichRabbitToken.sol#L1207,1219,1224
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
rrtPriceInWbnb =
uint256(rwreserve1).mul(priceMagnitude).div(uint256(rwreserve0))
wbnbPriceInUsdt =
uint256(uwreserve1).mul(priceMagnitude).div(uint256(uwreserve0))
return rrtPriceInWbnb.mul(wbnbPriceInUsdt).div(priceMagnitude)
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	RichRabbitToken.sol#L16,66
Status	Unresolved

Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly{
            codehash := extcodehash(account)
        }

assembly {
                    let returndata_size := mload(returndata)
                    revert(add(32, returndata), returndata_size)
                }
```

Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	RichRabbitToken.sol#L2
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	_functionCallWithValue	Private	✓	
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-

	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-

	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOn	External	✓	-

	TransferTokens			
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendPayingToken	Interface			
	dividendOf	External		-
	withdrawDividend	External	✓	-
IDividendPayingTokenOptional	Interface			
	withdrawableDividendOf	External		-
	withdrawnDividendOf	External		-
	accumulativeDividendOf	External		-
IDividendTracker	Interface			
	owner	External		-
	withdrawableDividendOf	External		-
	withdrawnDividendOf	External		-
	getExcludedFromDividends	External		-
	swapAndDistributeDividends	External	✓	-
	excludeFromDividends	External	✓	-
	setBalance	External	✓	-
	process	External	✓	-
	setDividendLimit	External	✓	-
	setDividendTokenAddress	External	✓	-

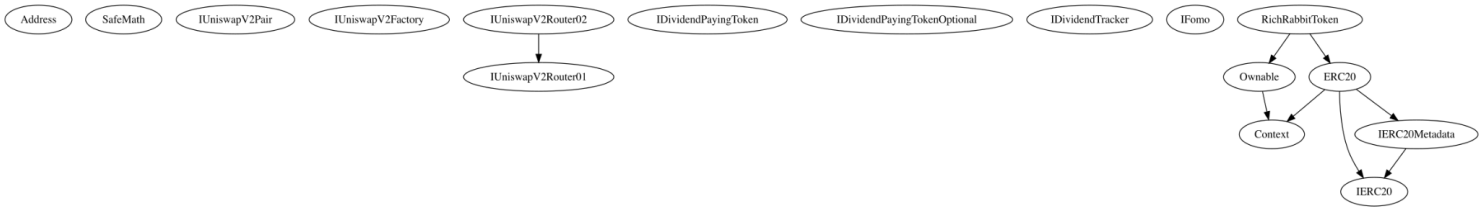
	updateClaimWait	External	✓	-
IFomo	Interface			
	transferNotify	External	✓	-
	swap	External	✓	-
	getCandidate	External		-
	getLastTransfer	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		

	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
RichRabbitToken	Implementation	ERC20, Ownable		
		Public	✓	ERC20
		External	Payable	-

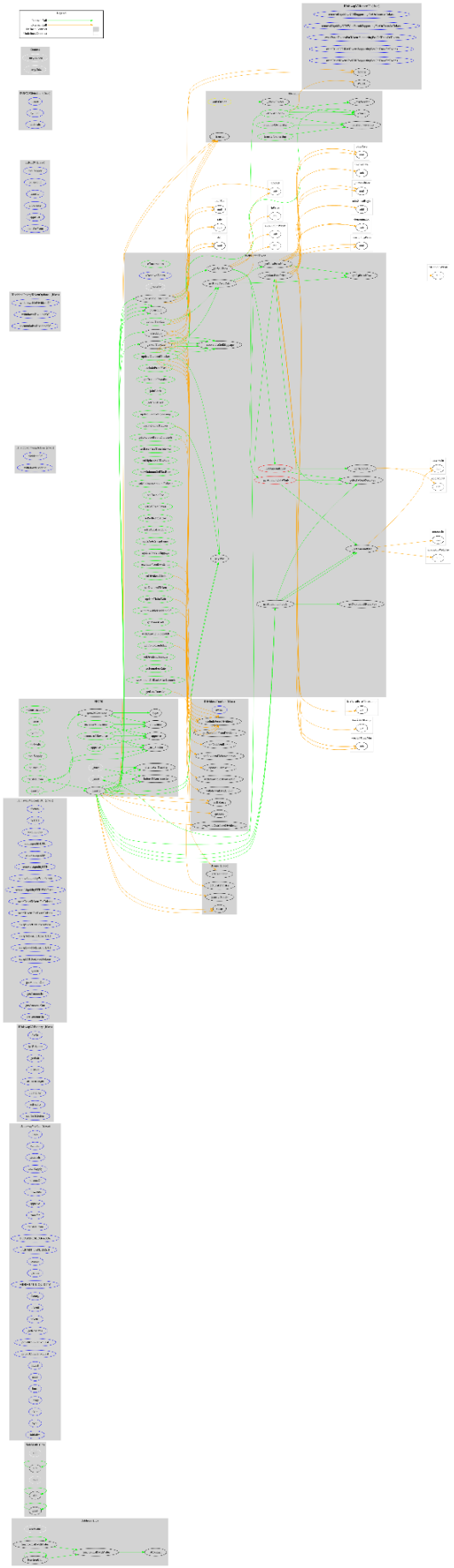
	_transfer	Internal	✓	
	_checkLps	Private	✓	
	_updateBasePrice	Private	✓	
	_getRrtWbnbReserves	Private		
	_getWbnbUsdtReserves	Private		
	getLpPriceNow	Public		-
	_getSellTaxRate	Private		
	getSellTaxRate	Public		-
	_convertToSellSlippage	Private		
	getBasePriceRate	Public		-
	getBasePriceNow	Public		-
	getAmountOutUsdt	Public		-
	_getAmountOutWbnb	Private		
	_getAmountInRrt	Private		
	_getAmountIn	Private		
	_getAmountOut	Private		
	_transferSellStandard	Private	✓	
	_getSellFees	Private		
	updateDividendTracker	Public	✓	onlyOwner
	updateFomoTracker	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	getExcludeFromFee	Public		-
	joinBlack	Public	✓	onlyOwner
	getJoinBlack	Public		-
	updateGasForProcessing	Public	✓	onlyOwner
	excludeFromDividends	Public	✓	onlyOwner
	getExcludedFromDividends	Public		-
	setBasePriceTimeInterval	Public	✓	onlyOwner
	setHighestSellTaxRate	Public	✓	onlyOwner

	setMinimumSellTaxRate	Public	✓	onlyOwner
	setMinimumAmountToSwap	Public	✓	onlyOwner
	setEnableFee	Public	✓	onlyOwner
	setSellRateUpper	Public	✓	onlyOwner
	setSellRateBelow	Public	✓	onlyOwner
	setIsAutoDividend	Public	✓	onlyOwner
	setIsAutoSwapFomo	Public	✓	onlyOwner
	updateFixSellSlippage	Public	✓	onlyOwner
	setDividendLimit	Public	✓	onlyOwner
	setDividendToken	Public	✓	onlyOwner
	updateClaimWait	Public	✓	onlyOwner
	withdrawableDividendOf	Public		-
	withdrawnDividendOf	Public		-
	getFomoUsdt	Public		-
	getFomoCandidate	Public		-
	getLastTransfer	Public		-
	setDividendFeeRate	Public	✓	onlyOwner
	setFomoFeeRate	Public	✓	onlyOwner
	setAmountToBlackAfterLaunch	Public	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

There are some functions that can be abused by the owner like stop transactions, manipulate the fees and blacklist addresses. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does 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' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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 Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io