Cyberscope

Audit Report

Reefer Token

March 2022

Type	BEP20
Network	BSC
Address	0x701b57da9EFF1D3F1Ce4E90171F602ff16fc05a4
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	REEFER
Compiler Version	v0.8.6+commit.11564f7e
Optimization	200 runs
Licence	None
Explorer	https://bscscan.com/token/0x701b57da9EFF1D3F1Ce4E90171F602ff16fc05a4
Symbol	REEFER
Decimals	18
Total Supply	100,000,000,000
Domain	reefertoken.io

Audit Updates

Initial Audit	7th March 2022
Corrected	

Contract Analysis

Critical Medium Minor Pass

Severity	Code	Description
●	ST	Contract Owner is not able to stop or pause transactions
●	OCTD	Contract Owner is not able to transfer tokens from specific address
●	OTUT	Owner Transfer User's Tokens
●	ELFM	Contract Owner is not able to increase fees more than a reasonable percent (25%)
●	ULTW	Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent
●	MT	Contract Owner is not able to mint new tokens
●	BT	Contract Owner is not able to burn tokens from specific wallet
●	BC	Contract Owner is not able to blacklist wallets from selling

OCTD - Owner Contract Tokens Drain

Criticality	minor
Location	contract.sol#L243,248

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling any of the `rescue` functions.

```
    function rescueBEP20Tokens(address tokenAddress) external onlyOwner{
        IERC20(tokenAddress).transfer(msg.sender,
IERC20(tokenAddress).balanceOf(address(this)));
    }

    function rescueBNB() external {
        uint256 BNBbalance = address(this).balance;
        payable(owner()).sendValue(BNBbalance);
    }
```

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.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L323,327,331,335

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling any of the set fees functions.

```
    function setMarketingFee(uint256 newFee) external onlyOwner{
        marketingFee = newFee;
        totalFees = BNBRewardsFee + marketingFee + liquidityFee + teamFee +
devFee;
    }
```

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.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	L01	Public Function could be Declared External
	L02	State Variables could be Declared Constant
	L04	Conformance to Solidity Naming Conventions
	L12	Using Variables before Declaration
	L07	Missing Events Arithmetic
	L14	Uninitialized Variables in Local Scope
	L13	Divide before Multiply Operation

L01 - Public Function could be Declared External

Criticality	minor
Location	Reefer.sol#L177,194,207,220,253,272,276,280,604,642

Description

Public functions that are never called by the contract should be declared external to save gas.

```
process
getAccountAtIndex
dividendTokenBalanceOf
withdrawableDividendOf
isExcludedFromFees
updateGasForProcessing
setAutomatedMarketMakerPair
excludeMultipleAccountsFromFees
updateRouter
...
```

Recommendation

Use the external attribute for functions never called from the contract.

L02 - State Variables could be Declared Constant

Criticality	minor
Location	Reefer.sol#L43,49

Description

Constant state variables should be declared constant to save gas.

```
devWallet
devFee
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	Reefer.sol#L147,151,41,559

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_account
BNBRewardsFee
_minInWei
_maxInWei
_rate
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

L12 - Using Variables before Declaration

Criticality	minor
Location	Reefer.sol#L408

Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or the variable has been declared in a different scope.

```
lastProcessedIndex
iterations
claims
```

Recommendation

The variables should be declared before any usage of them.

L07 - Missing Events Arithmetic

Criticality	minor
Location	Reefer.sol#L147,215,327,332,337,342

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
teamFee = newFee
liquidityFee = newFee
BNBRewardsFee = newFee
marketingFee = newFee
swapTokensAtAmount = amount * 10 ** 18
presaleRate = _rate
```

Recommendation

Emit an event for critical parameter changes.

L14 - Uninitialized Variables in Local Scope

Criticality	minor
Location	Reefer.sol#L408

Description

The are variables that are defined in the local scope and are not initialized.

```
claims
iterations
lastProcessedIndex
```

Recommendation

All the local scoped variables should be initialized.

L13 - Divide before Multiply Operation

Criticality	minor
Location	Reefer.sol#L417

Description

Performing divisions before multiplications may cause lose of prediction.

```
unitBalance = deltaBalance / (denominator - liquidityFee)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
DividendPayingToken	Implementation	ERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
	<Constructor>	Public	✓	ERC20
	<Receive Ether>	External	Payable	-
	distributeDividends	Public	Payable	-
	withdrawDividend	Public	✓	-
	_withdrawDividendOfUser	Internal	✓	
	dividendOf	Public		-
	withdrawableDividendOf	Public		-
	withdrawnDividendOf	Public		-
	accumulativeDividendOf	Public		-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_setBalance	Internal	✓	
DividendPayingTokenInterface	Interface			
	dividendOf	External		-
	distributeDividends	External	Payable	-
	withdrawDividend	External	✓	-

DividendPayingTokenOptionalInterface	Interface			
	withdrawableDividendOf	External		-
	withdrawnDividendOf	External		-
	accumulativeDividendOf	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		
	<Constructor>	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	✓	
	_beforeTokenTransfer	Internal	✓	
IPair	Interface			
	sync	External	✓	-
IFactory	Interface			
	createPair	External	✓	-
	getPair	External		-

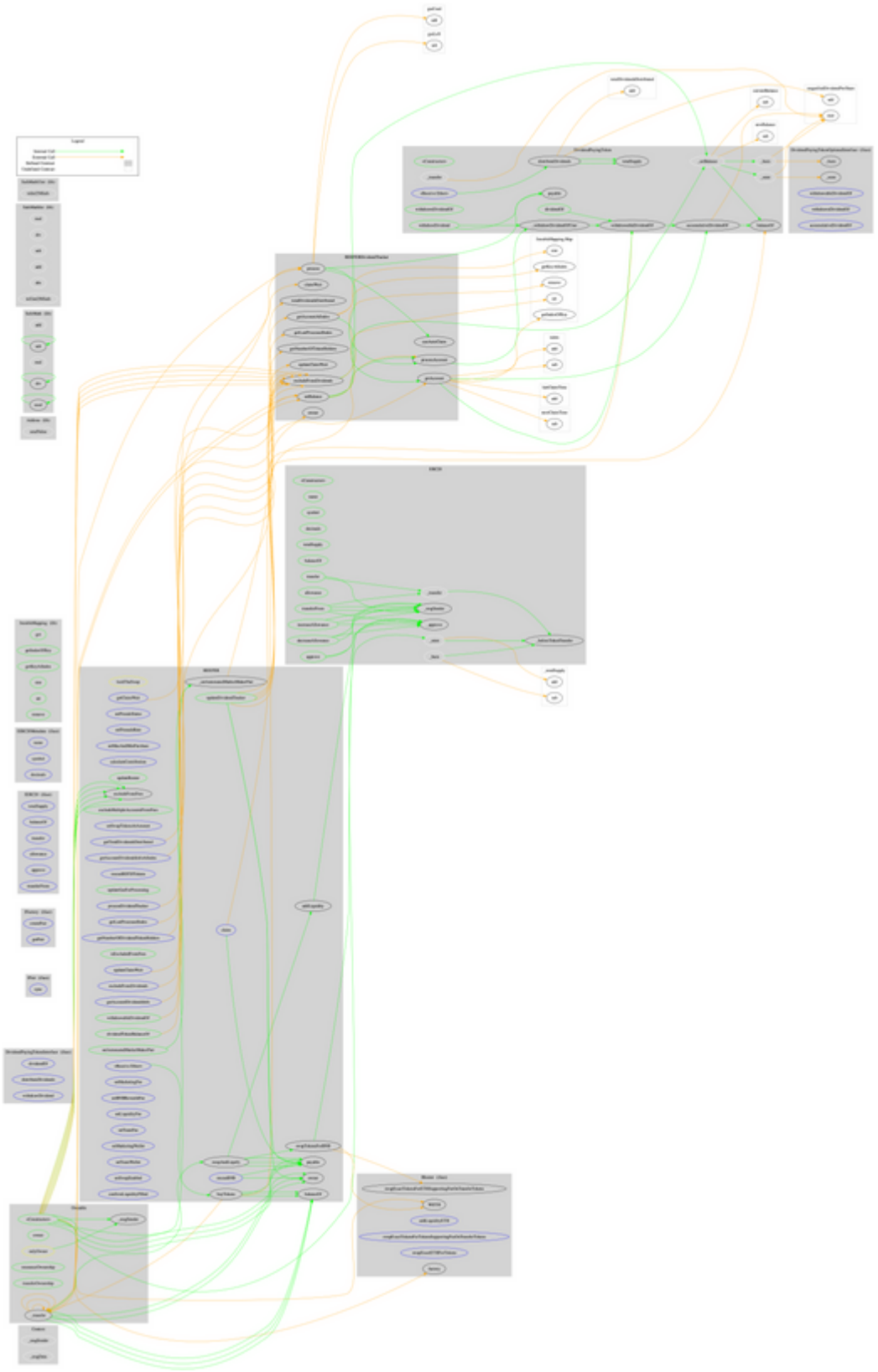
IRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
IterableMapping	Library			
	get	Public		-
	getIndexOfKey	Public		-
	getKeyAtIndex	Public		-
	size	Public		-
	set	Public	✓	-
	remove	Public	✓	-
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner

	transferOwnership	Public	✓	onlyOwner
Address	Library			
	sendValue	Internal	✓	
REEFER	Implementation	ERC20, Ownable		
	<Constructor>	Public	✓	ERC20
	<Receive Ether>	External	Payable	-
	setPresaleStatus	External	✓	onlyOwner
	setPreasaleRate	External	✓	onlyOwner
	setMaxAndMinPurchase	External	✓	onlyOwner
	buyTokens	Public	Payable	-
	calculateContribution	External		-
	updateDividendTracker	Public	✓	onlyOwner
	updateRouter	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	excludeMultipleAccountsFromFees	Public	✓	onlyOwner
	setSwapTokensAtAmount	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	excludeFromDividends	External	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	rescueBEP20Tokens	External	✓	onlyOwner
	rescueBNB	External	✓	-
	updateGasForProcessing	Public	✓	onlyOwner
	updateClaimWait	External	✓	onlyOwner
	getClaimWait	External		-
	getTotalDividendsDistributed	External		-
	isExcludedFromFees	Public		-
	withdrawableDividendOf	Public		-
	dividendTokenBalanceOf	Public		-
	getAccountDividendsInfo	External		-
	getAccountDividendsInfoAtIndex	External		-
	processDividendTracker	External	✓	-
	claim	External	✓	-
	getLastProcessedIndex	External		-

	getNumberOfDividendTokenHolders	External		-
	setMarketingFee	External	✓	onlyOwner
	setBNBRewardsFee	External	✓	onlyOwner
	setLiquidityFee	External	✓	onlyOwner
	setTeamFee	External	✓	onlyOwner
	setMarketingWallet	External	✓	onlyOwner
	setTeamWallet	External	✓	onlyOwner
	setSwapEnabled	External	✓	onlyOwner
	confirmLiquidityFIlled	External	✓	onlyOwner
	_transfer	Internal	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	addLiquidity	Private	✓	
	swapTokensForBNB	Private	✓	
REEFERDividendTracker	Implementation	DividendPayingToken, Ownable		
	<Constructor>	Public	✓	DividendPayingToken
	_transfer	Internal		
	withdrawDividend	Public		-
	excludeFromDividends	External	✓	onlyOwner
	updateClaimWait	External	✓	onlyOwner
	getLastProcessedIndex	External		-
	getNumberOfTokenHolders	External		-
	getAccount	Public		-
	getAccountAtIndex	Public		-
	canAutoClaim	Private		
	setBalance	External	✓	onlyOwner
	process	Public	✓	-
	processAccount	Public	✓	onlyOwner
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		

	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
SafeMathInt	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		
	add	Internal		
	abs	Internal		
	toUint256Safe	Internal		
SafeMathUint	Library			
	toInt256Safe	Internal		

Contract Flow



Domain Info

Domain Name	reefertoken.io
Registry Domain ID	c1fd7cfc85444c2c839c01316fc267e8-DONUTS
Creation Date	2022-01-05T15:42:42Z
Updated Date	2022-01-10T15:43:41Z
Registry Expiry Date	2023-01-05T15:42:42Z
Registrar WHOIS Server	whois.godaddy.com/
Registrar URL	http://www.godaddy.com/domains/search.aspx?ci=8990
Registrar	GoDaddy.com, LLC
Registrar IANA ID	146

The domain has been created 2 months before the creation of the audit. It will expire in 10 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

Reefer Token is an interesting project with a friendly and growing community. There are some functions that can be abused by the owner, like manipulating fees without limits and transferring accumulated fees to the team’s wallet directly. 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

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io