RugFreeCoins Audit

Sugar Daddy Doge Token

Smart Contract Security Audit

July 30, 2021

Contents

Audit details

Audited project

SugarDaddyDoge Token

Contract Address

0x5C4137ac4f0AF3830Fd3E2276E44E4a6e02f00b8

Client contact

SugarDaddyDoge Team

Blockchain

Binance smart chain

Project website

https://sugardaddydoge.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 SugarDaddyDoge to perform an audit of the smart contract.

https://bscscan.com/token/0x5C4137ac4f0AF3830Fd3E2276E44E4a6e02f00b8

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

Sugar Daddy Doge is a token built on the Binance Smart Chain. Each transaction, purchase and sale incur a 14% fee. The token was launched on Pancakeswap on the 6th of July.

Features:

- ❖ The automatic BNB reward of 7% is what SugarDaddyDoge's entire marketing strategy is based around: that tokens will be distributed among every holder proportional to how many tokens each individual hold.

- ❖ The liquidity fee of 3%, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity. This is a key element for decentralized exchanges like Pancakeswap.

- ❖ The sustainability fee of 3% marketing is what allows SugarDaddyDoge to hold the aforementioned promise. Tokens will be swapped into BNBs and will be sent to a marketing wallet per transaction. This way, SugarDaddyDoge will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.

- ❖ Controlled buyback wallet is used to save from massive dips in order to keep the token market price stable.

Tokenomics

- 7% of every trade goes to holders pockets in BNB.
- 3% of every trade goes to the marketing wallet.
- 3% of every trade goes to the liquidity pool.
- 1% of every trade goes to buyback tokens.

Roadmap

✓	Fair Launch on pancakeswap
✓	Dox + Audit
✓	Liquidity Locked
✓	Team Building
✓	Rap Anthem Release
	Coingecko Listing
	CoinMarketCap Listing
	Trustwallet Logo
	Project Awareness Campaigns
	Earnings Dashboard
	CERTIK AUDIT
	CEX listings
	Partnership Announcement
	Legal Entity
	Bridge Development
	Mass Adoption
	Partnerships
	Upgrading BNB Reward Pool
	Merchandise Release
	Top 100 Crypto Ranking

Doxxed Team

Mia
(Founder)

Mia (K Ekta) is the creator of the Sugar Daddy Doge token. She has a vision for meme coins to provide passive income, utility, transparency & security to investors. That's why she has doxxed, with complete transparency from the onset & is always available to her community on the voice chat or via DM.

Mia is a successful entrepreneur who has built many successful startups in the Digital Marketing and Consulting space, with firms based in the United States. She holds a graduate degree in Physics & is passionate about capturing new markets & business innovation.

She has incorporated purposeful token utilities which continue to incentivize investors. She integrated & innovated the popular tokenomics being used on Decentralized Tokens in the BSD or Binance Smart chain network and executed them with complete transparency.

It is difficult to find trust & transparency in the De-Fi space, however Mia is doxxed, intends to register Sugar Daddy as a legal entity in the US and everyone in her community feels secure with her at the helm. Mia built a strong organic community that is expected to grow massively. Her ideas are long-term as well as short-term, to keep attracting investors and she intends to go towards mass adoption.

Anthony Agape
Social Media Manager

Tixon Molianov
Tech Advisor

Timo Trippler
Advisor

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 by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who's interested in making financial transactions with any other party using SugarDaddyDoge as the currency.

Core concept

The BNB reward system

7% of each transaction gets converted to BNBs and is split amongst all holders. The rewards are sent to holders that have at least 10,000 SugarDaddyDoge tokens, holders will be eligible to receive tokens every one hour and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The liquidity fee of 3%, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

The fee of 3% marketing is what allows SugarDaddyDoge to promote the token and use funds to further development of the platform. Tokens will be swapped into BNBs and will be sent to a marketing wallet per transaction. This way, SugarDaddyDoge 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 controlled buyback collects 1% tax on each transaction, which is stored inside a private wallet. It is to save from massive dips in order to keep the token market price stable.

Potential to grow with score points

1.	Project efficiency	7/10
2.	Project uniqueness	6/10
3	Information quality	4/10
4	Service quality	710
5	System quality	7/10
6	Impact on the community	8/10
7	Impact on the business	8/10
8	Preparing for the future	6/10
Total Points		**6.63/10**

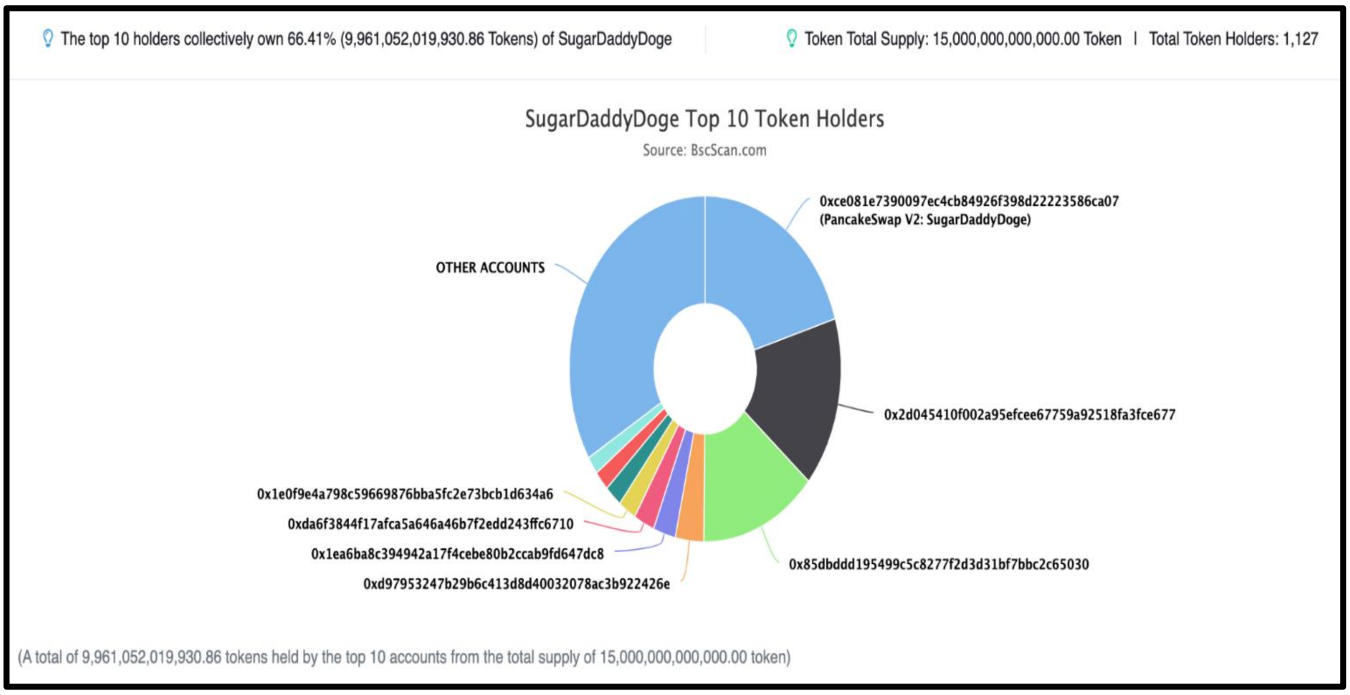
Contract details

Token contract details for 30th July 2021

Contract name	SugarDaddyDoge
Contract address	0x5C4137ac4f0AF3830Fd3E2276E44E4a6e02f00b8
Token supply	15,000,000,000,000
Token ticker	SugarDaddyDoge
Decimals	18
Token holders	1,127
Transaction count	16,261
Top 100% holders dominance	91.93%
Liquidity wallet	0xda6f3844f17afca5a646a46b7f2edd243ffc6710
Dividend Tracker	0xaa3faa4ca349f0041542bb024f3a5a51abfc6401
Contract deployer address	0xDA6f3844F17afCA5a646a46b7f2EdD243FfC6710
Contract's current owner address	0xda6f3844f17afca5a646a46b7f2edd243ffc6710

Top token holders

Top 10 Token Holders



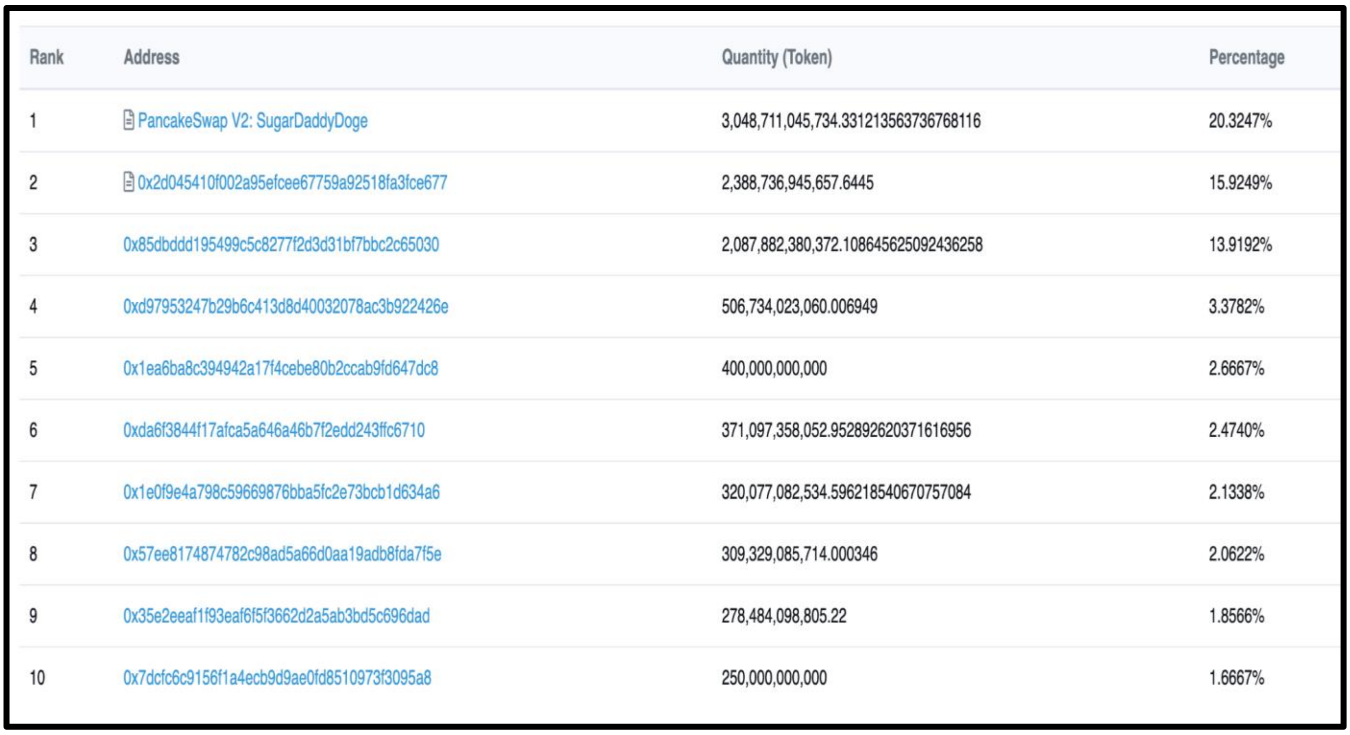
The top 10 holders collectively own 66.41% (9,961,052,019,930.86 Tokens) of SugarDaddyDoge

Token Total Supply: 15,000,000,000,000.00 Token | Total Token Holders: 1,127

SugarDaddyDoge Top 10 Token Holders

Source: BscScan.com

(A total of 9,961,052,019,930.86 tokens held by the top 10 accounts from the total supply of 15,000,000,000,000.00 token)

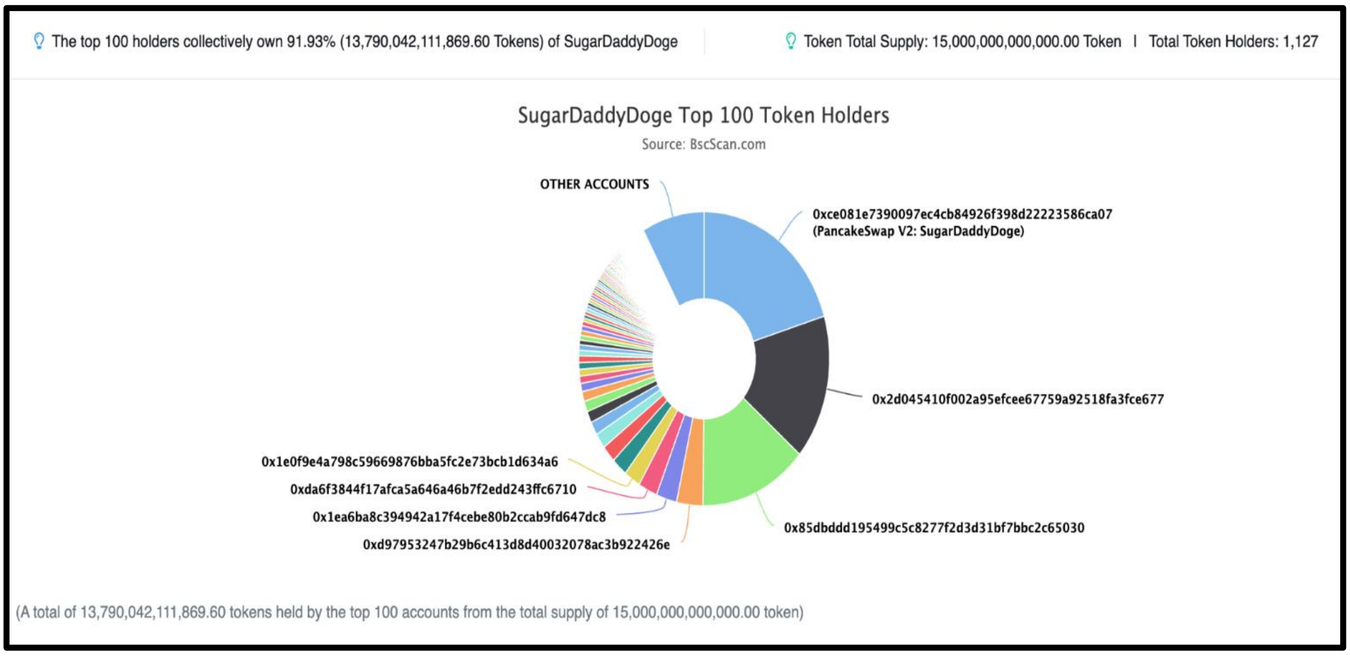


Rank	Address	Quantity (Token)	Percentage
1	PancakeSwap V2: SugarDaddyDoge	3,048,711,045,734.331213563736768116	20.3247%
2	0x2d045410f002a95efcee67759a92518fa3fce677	2,388,736,945,657.6445	15.9249%
3	0x85dbddd195499c5c8277f2d3d31bf7bbc2c65030	2,087,882,380,372.108645625092436258	13.9192%
4	0xd97953247b29b6c413d8d40032078ac3b922426e	506,734,023,060.006949	3.3782%
5	0x1ea6ba8c394942a17f4cebe80b2ccab9fd647dc8	400,000,000,000	2.6667%
6	0xda6f3844f17afca5a646a46b7f2edd243ffc6710	371,097,358,052.952892620371616956	2.4740%
7	0x1e0f9e4a798c59669876bba5fc2e73bcb1d634a6	320,077,082,534.596218540670757084	2.1338%
8	0x57ee8174874782c98ad5a66d0aa19adb8fda7f5e	309,329,085,714.000346	2.0622%
9	0x35e2eeaf1f93eaf6f5f3662d2a5ab3bd5c696dad	278,484,098,805.22	1.8566%
10	0x7dcfc6c9156f1a4ecb9d9ae0fd8510973f3095a8	250,000,000,000	1.6667%

Token distribution

Tokens are distributed as follows:

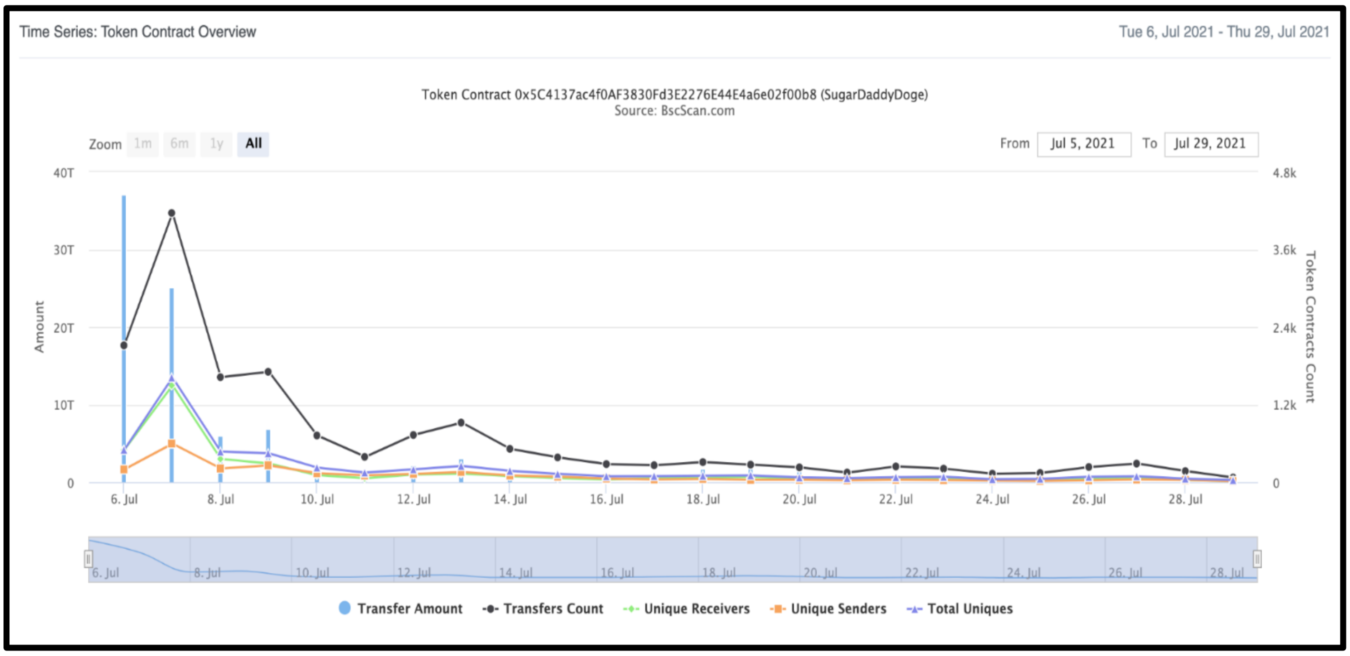
Top 100 Token Holders


The top 100 holders collectively own 91.93% (13,790,042,111,869.60 Tokens) of SugarDaddyDoge

Token Total Supply: 15,000,000,000,000.00 Token | Total Token Holders: 1,127

(A total of 13,790,042,111,869.60 tokens held by the top 100 accounts from the total supply of 15,000,000,000,000.00 token)

Contract interaction details


Time Series: Token Contract Overview

Tue 6, Jul 2021 - Thu 29, Jul 2021

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

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**		
L	Function Name	Visibility	Mutability	Modifiers
SugarDaddyDoge	**Implementation**	**ERC20, Ownable**		
L		Public ❗	🛑	ERC20
L		External ❗	💵	NO❗
L	updateDividendTracker	Public ❗	🛑	onlyOwner
L	updateUniswapV2Router	Public ❗	🛑	onlyOwner
L	excludeFromFees	Public ❗	🛑	onlyOwner
L	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
L	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
L	_setAutomatedMarketMakerPair	Private 🔐	🛑	
L	updateGasForProcessing	Public ❗	🛑	onlyOwner
L	updateClaimWait	External ❗	🛑	onlyOwner
L	getClaimWait	External ❗		NO❗
L	getTotalDividendsDistributed	External ❗		NO❗
L	isExcludedFromFees	Public ❗		NO❗
L	withdrawableDividendOf	Public ❗		NO❗

└	dividendTokenBalanceOf	Public ❗️		NO❗️
└	getAccountDividendsInfo	External ❗️		NO❗️
└	getAccountDividendsInfoAtIndex	External ❗️		NO❗️
└	processDividendTracker	External ❗️	🛑	NO❗️
└	claim	External ❗️	🛑	NO❗️
└	getLastProcessedIndex	External ❗️		NO❗️
└	setSwapTokensAtAmt	External ❗️	🛑	onlyOwner
└	withdraw	External ❗️	🛑	onlyOwner
└	getNumberOfDividendTokenHolders	External ❗️		NO❗️
└	_transfer	Internal 🔒	🛑	
└	setBNBRewardsfee	External ❗️	🛑	onlyOwner
└	swapAndSendToFee	Private 🔐	🛑	
└	excludeFromDividends	External ❗️	🛑	onlyOwner
└	setLiquidityFee	External ❗️	🛑	onlyOwner
└	setMarketingFee	External ❗️	🛑	onlyOwner
└	setMarketingWallet	External ❗️	🛑	onlyOwner
└	setBuybackFee	External ❗️	🛑	onlyOwner
└	setBuybackWallet	External ❗️	🛑	onlyOwner
└	swapAndLiquify	Private 🔐	🛑	
└	swapTokensForEth	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	

L	swapAndSendDividends	Private 🔐	🛑	
SugarDaddyDogeDividendTracker	**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
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
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 🔒	🛑	
DividendPayingTokenInterface	**Interface**			
└	dividendOf	External ❗		NO❗
└	distributeDividends	External ❗	💵	NO❗
└	withdrawDividend	External ❗	🛑	NO❗
DividendPayingTokenOptionalInterface	**Interface**			
└	withdrawableDividendOf	External ❗		NO❗
└	withdrawnDividendOf	External ❗		NO❗

└	accumulativeDividendOf	External ❗		NO❗
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 🔒	🛑	
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❗
IterableMapping	**Library**			
└	get	Public ❗		NO❗
└	getIndexOfKey	Public ❗		NO❗
└	getKeyAtIndex	Public ❗		NO❗
└	size	Public ❗		NO❗
└	set	Public ❗	🛑	NO❗
└	remove	Public ❗	🛑	NO❗
IUniswapV2Factory	**Interface**			
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🛑	NO❗
└	setFeeTo	External ❗	🛑	NO❗
└	setFeeToSetter	External ❗	🛑	NO❗

IUniswapV2Pair	**Interface**			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🛑	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗
└	mint	External ❗	🛑	NO❗
└	burn	External ❗	🛑	NO❗

└	swap	External ❗	🛑	NO❗
└	skim	External ❗	🛑	NO❗
└	sync	External ❗	🛑	NO❗
└	initialize	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❗

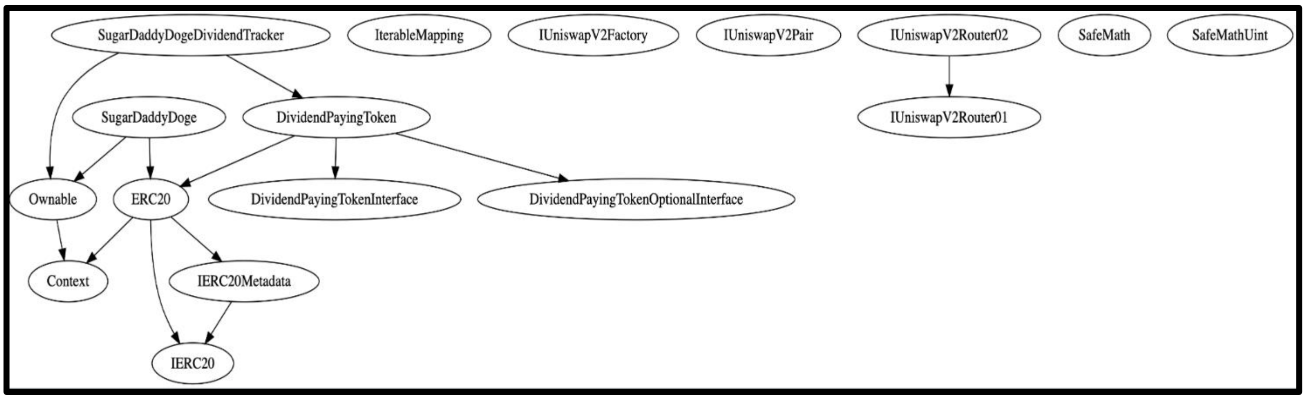
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗
IUniswapV2Router02	**Interface**	**IUniswapV2Router01**		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
Ownable	**Implementation**	**Context**		
└		Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
SafeMath	**Library**			
└	add	Internal 🔒		

└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
SafeMathUint	**Library**			
└	toInt256Safe	Internal 🔒		
SafeMathUint	**Library**			
└	toInt256Safe	Internal 🔒		

Legend

Symbol	**Meaning**
🛑	**Function can modify state**
💵	**Function is payable**

Inheritance Hierarchy



Security issue checking status

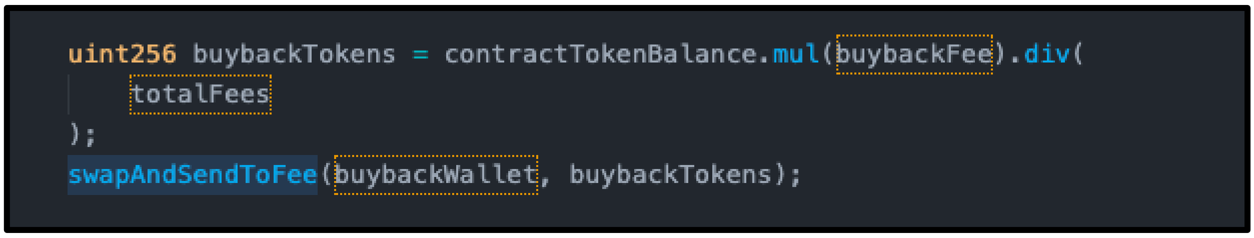
❖ **High severity issues**
No high severity issues found.

❖ **Medium severity issues**
No medium severity issues found.

❖ **Low severity issues**

Manual Buy Back

Buyback is not happening automatically. It's a controlled buyback mechanism where the fee is getting swapped to BNB and getting sent to a private wallet.

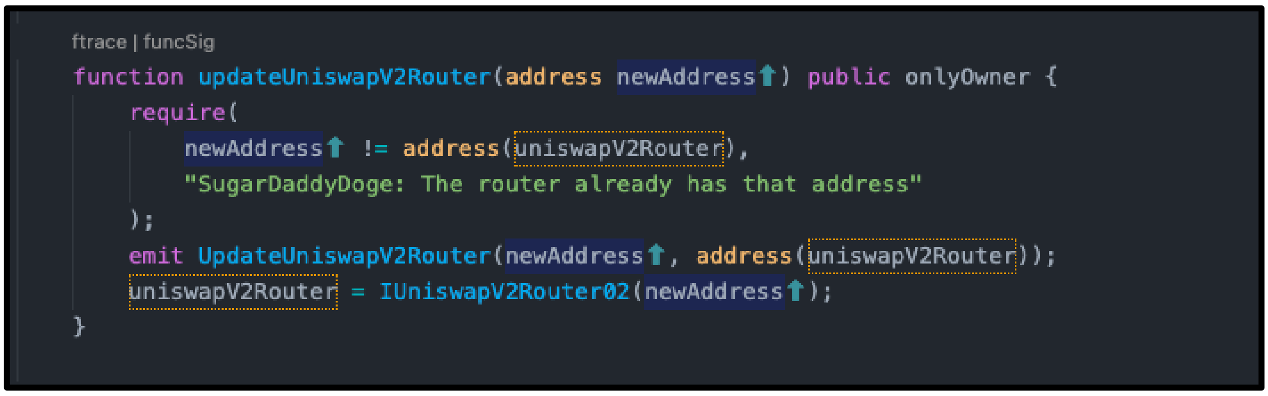


```
uint256 buybackTokens = contractTokenBalance.mul(buybackFee).div(
    totalFees
);
swapAndSendToFee(buybackWallet, buybackTokens);
```

Recommended having automatic buyback as it should be the ideal way to keep the mechanism dynamically handled.

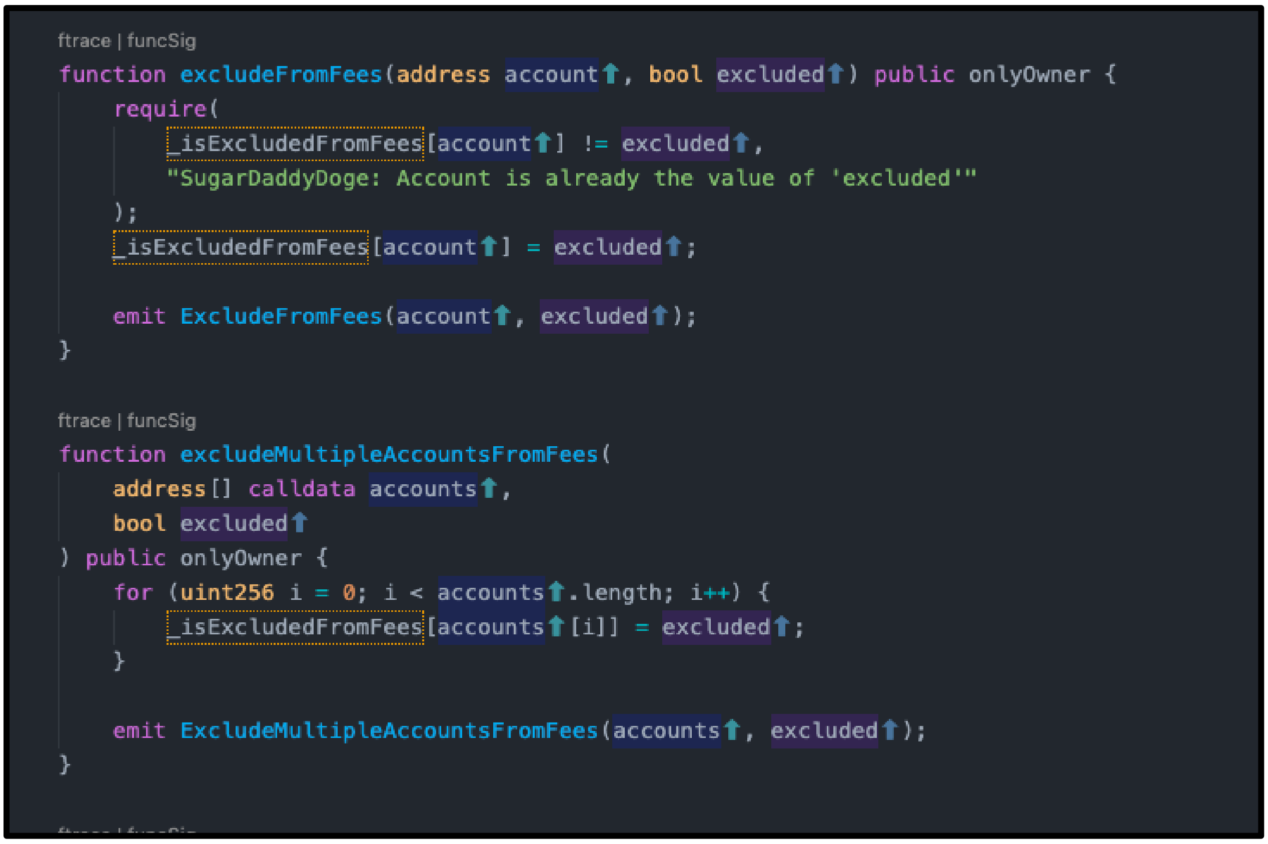
Owner privileges

❖ The owner can change the router address.



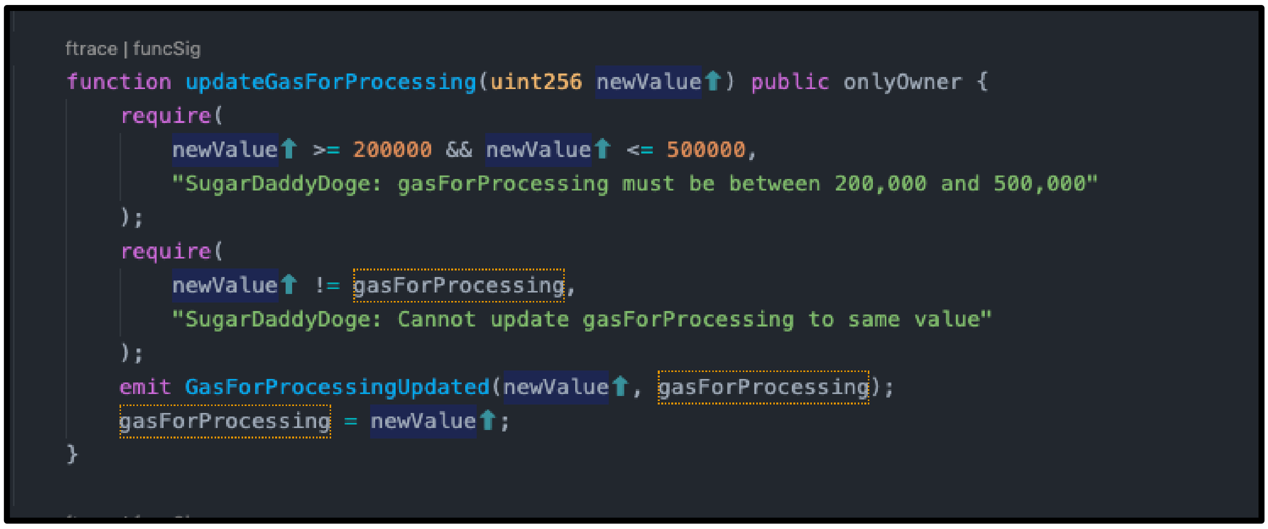
```
ftrace | funcSig
function updateUniswapV2Router(address newAddress) public onlyOwner {
    require(
        newAddress != address(uniswapV2Router),
        "SugarDaddyDoge: The router already has that address"
    );
    emit UpdateUniswapV2Router(newAddress, address(uniswapV2Router));
    uniswapV2Router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

❖ The owner can include/exclude accounts from fees.



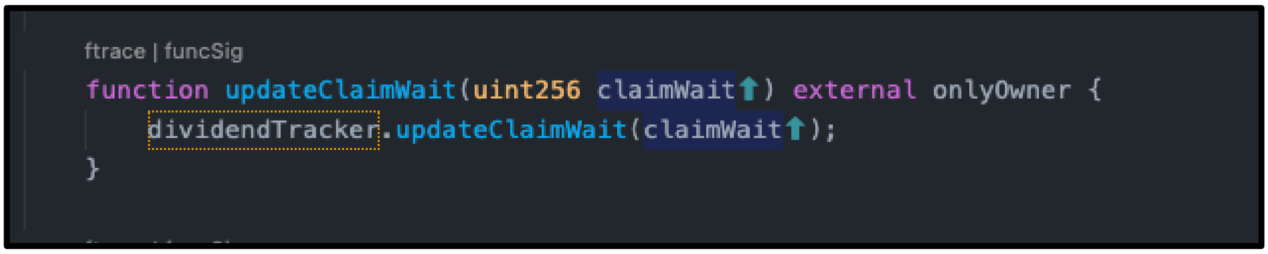
```
ftrace | funcSig
function excludeFromFees(address account, bool excluded) public onlyOwner {
    require(
        _isExcludedFromFees[account] != excluded,
        "SugarDaddyDoge: Account is already the value of 'excluded'"
    );
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}

ftrace | funcSig
function excludeMultipleAccountsFromFees(
    address[] calldata accounts,
    bool excluded
) public onlyOwner {
    for (uint256 i = 0; i < accounts.length; i++) {
        _isExcludedFromFees[accounts[i]] = excluded;
    }

    emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

❖ The owner can change the gas fee up to minimum 200,000 and max 500,000.



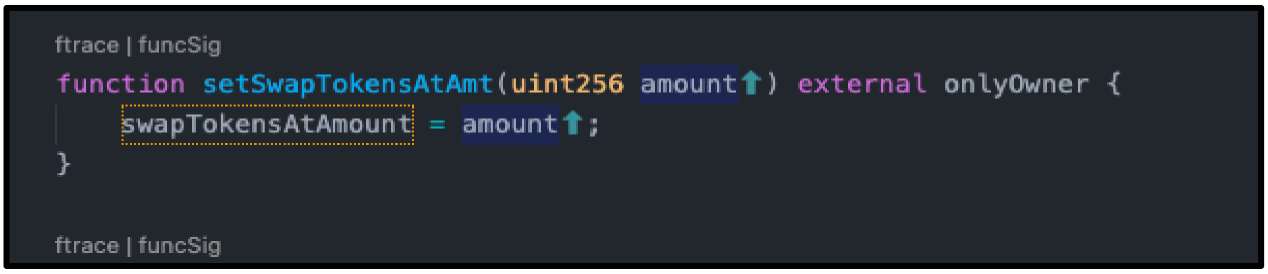
```
ftrace | funcSig
function updateGasForProcessing(uint256 newValue) public onlyOwner {
    require(
        newValue >= 200000 && newValue <= 500000,
        "SugarDaddyDoge: gasForProcessing must be between 200,000 and 500,000"
    );
    require(
        newValue != gasForProcessing,
        "SugarDaddyDoge: Cannot update gasForProcessing to same value"
    );
    emit GasForProcessingUpdated(newValue, gasForProcessing);
    gasForProcessing = newValue;
}
```

❖ The owner can change the claim wait time.



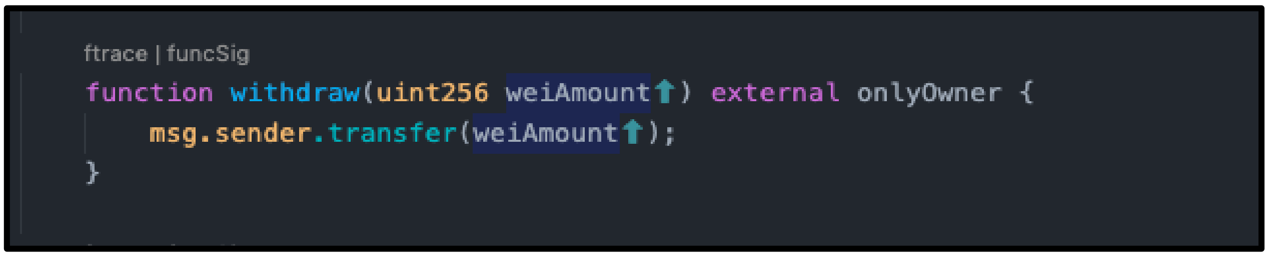
```
ftrace | funcSig
function updateClaimWait(uint256 claimWait) external onlyOwner {
    dividendTracker.updateClaimWait(claimWait);
}
```

❖ The owner can change the token swap limit to send BNBs to the liquidity pool.



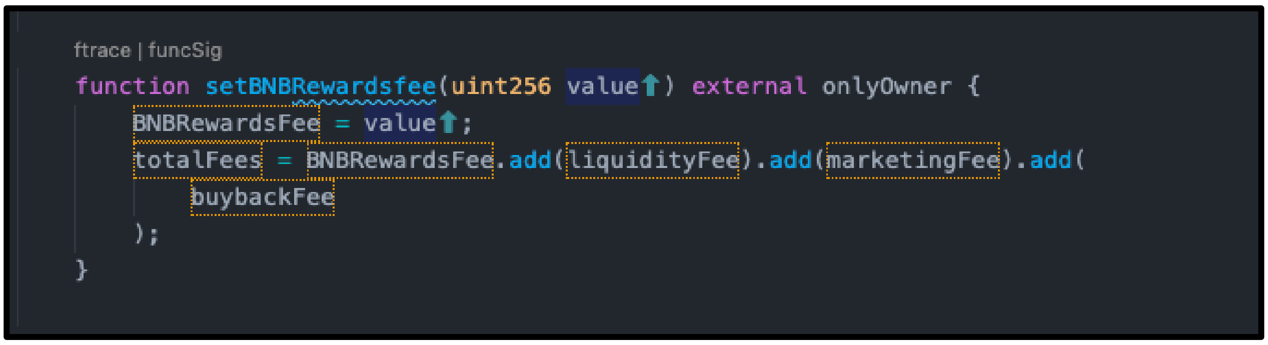
```
ftrace | funcSig
function setSwapTokensAtAmt(uint256 amount) external onlyOwner {
    swapTokensAtAmount = amount;
}

ftrace | funcSig
```

❖ The owner can withdraw BNB from the contract.



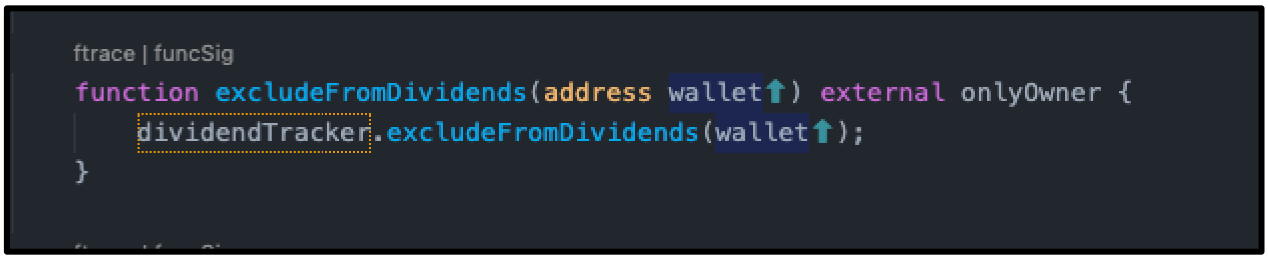
```
ftrace | funcSig
function withdraw(uint256 weiAmount) external onlyOwner {
    msg.sender.transfer(weiAmount);
}
```

❖ The owner can change the BNB reward fee.



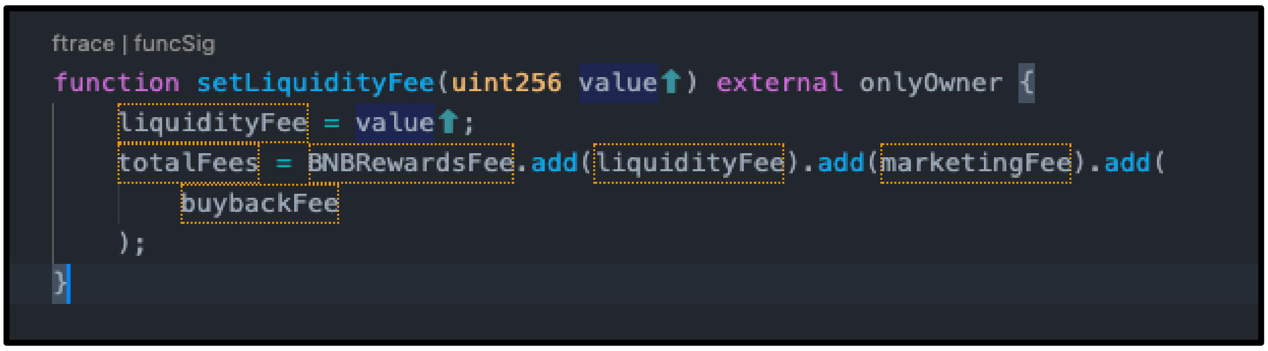
```
ftrace | funcSig
function setBNBRewardsfee(uint256 value) external onlyOwner {
    BNBRewardsFee = value;
    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee).add(
        buybackFee
    );
}
```

❖ The owner can exclude accounts from dividends.



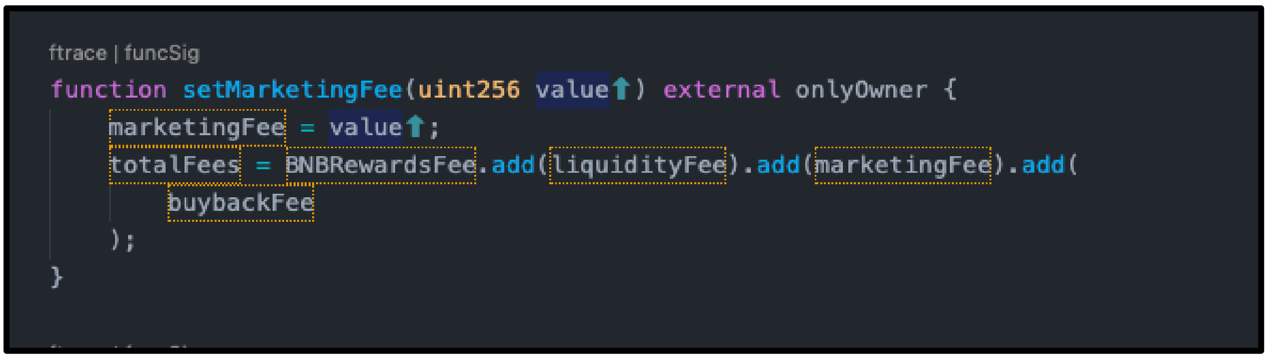
```
ftrace | funcSig
function excludeFromDividends(address wallet) external onlyOwner {
    dividendTracker.excludeFromDividends(wallet);
}
```

❖ The owner can change the liquidity fee.



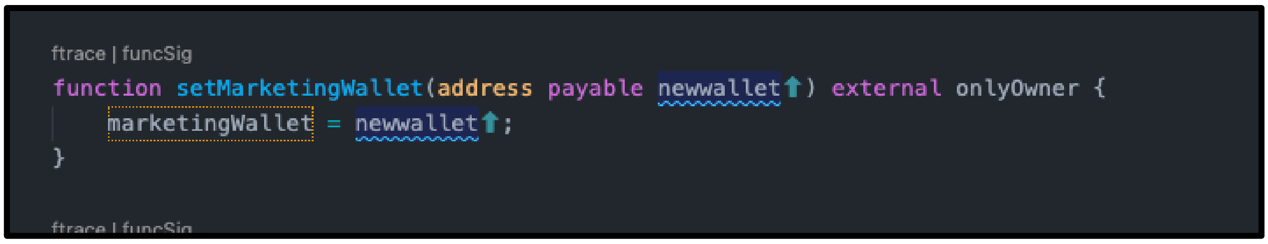
```
ftrace | funcSig
function setLiquidityFee(uint256 value) external onlyOwner {
    liquidityFee = value;
    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee).add(
        buybackFee
    );
}
```

❖ The owner can change the marketing fee.



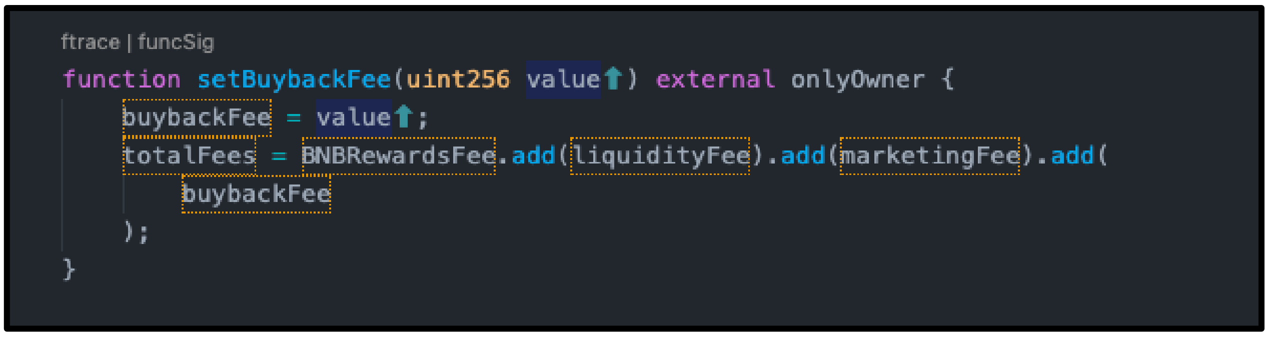
```
ftrace | funcSig
function setMarketingFee(uint256 value) external onlyOwner {
    marketingFee = value;
    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee).add(
        buybackFee
    );
}
```

❖ The owner can change the marketing wallet address.



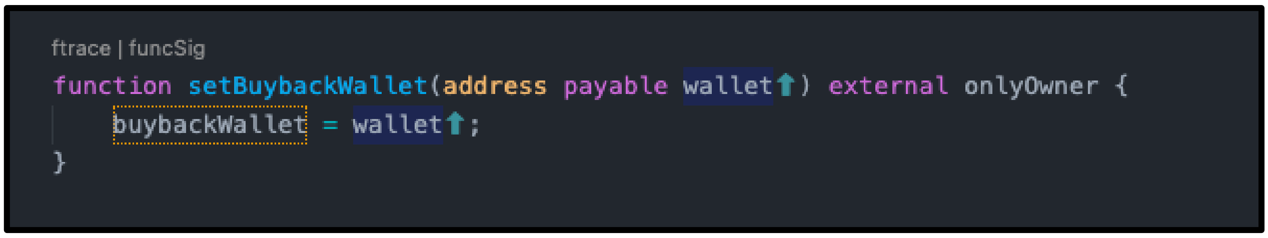
```
ftrace | funcSig
function setMarketingWallet(address payable newwallet) external onlyOwner {
    marketingWallet = newwallet;
}

ftrace | funcSig
```

❖ The owner can change the buy back fee.



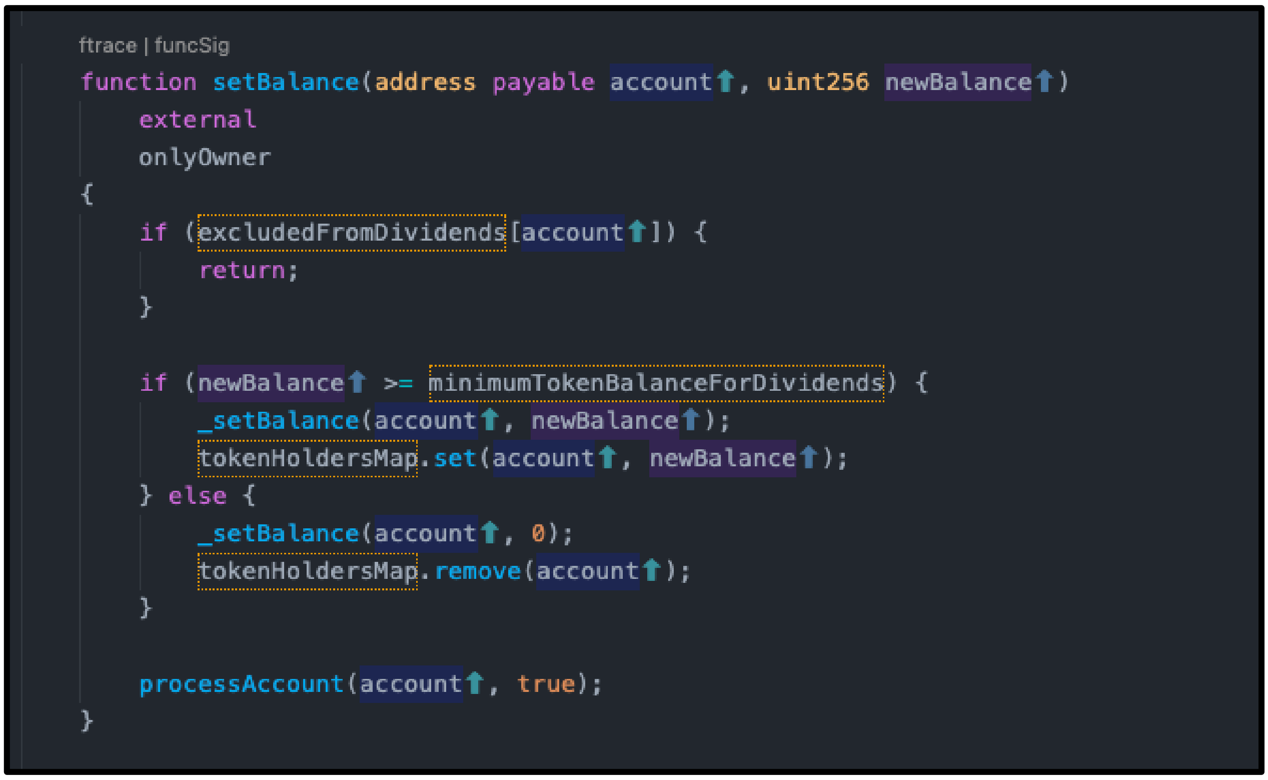
```
ftrace | funcSig
function setBuybackFee(uint256 value) external onlyOwner {
    buybackFee = value;
    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee).add(
        buybackFee
    );
}
```

❖ The owner can change the buy back wallet address.



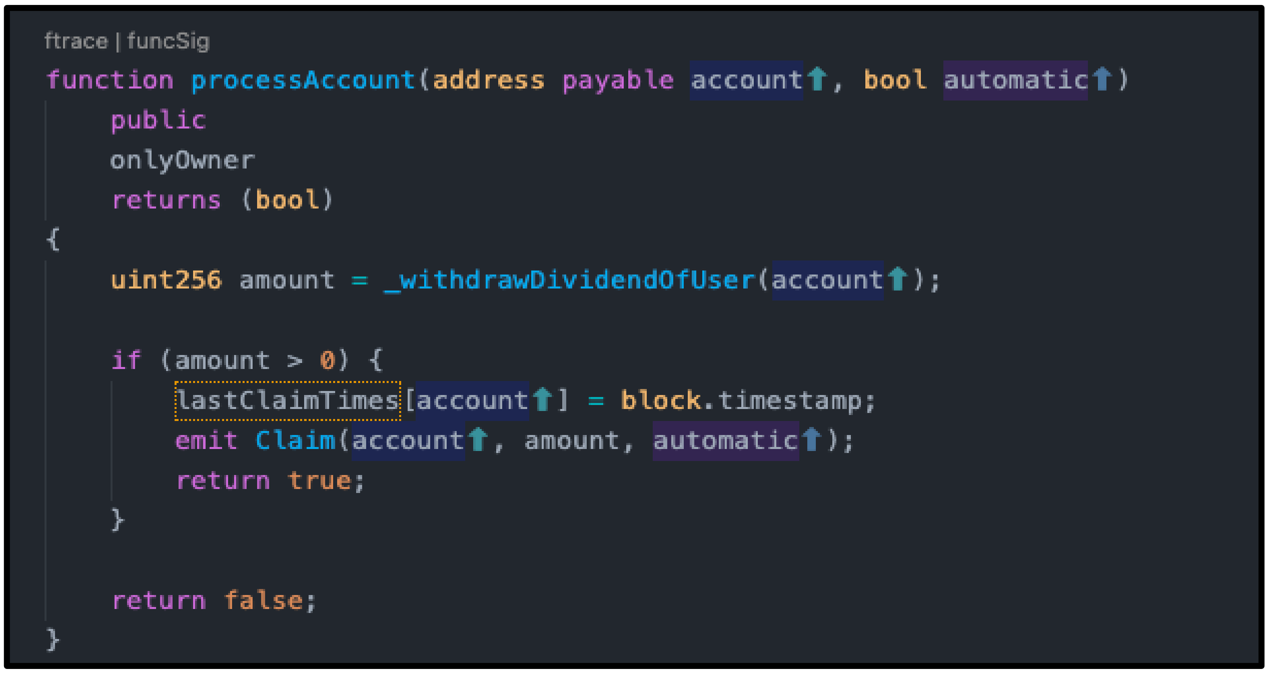
```
ftrace | funcSig
function setBuybackWallet(address payable wallet) external onlyOwner {
    buybackWallet = wallet;
}
```

❖ The owner can change the dividend receiving percentage on wallets by changing the wallet account balance, manually.



```
ftrace | funcSig
function setBalance(address payable account, uint256 newBalance)
    external
    onlyOwner
{
    if (excludedFromDividends[account]) {
        return;
    }

    if (newBalance >= minimumTokenBalanceForDividends) {
        _setBalance(account, newBalance);
        tokenHoldersMap.set(account, newBalance);
    } else {
        _setBalance(account, 0);
        tokenHoldersMap.remove(account);
    }

    processAccount(account, true);
}
```

❖ The owner can manually process dividends by selecting the wallets.



```
ftrace | funcSig
function processAccount(address payable account, bool automatic)
    public
    onlyOwner
    returns (bool)
{
    uint256 amount = _withdrawDividendOfUser(account);

    if (amount > 0) {
        lastClaimTimes[account] = block.timestamp;
        emit Claim(account, amount, automatic);
        return true;
    }

    return false;
}
```

Audit conclusion

While conducting the audit of the SugarDaddyDoge token smart contract, it was observed that there is nothing alamaring with the code and the contract contains only a low severity issue.