RugFreeCoins Audit

DOGEBOT Token Audit

Smart Contract Security Audit

August 25, 2021

Contents

Audit details	1
Disclaimer	2
Background	3
About the project	4
Target market and the concept	7
Potential to grow with score points	9
Total Points	9
Contract details	10
Token distribution	11
Contract code function details	12
Contract description table	13
Security issue checking status	22
Owner privileges	24
Audit conclusion	26

Audit details

Audited project

DOGEBOT Token

Contract Address

0xb99261a78ee7f7bde822eb87c80c23dfb4333dc8

Client contact

DOGEBOT Token Team

Blockchain

Binance smart chain

Project website

https://www.dogeb0t.com/

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 DOGEBOT to perform an audit of the smart contract.

https://www.bscscan.com/address/0xb99261a78ee7f7bde822eb87c80c23dfb4333dc8

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 improve the security posture of the smart contract by remediating the issues that were identified.

About the project

DOGEB0T is a token built on the Binance Smart Chain. Each transaction, purchase incur a 7% fee, and sales incur a 14% fee. Also, manual buying tax of 15% and selling tax of 30% will be in place to activate the shield when a big sell off has happened or when the price is dropping for some weeks in a row.

DOGEB0T is the first project to have competitive liquidity mining. That means that in parallel to the liquidity available in the exchanges, there is a pool reserved for competitions in which users will get free DB0T tokens to basically carry out marketing tasks. This revolutionary and smart concept will act as a top layer of our marketing campaigns as holders are mining tokens while helping the project grow in holders' market capitalization.

Features

- 5% of regular fee when buying, 10% of regular fee when selling, 1% of shield fee when buying and 8.5% of shield fee when selling of each transaction gets converted to BNBs and is split amongst all holders. The holders will be eligible to receive tokens everyone hour and rewards are proportional to how many tokens each individual holds.

- **The liquidity fee of 2%** of regular fee when buying, 4% of regular fee when selling, 14% of shield fee when buying and 21.5% of shield fee when selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

Tokenomics

7% fee when buying

- 85% of trade goes to holders' pockets in BNB.
- 2% of trade goes to a private wallet for liquidity.

14% fee when selling

- 10% of trade goes to holders' pockets in BNB.
- 4% of trade goes to a private wallet for liquidity.

Shield protocol (Manual fee setup)

. **15% fee when buying**

- 1% of trade goes to holders' pockets in BNB.
- 14% of trade goes to a private wallet for liquidity.

30% fee when selling

- 8.5% of trade goes to holders' pockets in BNB.
- 21.5% of trade goes to a private wallet for liquidity.

Roadmap

Ignition

- Initial marketing campaign
- Top vote webs listing
- Full audit
- First liquidity competition
- Presale on Dxsale

Launch

- PancakeSwap Listing (trading ON)
- Nuclear ignition (1st 5% Burning)
- Trip welcome (1st 5% airdrop)
- 2000 DB0T holders
- 10BNB Giveaway competition

Impulse

- Coingeko listing
- Coinmarketcap listing
- 25000 Telegram members
- 10000 DB0T holders
- Blackhole (2nd 5% burning)

Boost

- Wormhole gift 5% airdrop
- Chart Viewer v1
- NFT MarketPlace DCL
- Merchandise store
- Exchange listings

Target market and the concept

Target market

- Anyone who's interested in Crypto space with long term investment plans.
- Anyone who's ready to earn a passive income from BNB by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who's interested in taking part with DOGEB0T future plans.
- Anyone who's interested in taking part in making decisions in terms of marketing and development of the CaptainBNB ecosystem being a token holder.
- Anyone who's interested in making financial transactions with any other party DOGEB0T using as the currency.
- Anyone who's interested in doing marketing tasks and getting paid in BNB or DOGEB0T tokens.

Core concept

DOGEB0T is a competitive liquidity mining project to mine DB0Ts, BNBs, and prices in weekly competitions by doing tasks.

The BNB reward system

5% of regular fee when buying, 10% of regular fee when selling, 1% of shield fee when buying and 8.5% of shield fee when selling of each transaction gets converted to BNBs and is split amongst all holders. The rewards are sent to holders that have at least 5,000,000 DOGEB0T tokens, holders will be eligible to receive tokens everyone hour and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The marketing wallet will hold 6% tokens from the total supply that reflection will be distributed to the wallet proportional to the amounts that the wallet holds. This way, DOGEB0T 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 2% of regular fee when buying, 4% of regular fee when selling, 14% of shield fee when buying and 21.5% of shield fee when selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

Anti-dumping

The DOGEB0T contract includes a function that stops all sales above 0.05% of the total supply. This will discourage (mini)-whales from dumping all their bags at once.

Anti-whale strategy

Anti-pump and dump: Certain groups of individuals practice pump and dump schemes, in order to lure in outside investors by the looks of a bullish chart, and sell at a high point. DOGEB0T have shield mechanism in place, where owners will manually activate the shield when big sell off happens or when the price is dropping for some weeks in a row, and a large portion from the fee will be allocated for liquidity in a private wallet.

The Protection Shield will be manually activated by owners and suddenly trigger a change in the reflection algorithm to increase the sell fee and buy fee to protect the liquidity and market capitalization. This will cause more friction to drop and a gentle levitation force to rise while the market cap is kept healthy.

Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	9/10
3	Information quality	9/10
4	Service quality	9/10
5	System quality	9/10
6	Impact on the community	9/10
7	Impact on the business	9/10
8	Preparing for the future	8/10
Total Points		**8.88/10**

Contract details

Token contract details for 25th August 2021

Contract name	DOGEBOT
Contract address	0xb99261a78ee7f7bde822eb87c80c23dfb4333dc8
Token supply	100,000,000,000
Token ticker	DB0T
Decimals	18
Token holders	2
Transaction count	2
Liquidity wallet	0xaf5e609025d06a8019c22009625807fe089609ad
Dividend Tracker	0x6e0b11a2c43a8ae66b7b4bcc589204b8fffb01cf
Contract deployer address	0xAF5E609025D06a8019c22009625807Fe089609Ad
Contract's current owner address	0xaf5e609025d06a8019c22009625807fe089609ad

Token distribution

Tokens are distributed as follows:

70% of the circulating supply will be distributed to the holders in presale, competitions, and airdrops. The liquidity will be locked for 1 year and 75% of the presale liquidity will be directly added to Pancakeswap.

Within the liquidity supply, 10% is locked and reserved for the nuclear ignition and blackhole (Video streamed burning), 3% of the supply will be locked for trip welcome and wormhole gift (Video streamed airdrops), 6% will be kept on spaceship wallet for marketing and events, 3% will be kept for ecosystem developing, audits and NFT, 2% will be a lock for DOGEB0T team and 2% Dxsale Presale fees.

The marketing wallet will hold 6% tokens from the total supply that reflection will be distributed to the wallet proportional to the amounts that the wallet holds.

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	informational
		Business logics	informational
		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

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 its visibility and mutability.

Contract	Type	Bases		
└	Function Name	Visibility	Mutability	Modifiers
IERC20	**Interface**			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IERC20Metadata	**Interface**	**IERC20**		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
SafeMath	**Library**			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		

└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
ERC20	**Implementation**	**Context, IERC20, IERC20Metadata**		
└		Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
SafeMathUint	**Library**			
└	toInt256Safe	Internal 🔒		
SafeMathInt	**Library**			
└	mul	Internal 🔒		

└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
DividendPayingTokenInterface	**Interface**			
└	dividendOf	External ❗		NO❗
└	distributeDividends	External ❗	💰	NO❗
└	withdrawDividend	External ❗	🛑	NO❗
DividendPayingTokenOptionalInterface	**Interface**			
└	withdrawableDividendOf	External ❗		NO❗
└	withdrawnDividendOf	External ❗		NO❗
└	accumulativeDividendOf	External ❗		NO❗
DividendPayingToken	**Implementation**	**ERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface**		
└		Public ❗	🛑	ERC20
└		External ❗	💰	NO❗
└	distributeDividends	Public ❗	💰	NO❗
└	withdrawDividend	Public ❗	🛑	NO❗
└	_withdrawDividendOfUser	Internal 🔒	🛑	
└	dividendOf	Public ❗		NO❗
└	withdrawableDividendOf	Public ❗		NO❗
└	withdrawnDividendOf	Public ❗		NO❗
└	accumulativeDividendOf	Public ❗		NO❗

└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_setBalance	Internal 🔒	🛑	
IterableMapping	**Library**			
└	get	Public ❗		NO❗
└	getIndexOfKey	Public ❗		NO❗
└	getKeyAtIndex	Public ❗		NO❗
└	size	Public ❗		NO❗
└	set	Public ❗	🛑	NO❗
└	remove	Public ❗	🛑	NO❗
Ownable	**Implementation**	**Context**		
└		Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	getUnlockTime	Public ❗		NO❗
└	getTime	Public ❗		NO❗
└	lock	Public ❗	🛑	onlyOwner
└	unlock	Public ❗	🛑	NO❗
IUniswapV2Pair	**Interface**			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗

L	allowance	External ❗		NO❗
L	approve	External ❗	●	NO❗
L	transfer	External ❗	●	NO❗
L	transferFrom	External ❗	●	NO❗
L	DOMAIN_SEPARATOR	External ❗		NO❗
L	PERMIT_TYPEHASH	External ❗		NO❗
L	nonces	External ❗		NO❗
L	permit	External ❗	●	NO❗
L	MINIMUM_LIQUIDITY	External ❗		NO❗
L	factory	External ❗		NO❗
L	token0	External ❗		NO❗
L	token1	External ❗		NO❗
L	getReserves	External ❗		NO❗
L	price0CumulativeLast	External ❗		NO❗
L	price1CumulativeLast	External ❗		NO❗
L	kLast	External ❗		NO❗
L	mint	External ❗	●	NO❗
L	burn	External ❗	●	NO❗
L	swap	External ❗	●	NO❗
L	skim	External ❗	●	NO❗
L	sync	External ❗	●	NO❗
L	initialize	External ❗	●	NO❗
IUniswapV2Factory	**Interface**			
L	feeTo	External ❗		NO❗
L	feeToSetter	External ❗		NO❗
L	getPair	External ❗		NO❗
L	allPairs	External ❗		NO❗

└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🛑	NO❗
└	setFeeTo	External ❗	🛑	NO❗
└	setFeeToSetter	External ❗	🛑	NO❗
IUniswapV2Router01	**Interface**			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityETH	External ❗	💰	NO❗
└	removeLiquidity	External ❗	🛑	NO❗
└	removeLiquidityETH	External ❗	🛑	NO❗
└	removeLiquidityWithPermit	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
└	swapExactTokensForTokens	External ❗	🛑	NO❗
└	swapTokensForExactTokens	External ❗	🛑	NO❗
└	swapExactETHForTokens	External ❗	💰	NO❗
└	swapTokensForExactETH	External ❗	🛑	NO❗
└	swapExactTokensForETH	External ❗	🛑	NO❗
└	swapETHForExactTokens	External ❗	💰	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗
IUniswapV2Router02	**Interface**	**IUniswapV2Router01**		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗

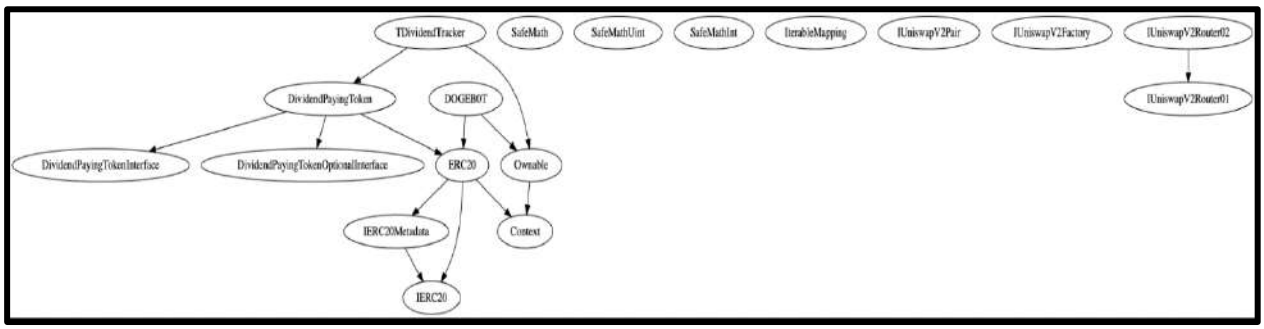
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External		NO
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External		NO
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External		NO
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External		NO
DOGEB0T	**Implementation**	**ERC20, Ownable**		
L		Public		ERC20
L		External		NO
L	whitelistDxSale	Public		onlyOwner
L	updateDividendTracker	Public		onlyOwner
L	updateUniswapV2Router	Public		onlyOwner
L	excludeFromFees	Public		onlyOwner
L	excludeMultipleAccountsFromFees	Public		onlyOwner
L	setMaxTxAmount	External		onlyOwner
L	setSwapTokensAtAmount	External		onlyOwner
L	setAutomatedMarketMakerPair	Public		onlyOwner
L	_setAutomatedMarketMakerPair	Private		
L	setBNBRewardFee	External		onlyOwner
L	setLiquidityFee	External		onlyOwner
L	updateLiquidityWallet	Public		onlyOwner
L	updateGasForProcessing	Public		onlyOwner
L	updateClaimWait	External		onlyOwner
L	getClaimWait	External		NO
L	getTotalDividendsDistributed	External		NO

L	isExcludedFromFees	Public ❗		NO❗
L	withdrawableDividendOf	Public ❗		NO❗
L	dividendTokenBalanceOf	Public ❗		NO❗
L	getAccountDividendsInfo	External ❗		NO❗
L	getAccountDividendsInfoAtIndex	External ❗		NO❗
L	processDividendTracker	External ❗	🛑	NO❗
L	claim	External ❗	🛑	NO❗
L	getLastProcessedIndex	External ❗		NO❗
L	getNumberOfDividendTokenHolders	External ❗		NO❗
L	_transfer	Internal 🔒	🛑	
L	swapAndLiquify	Private 🔐	🛑	
L	swapTokensForEth	Private 🔐	🛑	
L	addLiquidity	Private 🔐	🛑	
L	swapAndSendDividends	Private 🔐	🛑	
TDividendTracker	**Implementation**	**DividendPayingToken, Ownable**		
L		Public ❗	🛑	DividendPayingToken
L	_transfer	Internal 🔒	🛑	
L	withdrawDividend	Public ❗	🛑	NO❗
L	excludeFromDividends	External ❗	🛑	onlyOwner
L	updateClaimWait	External ❗	🛑	onlyOwner
L	getLastProcessedIndex	External ❗		NO❗
L	getNumberOfTokenHolders	External ❗		NO❗
L	getAccount	Public ❗		NO❗
L	getAccountAtIndex	Public ❗		NO❗
L	canAutoClaim	Private 🔐		
L	setBalance	External ❗	🛑	onlyOwner
L	process	Public ❗	🛑	NO❗
L	processAccount	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.**
- **Informational**
  - **Liquidity is sent to the owner's wallet, but the wallet address can be changed from the contract write section as it's an owner's privilege and the owner already added the correct liquidity address.**

```
ftrace | funcSig
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(uniswapV2Router), tokenAmount);

    // add the liquidity
    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        liquidityWallet,
        block.timestamp
    );
}
```

```
dividendTracker = new TDividendTracker();

liquidityWallet = owner();

IUniswapV2Router02 uniswapV2Router = IUniswapV
```

The addLiquidity function calls the uniswapV2Router.addLiquidityETH function with the address specified as owner() for acquiring the generated LP tokens from the DOGEBOT-BNB pool. As a result,over time the _owner address will accumulate a significant portion of LP tokens. If the _owner is an EOA(Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation:

We advise the address of the uniswapV2Router.addLiquidityETH function call to be replaced by the contract itself.

- **No maximum fee limits**

```
ftrace | funcSig
function setBNBRewardFee(uint256 newFee) external onlyOwner {
    BNBRewardsFee = newFee;
    totalFees = BNBRewardsFee.add(liquidityFee);
}

ftrace | funcSig
function setLiquidityFee(uint256 newFee) external onlyOwner {
    liquidityFee = newFee;
    totalFees = BNBRewardsFee.add(liquidityFee);
}
```

Owner can change fees without any limitation.

Recommendation:

It’s better to add a maximum fee limit validation.

Owner privileges

- The owner can update v2 Router address.

```
ftrace | funcSig
function updateUniswapV2Router(address newAddress) public onlyOwner {
    require(
        newAddress != address(uniswapV2Router),
        "The router already has that address"
    );
    emit UpdateUniswapV2Router(newAddress, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress);
}
```

- The owner can exclude wallets from fees.

```
ftrace | funcSig
function excludeFromFees(address account, bool excluded) public onlyOwner {
    require(
        _isExcludedFromFees[account] != excluded,
        "Account is already the value of 'excluded'"
    );
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}
```

- The owner can change the max transaction amount.

```
ftrace | funcSig
function setMaxTxAmount(uint256 amount) external onlyOwner {
    maxTransactionAmount = amount * 10**18;
}
```

❖ The owner can change BNB reward and liquidity fee.

```
ftrace | funcSig
function setBNBRewardFee(uint256 newFee) external onlyOwner {
    BNBRewardsFee = newFee;
    totalFees = BNBRewardsFee.add(liquidityFee);
}

ftrace | funcSig
function setLiquidityFee(uint256 newFee) external onlyOwner {
    liquidityFee = newFee;
    totalFees = BNBRewardsFee.add(liquidityFee);
}
```

❖ The owner can update the liquidity wallet.

```
ftrace | funcSig
function updateLiquidityWallet(address newLiquidityWallet)
    public
    onlyOwner
{
    require(
        newLiquidityWallet != liquidityWallet,
        "The liquidity wallet is already this address"
    );
    excludeFromFees(newLiquidityWallet, true);
    emit LiquidityWalletUpdated(newLiquidityWallet, liquidityWallet);
    liquidityWallet = newLiquidityWallet;
}
```

❖ The owner can update the claim wait.

```
ftrace | funcSig
function updateClaimWait(uint256 claimWait) external onlyOwner {
    dividendTracker.updateClaimWait(claimWait);
}
```

- The owner can exclude wallets from dividend.

```
ftrace | funcSig
function excludeFromDividends(address account) external onlyOwner {
    require(!excludedFromDividends[account]);
    excludedFromDividends[account] = true;

    _setBalance(account, 0);
    tokenHoldersMap.remove(account);

    emit ExcludeFromDividends(account);
}
```

- The owner can change the token swap trigger amount.

```
ftrace | funcSig
function setSwapTokensAtAmount(uint256 amount) external onlyOwner {
    swapTokensAtAmount = amount * 10**18;
}
```

Audit conclusion

While conducting the audit of the DOGEBOT smart contract, it was observed that there is nothing alarming with the code and it only contains informational concerns.