RugFreeCoins Audit

Meta Shiba Inu Token

Smart Contract Security Audit

April 09, 2022

Contents

Audit details

Audited project
Meta Shiba Inu Token

Contract Address
0x6fDfA944478Fd4D87B16902147062CcDC985f2eF

Client contact
Meta Shiba Inu Team

Blockchain
Binance smart chain

Project website
https://metashibainu.info/

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

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

Background

Rugfreecoins was commissioned by the Meta Shiba Inu Team to perform an audit of the smart contract.

https://bscscan.com/token/0x6fDfA944478Fd4D87B16902147062CcDC985f2eF

The focus of this audit is to verify that the smart contract is secure, resilient, and working according to the specifications.

The information in this report should be used to understand the risk exposure of the smart contract, project feasibility, long-term sustainability, and as a guide to improving the security posture of the smart contract by remediating the issues that were identified.

.

About the project

Meta Shiba Inu is a token built on the Binance Smart Chain that is with an innovative investment use case the main purpose of which is to seek out constant revenue sources, which in turn, powers reward combined with the VR entertainment and VR gaming. Each transaction, purchase, and sale incur an 11% fee.

Features

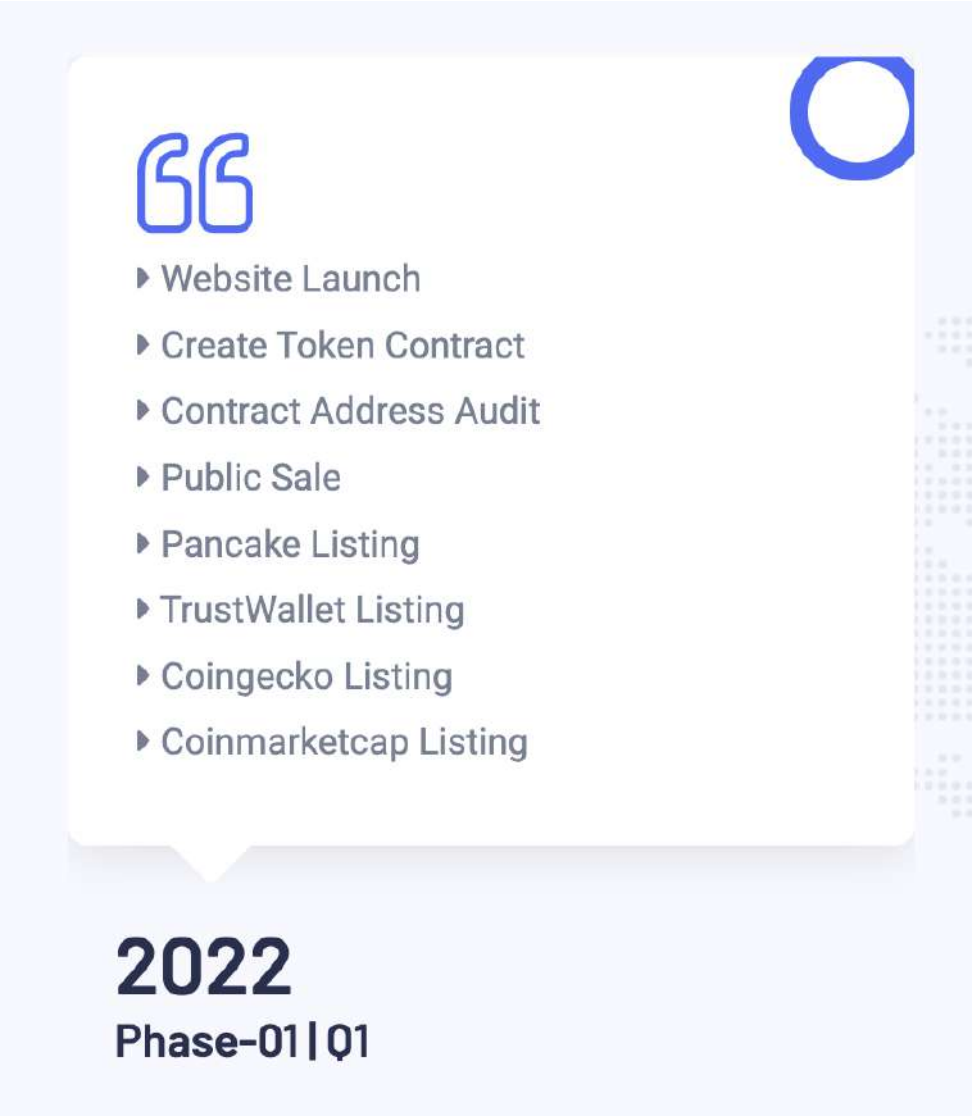
- The **BUSD rewards** will be distributed among every holder proportional to how many tokens each individual holds in values of **3% when buying and selling.**

- The **sustainability fee of 1.5% when buying and selling for marketing and 2% when buying and 3% when selling for dev** is what allows Meta Shiba Inu to hold the aforementioned promise. Tokens will be swapped into BNB and will be sent to a marketing wallet and dev wallet. This way, Meta Shiba Inu will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.

- The additional component included under the sustainability section is a **liquidity fee of 1.5% from buying and selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

- Meta Shiba Inu has the burn strategy that a **1% fee in each transaction when buying selling** is getting charged that benefits and rewards those who invest long-term. This feature slowly reduces supply making each Meta Shiba Inu more and more valuable.

ROADMAP


Website Launch
Create Token Contract
Contract Address Audit
Public Sale
Pancake Listing
TrustWallet Listing
Coingecko Listing
Coinmarketcap Listing
2022
Phase-01 | Q1

- Start developing Metaverse
- CEX Listing
- 3rd party promoter Listing
- Userbase increment generator program launch
- 50,000 Holder Target

2022

Phase-02 | Q2

- 100,000 Holder Target
- Midium Scale marketing launch
- Top CEX Listing
- Deploy demo VR [metaverse] based gaming with implementation of $MSHIB
- Upgrade Website v2
- Develop & deploy $MSHIB powered #metaverse/ VR #Gaming platform
- A vast massive marketing campaign to gain mainstream attention

2022

Phase-03 | Q2-Q4

Tokenomics

9% fee when buying

- 3% of trade goes to holders' pockets in BUSD.
- 1.5% of trade goes to the marketing wallet in BNB.
- 2% of trade goes to the dev wallet in BNB.
- 2% of trade goes to the burn.
- 1.5% of trade goes to the liquidity pool.

10% fee when selling

- 3% of trade goes to holders' pockets in BUSD.
- 1.5% of trade goes to the marketing wallet in BNB.
- 3% of trade goes to the dev wallet in BNB.
- 2% of trade goes to the burn.
- 1.5% of trade goes to the liquidity pool.

Target market and the concept

Target market

- Anyone who's interested in the Crypto space with long-term investment plans.
- Anyone who's ready to earn a passive income in BUSD by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who's interested in taking part with meta Shiba Inu AR entertainment and VR gaming.
- Anyone who's interested in taking part with the future plans of the Meta Shiba Inu token.
- Anyone who's interested in making financial transactions with any other party using Meta Shiba Inu as the currency.

Core concept

The Meta Shiba Inu reward system

3% of each transaction when buying and selling get converted to BUSD and is split amongst all holders. Holders will be eligible to receive tokens every one hour and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

- The **sustainability fee of 1.5% when buying and selling for marketing and 2% when buying and 3% when selling for dev** is what allows Meta Shiba Inu to promote the token and use funds to further the development of the platform. Tokens will be swapped into BNB and will be sent to a marketing wallet and dev wallet This way, Meta Shiba Inu will have access to the funds without selling tokens as the traditional way, which will enable them to consume funds without hurting the project.

- **The liquidity fee of 1.5%** is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

- Meta Shiba Inu has the burn strategy that a **2% fee in each transaction when buying and selling** is getting charged that benefits and rewards those who invest long-term. This feature slowly reduces supply making each Meta Shiba Inu more and more valuable.

Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	9/10
3	Information quality	8/10
4	Service quality	8/10
5	System quality	8/10
6	Impact on the community	9/10
7	Impact on the business	9/10
8	Preparing for the future	9/10
Total Points		**8.625/10**

Contract details

Token contract details for 9th April 2022

Contract name	Meta Shiba Inu
Contract address	0x6fDfA944478Fd4D87B16902147062CcDC985f2eF
Token supply	1,000,000,000,000,000
Token ticker	MSHIB
Decimals	9
Token holders	3
Transaction count	4
Dividend tracker	0x512a2f5355ef18b75c7d19f97b4008092172b0b1
Reward token	0xe9e7cea3dedca5984780bafc599bd69add087d56
Marketing wallet	0xdecdfb80430c6e31cab9785aad63f54c8495b29e
Contract deployer address	-
Contract’s current owner address	0x51f2333a56d5ade9d71cdcfa3bbe2c1e3283774d

Tokens are distributed as follows:

TOKEN SUPPLY

MSHIB Tokenomics

SMART CONTRACT 100%

0x13A62150224CE665A95B6D8Ef2f0DAEa9a14a44c

MAX SUPPLY 100%

1,000,000,000,000,000 MSHIB

PRESALE AND LIQUIDITY 80%

Presale and Liquidity by PinkSale. [Liquidity Locked for 12 months]

MARKETING 5%

Promotions, Giveaways and Airdrops. [Locked for 2 months]

TEAM 15%

[Locked for 3-6 months]

Contract code function details

No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	pass
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	pass
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	pass
		Selfdestruct function security	pass
3	Business security	Access control of owners	pass
		Business logics	pass
		Business implementations	pass
4	Integer overflow/underflow		pass
5	Reentrancy		pass
6	Exceptional reachable state		pass
7	Transaction ordering dependence		pass
8	Block properties dependence		pass
9	Pseudo random number generator (PRNG)		pass
10	DoS (Denial of Service)		pass
11	Token vesting implementation		pass
12	Fake deposit		pass

13	**Event security**		**pass**

Contract description table

The below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions, and implementations with their visibility and mutability.

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
Ownable	**Implementation**	**Context**		
L		Public ❗	🛑	NO❗
L	owner	Public ❗		NO❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	_transferOwnership	Internal 🔒	🛑	
IERC20	**Interface**			
L	totalSupply	External ❗		NO❗
L	balanceOf	External ❗		NO❗
L	transfer	External ❗	🛑	NO❗
L	allowance	External ❗		NO❗

⌊	approve	External ❗	🛑	NO❗
⌊	transferFrom	External ❗	🛑	NO❗
SafeMath	**Library**			
⌊	tryAdd	Internal 🔒		
⌊	trySub	Internal 🔒		
⌊	tryMul	Internal 🔒		
⌊	tryDiv	Internal 🔒		
⌊	tryMod	Internal 🔒		
⌊	add	Internal 🔒		
⌊	sub	Internal 🔒		
⌊	mul	Internal 🔒		
⌊	div	Internal 🔒		
⌊	mod	Internal 🔒		
⌊	sub	Internal 🔒		
⌊	div	Internal 🔒		
⌊	mod	Internal 🔒		
IUniswapV2 Router01	**Interface**			
⌊	factory	External ❗		NO❗
⌊	WETH	External ❗		NO❗

L	addLiquidity	External ❗	🛑	NO❗
L	addLiquidityETH	External ❗	💵	NO❗
L	removeLiquidity	External ❗	🛑	NO❗
L	removeLiquidityETH	External ❗	🛑	NO❗
L	removeLiquidityWithPermit	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
L	swapExactTokensForTokens	External ❗	🛑	NO❗
L	swapTokensForExactTokens	External ❗	🛑	NO❗
L	swapExactETHForTokens	External ❗	💵	NO❗
L	swapTokensForExactETH	External ❗	🛑	NO❗
L	swapExactTokensForETH	External ❗	🛑	NO❗
L	swapETHForExactTokens	External ❗	💵	NO❗
L	quote	External ❗		NO❗
L	getAmountOut	External ❗		NO❗
L	getAmountIn	External ❗		NO❗
L	getAmountsOut	External ❗		NO❗
L	getAmountsIn	External ❗		NO❗
IUniswapV2 Router02	**Interface**	**IUniswapV2 Router01**		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗

L	swapExactTokensForTokensSupporting FeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactETHForTokensSupportingFee OnTransferTokens	External ❗	💵	NO❗
L	swapExactTokensForETHSupportingFee OnTransferTokens	External ❗	🛑	NO❗
IUniswapV2 Factory	**Interface**			
L	feeTo	External ❗		NO❗
L	feeToSetter	External ❗		NO❗
L	getPair	External ❗		NO❗
L	allPairs	External ❗		NO❗
L	allPairsLength	External ❗		NO❗
L	createPair	External ❗	🛑	NO❗
L	setFeeTo	External ❗	🛑	NO❗
L	setFeeToSetter	External ❗	🛑	NO❗
IDividend Distributor	**Interface**			
L	setDistributionCriteria	External ❗	🛑	NO❗
L	setShare	External ❗	🛑	NO❗
L	deposit	External ❗	🛑	NO❗
L	process	External ❗	🛑	NO❗
L	purge	External ❗	🛑	NO❗

Dividend Distributor	Implementation	IDividend Distributor		
L		Public ❗	🛑	NO❗
L		External ❗	💵	NO❗
L	setDistributionCriteria	External ❗	🛑	onlyToken
L	purge	External ❗	🛑	onlyToken
L	setShare	External ❗	🛑	onlyToken
L	deposit	External ❗	🛑	onlyToken
L	process	External ❗	🛑	onlyToken
L	shouldDistribute	Internal 🔒		
L	distributeDividend	Internal 🔒	🛑	
L	claimDividend	External ❗	🛑	NO❗
L	getUnpaidEarnings	Public ❗		NO❗
L	getHolderDetails	Public ❗		NO❗
L	getCumulativeDividends	Internal 🔒		
L	getLastProcessedIndex	External ❗		NO❗
L	getNumberOfTokenHolders	External ❗		NO❗
L	getShareHoldersList	External ❗		NO❗
L	totalDistributedRewards	External ❗		NO❗
L	addShareholder	Internal 🔒	🛑	
L	removeShareholder	Internal 🔒	🛑	

MSHIB	Implementation	IERC20, Ownable		
L		Public ❗	🛑	NO❗
L		External ❗	💵	NO❗
L	totalSupply	External ❗		NO❗
L	name	Public ❗		NO❗
L	symbol	Public ❗		NO❗
L	decimals	Public ❗		NO❗
L	balanceOf	Public ❗		NO❗
L	getHolderDetails	Public ❗		NO❗
L	getLastProcessedIndex	Public ❗		NO❗
L	getNumberOfTokenHolders	Public ❗		NO❗
L	totalDistributedRewards	Public ❗		NO❗
L	allowance	External ❗		NO❗
L	approve	Public ❗	🛑	NO❗
L	_approve	Internal 🔒	🛑	
L	approveMax	External ❗	🛑	NO❗
L	transfer	External ❗	🛑	NO❗
L	transferFrom	External ❗	🛑	NO❗
L	_transferFrom	Internal 🔒	🛑	
L	_basicTransfer	Internal 🔒	🛑	

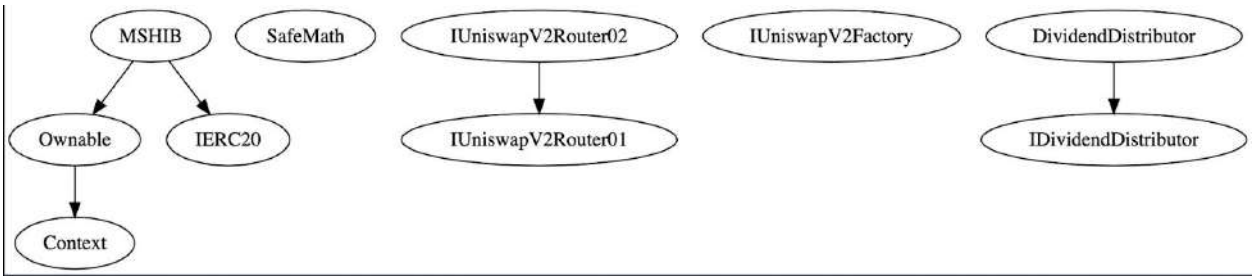
L	shouldTakeFee	Internal 🔒		
L	takeFee	Internal 🔒	🛑	
L	shouldSwapBack	Internal 🔒		
L	clearStuckBalance	External ❗	🛑	onlyOwner
L	getBep20Tokens	External ❗	🛑	onlyOwner
L	updateBuyFees	Public ❗	🛑	onlyOwner
L	updateSellFees	Public ❗	🛑	onlyOwner
L	updateSwapPercentages	Public ❗	🛑	onlyOwner
L	enableTrading	Public ❗	🛑	onlyOwner
L	whitelistPreSale	Public ❗	🛑	onlyOwner
L	___claimRewards	Public ❗	🛑	NO❗
L	claimProcess	Public ❗	🛑	NO❗
L	blackListWallets	Public ❗	🛑	onlyOwner
L	isBlacklisted	Public ❗		NO❗
L	isRewardExclude	Public ❗		NO❗
L	isFeeExclude	Public ❗		NO❗
L	isMaxSellExcluded	Public ❗		NO❗
L	isMaxBuyExcluded	Public ❗		NO❗
L	setMaxSellExclude	External ❗	🛑	onlyOwner
L	changeMaxSellToken	External ❗	🛑	onlyOwner
L	swapBackInBnb	Internal 🔒	🛑	swapping

L	swapAndLiquify	Private 🔐	🛑	
L	swapTokensForEth	Private 🔐	🛑	
L	swapTokensForTokens	Private 🔐	🛑	
L	addLiquidity	Private 🔐	🛑	
L	setIsDividendExempt	External ❗	🛑	onlyOwner
L	setIsFeeExempt	External ❗	🛑	onlyOwner
L	setMaxBuyExcluded	External ❗	🛑	onlyOwner
L	addAuthorizedWallets	External ❗	🛑	onlyOwner
L	setMarketingWallet	External ❗	🛑	onlyOwner
L	setTreasuryWallet	External ❗	🛑	onlyOwner
L	changeMaxBuyTokens	External ❗	🛑	onlyOwner
L	setSwapBackSettings	External ❗	🛑	onlyOwner
L	setDistributionCriteria	External ❗	🛑	onlyOwner
L	setDistributorSettings	External ❗	🛑	onlyOwner
L	purgeBeforeSwitch	Public ❗	🛑	onlyOwner
L	includeMeinRewards	Public ❗	🛑	NO❗
L	switchToken	Public ❗	🛑	onlyOwner

Legend

Symbol	Meaning
	Function can modify state
	Function is payable

Inheritance Hierarchy



Security issue checking status

- **High severity issues**
  No High severity issues found

- **Medium severity issues**
  No medium severity issues found

- **Low severity issues**
  No low severity issues found

- **Centralization Risk**
  No Centralization risk found

Owner privileges

- The owner can get contract bnb balance to owner wallet

```
ftrace | funcSig
function clearStuckBalance(uint256 amountPercentage) external onlyOwner {
    uint256 amountBNB = address(this).balance;
    payable(msg.sender).transfer((amountBNB * amountPercentage) / 100);
}
```

- The owner can get any bep20 tokens in contract to owner wallet

```
ftrace | funcSig
function getBep20Tokens(address _tokenAddress, uint256 amount)
    external
    onlyOwner
{
    require(
        IERC20(_tokenAddress).balanceOf(address(this)) >= amount,
        "No Enough Tokens"
    );
    IERC20(_tokenAddress).transfer(msg.sender, amount);
}
```

❖ The owner can change all buy and sell fees maximum up to 20%

```
ftrace | funcSig
function updateBuyFees(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 burn,
    uint256 treasury
) public onlyOwner {
    buyRewardFee = reward;
    buyMarketingFee = marketing;
    buyLiquidityFee = liquidity;
    buyBurnFee = burn;
    buyTreasuryFee = treasury;

    buyTotalFees = reward.add(marketing).add(liquidity).add(burn).add(
        treasury
    );

    require(buyTotalFees <= 2000, "Fees can not be greater than 20%");
}

ftrace | funcSig
function updateSellFees(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 burn,
    uint256 treasury
) public onlyOwner {
    sellRewardFee = reward;
    sellMarketingFee = marketing;
    sellLiquidityFee = liquidity;
    sellBurnFee = burn;
    sellTreasuryFee = treasury;

    sellTotalFees = reward.add(marketing).add(liquidity).add(burn).add(
        treasury
    );
    require(sellTotalFees <= 2000, "Fees can not be greater than 20%");
}
```

❖ The owner can change swap percentages

```
// update swap percentages
ftrace | funcSig
function updateSwapPercentages(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 treasury
) public onlyOwner {
    rewardSwap = reward;
    marketingSwap = marketing;
    liquiditySwap = liquidity;
    treasurySwap = treasury;

    totalSwap = reward.add(marketing).add(liquidity).add(treasury);
}

// switch Trading
```

❖ The owner can enable trading, once enabled cannot disable again

```
// switch Trading
ftrace | funcSig
function enableTrading() public onlyOwner {
    tradingOpen = true;
}
```

❖ The owner can whitelist presale address

```
ftrace | funcSig
function whitelistPreSale(address _preSale) public onlyOwner {
    isFeeExempt[_preSale] = true;
    isDividendExempt[_preSale] = true;
    isAuthorized[_preSale] = true;
}
```

❖ The owner can blacklist and unblock wallets

```
ftrace | funcSig
function blackListWallets(address wallet, bool _status) public onlyOwner {
    isBlacklist[wallet] = _status;
}
```

- The owner can include/exclude wallets from max sell limit and can change max sell amount minimum up to 1%

```
ftrace | funcSig
function setMaxSellExclude(address holder, bool exempt) external onlyOwner {
    isMaxSellExclude[holder] = exempt;
}

ftrace | funcSig
function changeMaxSellToken(uint256 percentage) external onlyOwner {
    require(percentage >= 1, "Max buy percentage can not be less than 1%");
    maxSellToken = _totalSupply.mul(percentage).div(100);
}
```

- The owner can include/exclude wallets from rewards

```
ftrace | funcSig
function setIsDividendExempt(address holder, bool exempt)
    external
    onlyOwner
{
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if (exempt) {
        dividendTracker.setShare(holder, 0);
    } else {
        dividendTracker.setShare(holder, _balances[holder]);
    }
}
```

- The owner can include/exclude wallets from fees

```
ftrace | funcSig
function setIsFeeExempt(address holder, bool exempt) external onlyOwner {
    isFeeExempt[holder] = exempt;
}
```

- The owner can include/exclude wallets from max buy limit

```
ftrace | funcSig
function setMaxBuyExcluded(address holder, bool exempt) external onlyOwner {
    isMaxBuyExclude[holder] = exempt;
}
```

- The owner can add/remove authorized wallets (only authorized wallets can do transactions when trading is disabled)

```
ftrace | funcSig
function addAuthorizedWallets(address holder, bool exempt)
    external
    onlyOwner
{
    isAuthorized[holder] = exempt;
}
```

- The owner can change marketing and treasury wallet

```
ftrace | funcSig
function setMarketingWallet(address _marketingFeeReceiver)
    external
    onlyOwner
{
    marketingWallet = _marketingFeeReceiver;
}

ftrace | funcSig
function setTreasuryWallet(address _wallet) external onlyOwner {
    treasuryWallet = _wallet;
}
```

- The owner can change max buy tokens minimum up to 1%

```
ftrace | funcSig
function changeMaxBuyTokens(uint256 percentage) external onlyOwner {
    require(percentage >= 1, "Max buy percentage can not be less than 1%");
    maxBuyToken = _totalSupply.mul(percentage).div(100);
}
```

- The owner can enable/disable swapping and can change swap point

```
ftrace | funcSig
function setSwapBackSettings(bool _enabled, uint256 _amount)
    external
    onlyOwner
{
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

- The owner can change minimum reward distribution period and can change minimum distribution reward amount

```
ftrace | funcSig
function setDistributionCriteria(
    uint256 _minPeriod,
    uint256 _minDistribution
) external onlyOwner {
    dividendTracker.setDistributionCriteria(_minPeriod, _minDistribution);
}
```

- The owner can get tokens in the reward tracker to owner wallet (this will use before change reward token address)

```
ftrace | funcSig
function purgeBeforeSwitch() public onlyOwner {
    dividendTracker.purge(msg.sender);
}
```

❖ The owner can change reward token address

```
ftrace | funcSig
function switchToken(address rewardToken, bool isIncludeHolders)
    public
    onlyOwner
{
    require(
        rewardToken != router.WETH(),
        "Can not reward BNB in this tracker"
    );
    REWARD = rewardToken;
    // get current shareholders list
    address[] memory currentHolders = dividendTracker.getShareHoldersList();
    dividendTracker = new DividendDistributor(rewardToken);
    if (isIncludeHolders) {
        // add old share holders to new tracker
        for (uint256 i = 0; i < currentHolders.length; i++) {
            try
                dividendTracker.setShare(
                    currentHolders[i],
                    _balances[currentHolders[i]]
                )
            {} catch {}
        }
    }

    emit ChangeRewardTracker(rewardToken);
}
```

Audit conclusion

RugFreeCoins team has performed in-depth testings, line by line manual code review, and automated audit of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status: **PASSED**

Number of risk issues: **0**

Solidity code functional issue level: **PASSED**

Number of owner privileges: **18**

Centralization risk correlated to the active owner: **LOW**

Smart contract active ownership: **YES**