



RugFreeCoins Audit



BitDogeCoin Token Audit
Smart Contract Security Audit
August 29, 2021

Contents

Audit details	1
Disclaimer	2
Background	3
About the project	4
Target market and the concept	8
Potential to grow with score points	10
Total Points	10
Contract details	11
Top token holders	12
Token distribution	13
Contract interaction details	14
Contract code function details	15
Contract description table	16
Security issue checking status	21
Owner privileges	22
Audit conclusion	28

Audit details



Audited project

BitDogeCoin Token



Contract Address

0xe81bfbb53a195fa3b6d1b12c4c1ea5452490bb8c



Client contact

BitDogeCoin Token Team



Blockchain

Binance smart chain



Project website

<https://www.bitdogecoin.org/>

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 BitDogeCoin to perform an audit of the smart contract.

<https://bscscan.com/address/0xe81bfbb53a195fa3b6d1b12c4c1ea5452490bb8c#code>

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

BitDogeCoin is a token built on the Binance Smart Chain. Each transaction, purchase incur a 16% fee, and sales incur an 32% fee. The BitDoge Token launched on 27th August 2021 on PancakeSwap.

Features

- ❖ BitDoge improves with a new suite of innovations that will help increase returns for investors, which will be exchanging 3% of the tax when buying and 6% of tax when selling for BNB buys back from the supply every minute and burn all tokens bought automatically.

When the BuyBack Strikes the total sell fee is increased to 48% and gradually reverts to normal ensuring that no one sells when the price is up.

- ❖ Reward mechanism that investors can accumulate BTC by just holding as the smart contract automatically distributes 7% when buying and 14% when selling from every transaction tax amongst holders.
- ❖ The **sustainability fee of 5% marketing when buying and 10% when selling** is what allows Bitdoge to hold the aforementioned promise. Tokens will be swapped into BNBs and will be sent to a marketing wallet per transaction. This way, BitDoge will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.
- ❖ The liquidity fee of 1% when buying and 2% when selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity. This is a key element for decentralized exchanges like Pancakeswap.

Tokenomics

16% fee when buying

- ❖ 7% of trade goes to holders' pockets in BTC.
- ❖ 5% of trade goes to marketing.
- ❖ 3% of trade goes to buyback & burn
- ❖ 1% of trade goes to the liquidity pool.

32% fee when selling

- ❖ 14% of trade goes to holders' pockets in BTC.
- ❖ 10% of trade goes to marketing.
- ❖ 6% of trade goes to buyback & burn
- ❖ 2% of trade goes to the liquidity pool.

Roadmap

Phase 1

- ❖ Website Launch
- ❖ Poocoin and other big website banner ads
- ❖ 5000 Telegram Members
- ❖ FairLaunch on PancakeSwap
- ❖ BitApp Release (Track your BTC earnings)
- ❖ Coingecko listing
- ❖ Blockfolio Application
- ❖ Influencer marketing push
- ❖ \$1M Market Cap
- ❖ BitGames realese (Arcade play-to earn games)

Phase 2

- ❖ +20k members in Telegram group
- ❖ +5k followers in Twitter
- ❖ 10,000 holders
- ❖ BitSwap (Swap, Charting, Sniping orders and more !)
- ❖ Partnership with influencer Youtube, Twitter, & TikTok
- ❖ CoinmarketCap listing
- ❖ Massive Marketing
- ❖ \$25M Market Cap
- ❖ BitFarms (Auto Compounding Vaults)
- ❖ \$50M Market Cap
- ❖ \$100M Market Cap

Phase 3

- ❖ BitMarket (NFTs with real value and Marketplace with bids)
- ❖ BitDoge Merch Shop
- ❖ Marketing: BitDoge goes to mainstream
- ❖ Listing on CEXes
- ❖ \$500M MarketCap
- ❖ Charity donations
- ❖ Research donations
- ❖ \$1B MarketCap
- ❖ More to be announced
- ❖ Research donation
- ❖ More to be announced

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 in BTC by holding tokens.
- ❖ Anyone who's interested in trading tokens.
- ❖ All Bitcoin investors and fans out there.
- ❖ Anyone who's interested in stake BTC or Bitdoge and earn rewards.
- ❖ Anyone who's interested in taking part with the future games that's going to be built by the BitDoge team.
- ❖ Anyone who's interested in staking cryptos and earning passive income.
- ❖ Anyone who's interested in taking part with NFT marketplace (BITMARKET) to buy and sell cryptos.
- ❖ Anyone who's interested in making financial transactions with any other party using BitDoge or BTC as the currency.

Core concept

The BTC reward system

7% of fee when buying and 14% of fee when selling gets converted to BTC, and is split amongst all holders. The rewards are sent to holders that have at least 1\$ worth BTC, holders will be eligible to receive tokens every 30 mins and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

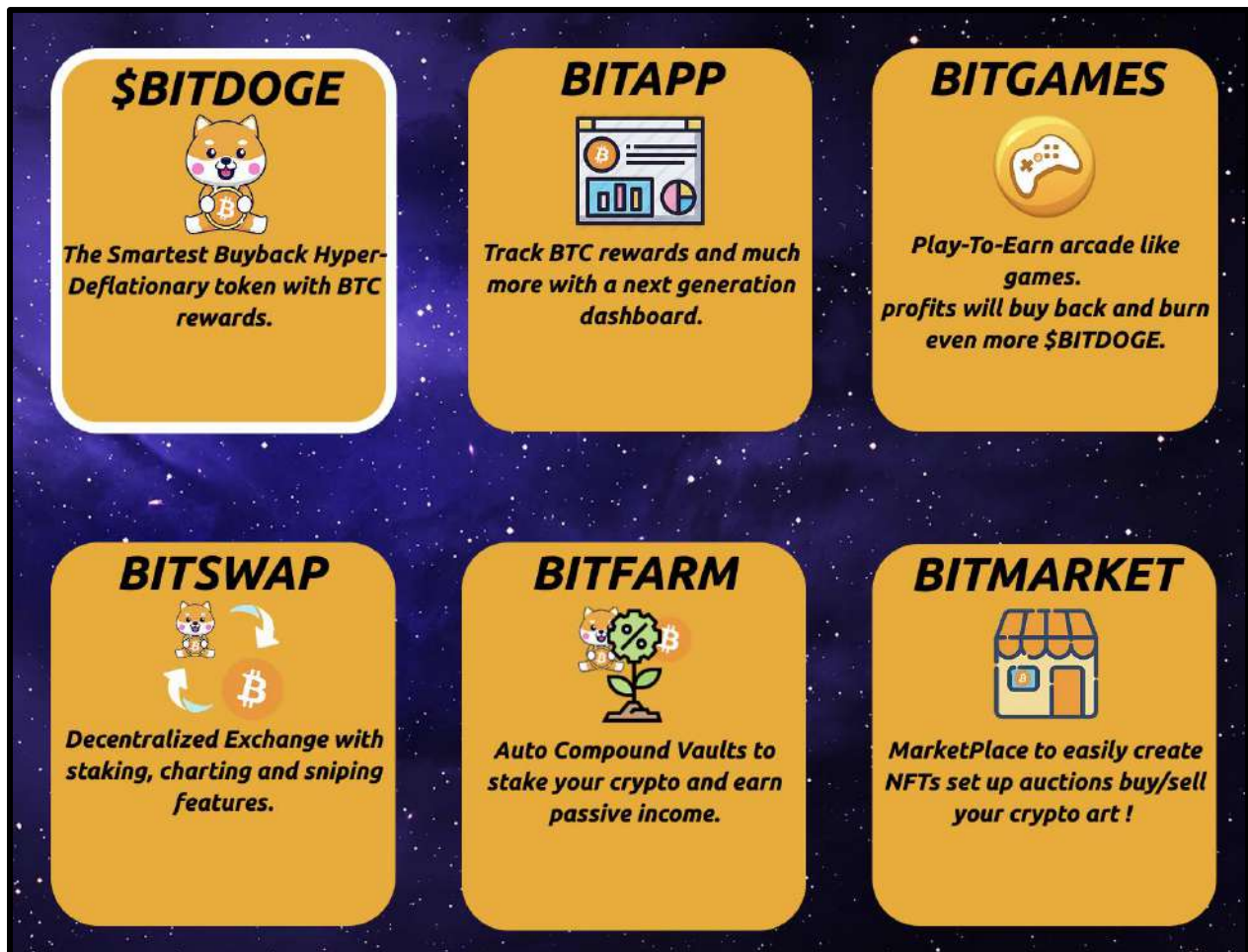
The buyback and burn mechanism collects 3% tax when buying and 6% when selling on each transaction, which is stored inside the contract. Whenever a buy or sell occurs, a fraction of the buyback amount is used to automatically purchase tokens from the liquidity pool. Those tokens are immediately burned after purchase, which keeps the token price stable.

Also, when the BuyBack Strikes the total sell fee is increased to 48% and gradually reverts to normal ensuring that no one sells when the price is up.

The **fee of 5% when buying and 10% when selling for marketing** is what allows BitDoge to promote the token and use funds to further development of the platform. Tokens will be swapped into BNB and will be sent to a marketing wallet per transaction. This way, BitDoge 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% when buying and 2% when selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

The future plan



Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	8/10
3	Information quality	9/10
4	Service quality	9/10
5	System quality	8/10
6	Impact on the community	9/10
7	Impact on the business	9/10
8	Preparing for the future	8/10
Total Points		8.63/10

Contract details

Token contract details for 29th August 2021

Contract name	BitDoge Coin
Contract address	0xe81bfbb53a195fa3b6d1b12c4c1ea5452490bb8c
Token supply	1,000,000,000,000,000
Token ticker	BITDOGE
Decimals	9
Token holders	73
Transaction count	862
Top 100% holders dominance	100%
Auto liquidity receiver	0xebadd1f2935008c35710bd76571087c7c11b6466
Marketing wallet address	0xebadd1f2935008c35710bd76571087c7c11b6466
Contract deployer address	0xEbAdD1F2935008c35710Bd76571087C7c11B6466
Contract's current owner address	0xebadd1f2935008c35710bd76571087c7c11b6466

Top token holders

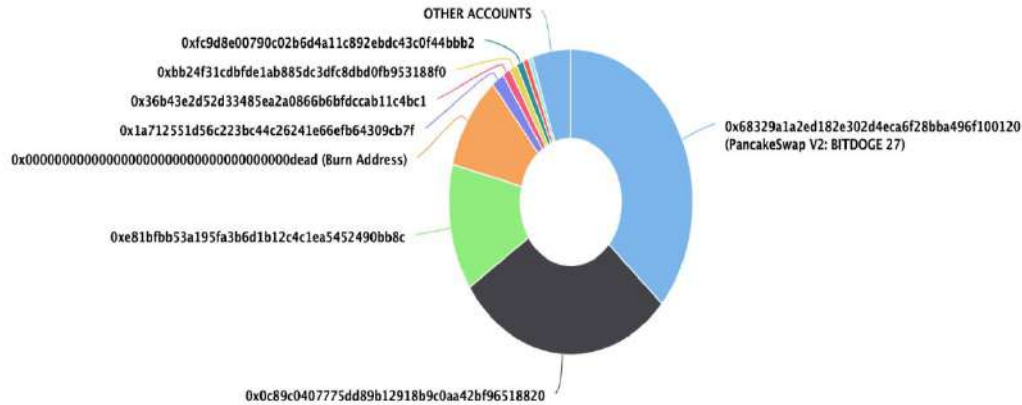
Top 10 Token Holders

The top 10 holders collectively own 94.89% (948,899,052,385,819.00 Tokens) of BitDoge Coin

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 72

BitDoge Coin Top 10 Token Holders

Source: BscScan.com



(A total of 948,899,052,385,819.00 tokens held by the top 10 accounts from the total supply of 1,000,000,000,000,000.00 token)

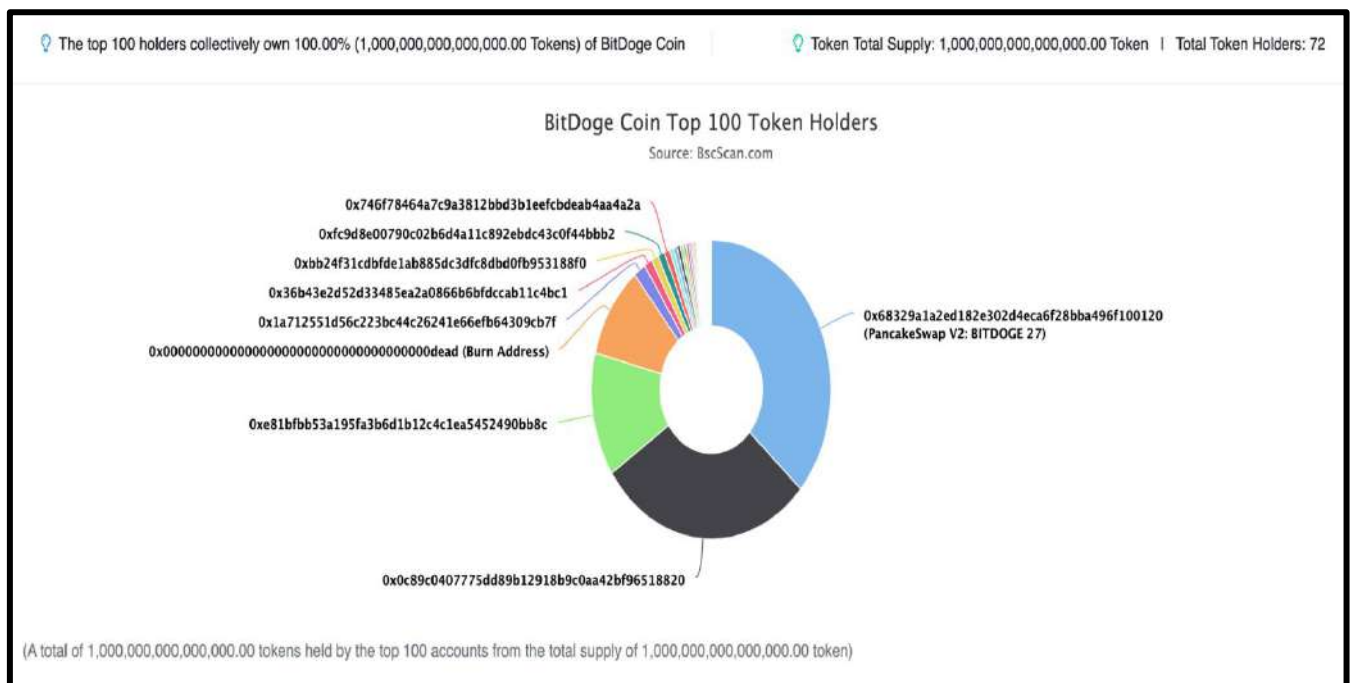
Rank	Address	Quantity (Token)	Percentage
1	PancakeSwap V2: BITDOGE 27	365,906,105,053,889.337288476	36.5906%
2	0x0c89c0407775dd89b12918b9c0aa42bf96518820	290,000,000,000,000	29.0000%
3	0xe81bfbb53a195fa3b6d1b12c4c1ea5452490bb8c	133,718,438,932,736.078034548	13.3718%
4	Burn Address	100,000,000,000,000	10.0000%
5	0x1a712551d56c223bc44c26241e66efb64309cb7f	16,558,395,910,966.246843675	1.6558%
6	0x36b43e2d52d33485ea2a0866b6bdfccab11c4bc1	10,876,834,014,707.690026009	1.0877%
7	0xbb24f31c9bfe1ab885dc3dfc8dbd0fb953188f0	9,736,982,950,238.717429062	0.9737%
8	0xfc9d8e00790c02b6d4a11c892ebdc43c0f44bbb2	9,368,790,286,859.869592444	0.9369%
9	0x746f78464a7c9a3812bbd3b1eefcbdeab4aa4a2a	7,120,983,677,833.154878598	0.7121%
10	0x29d8c463ed64f92a6930d1b3e54a4866812d9a90	5,612,521,558,587.635223233	0.5613%

Token distribution

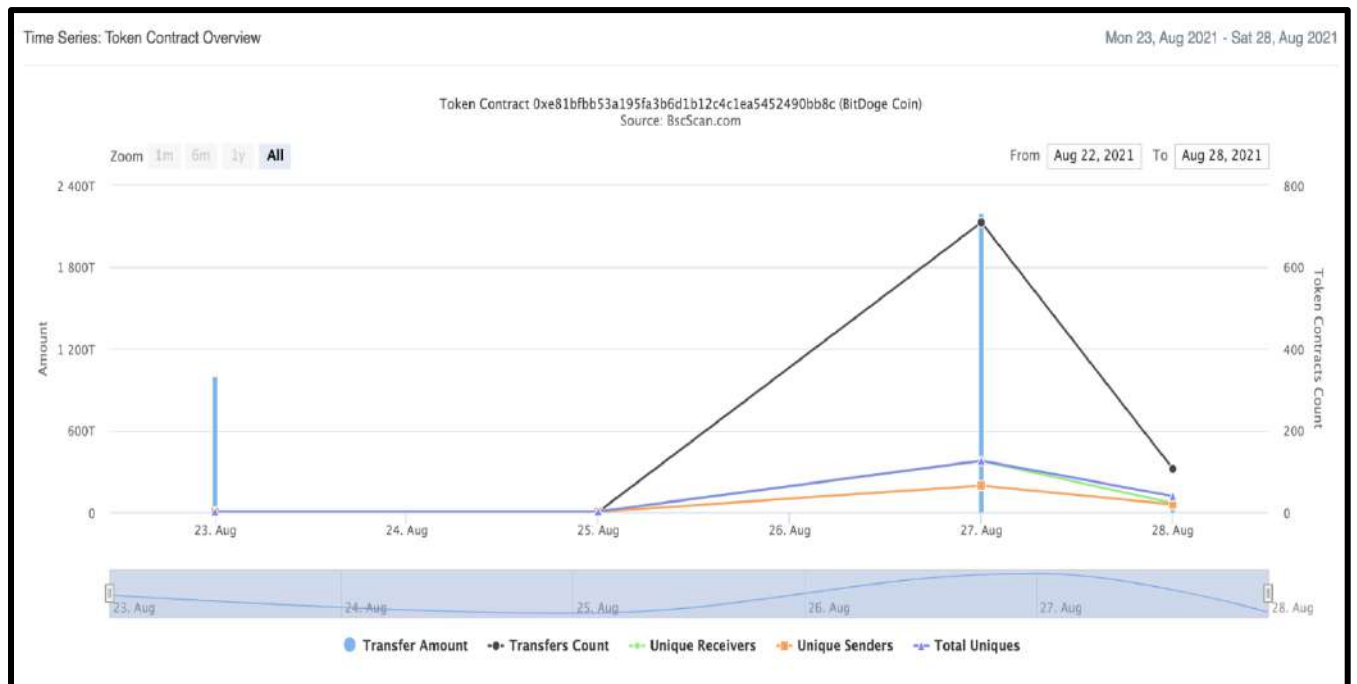
Tokens are distributed as follows:

TOTAL SUPPLY 1 QUADRILLION (100%)	LIQUIDITY 600 TRILLION (60%)
INITIAL BURN 100 TRILLION (10%)	AIRDROP 200 TRILLION (20%)
MARKETING 100 TRILLION (10%)	NO TEAM WALLET

Top 100 Token Holders



Contract interaction details
















































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	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
SafeMath	Library			
L	add	Internal 		
L	sub	Internal 		
L	sub	Internal 		
L	mul	Internal 		
L	div	Internal 		
L	div	Internal 		
IBEP20	Interface			
L	totalSupply	External 		NO 
L	decimals	External 		NO 
L	symbol	External 		NO 
L	name	External 		NO 
L	getOwner	External 		NO 
L	balanceOf	External 		NO 
L	transfer	External 		NO 
L	allowance	External 		NO 
L	approve	External 		NO 
L	transferFrom	External 		NO 
IDEXFactory	Interface			
L	createPair	External 		NO 

IDEXRouter	Interface			
L	factory	External ¶		NO¶
L	WETH	External ¶		NO¶
L	addLiquidity	External ¶		NO¶
L	addLiquidityETH	External ¶		NO¶
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ¶		NO¶
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ¶		NO¶
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ¶		NO¶
BitDogeAuth	Implementation			
L		Public ¶		NO¶
L	authorizeFor	Public ¶		authorizedFor
L	authorizeForMultiplePermissions	Public ¶		authorizedFor
L	unauthorizeFor	Public ¶		authorizedFor
L	unauthorizeForMultiplePermissions	Public ¶		authorizedFor
L	isOwner	Public ¶		NO¶
L	isAuthorizedFor	Public ¶		NO¶
L	isAuthorizedFor	Public ¶		NO¶
L	transferOwnership	Public ¶		onlyOwner
L	getPermissionNameToIndex	Public ¶		NO¶
L	getPermissionUnlockTime	Public ¶		NO¶
L	isLocked	Public ¶		NO¶
L	lockPermission	Public ¶		authorizedFor
L	unlockPermission	Public ¶		NO¶

IDividendBit_Doge	Interface			
L	setDistributionCriteria	External 		NO 
L	setShare	External 		NO 
L	deposit	External 		NO 
L	process	External 		NO 
L	claimDividend	External 		NO 
DividendBit_Doge	Implementation	IDividendBit_Doge		
L		Public 		NO 
L	setDistributionCriteria	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	getCumulativeDividends	Internal 		
L	addShareholder	Internal 		
L	removeShareholder	Internal 		
BitDoge	Implementation	IBEP20, BitDogeAuth		
L		Public 		BitDogeAuth
L		External 		NO 
L	totalSupply	External 		NO 
L	decimals	External 		NO 
L	symbol	External 		NO 
L	name	External 		NO 

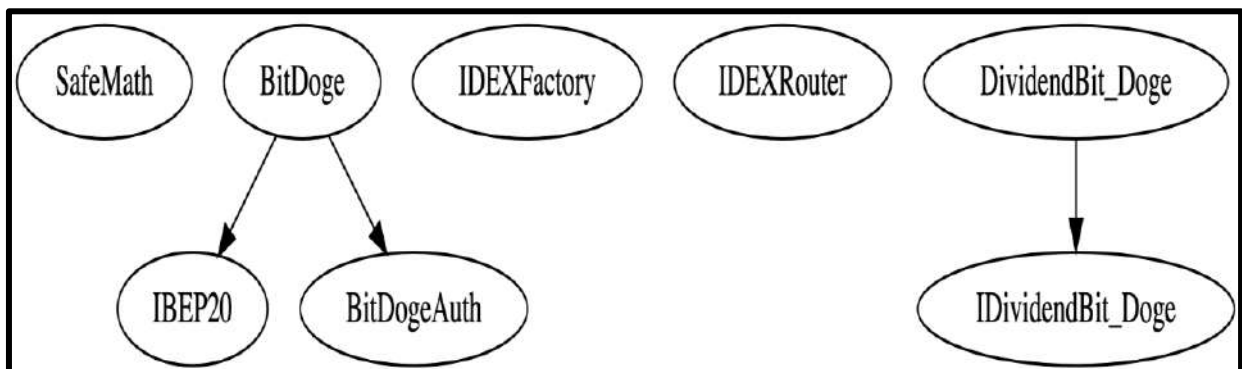
L	getOwner	External 		NO 
L	balanceOf	Public 		NO 
L	allowance	External 		NO 
L	approve	Public 		NO 
L	approveMax	External 		NO 
L	transfer	External 		NO 
L	transferFrom	External 		NO 
L	_transferFrom	Internal 		
L	_basicTransfer	Internal 		
L	checkTxLimit	Internal 		
L	shouldTakeFee	Internal 		
L	getTotalFee	Public 		NO 
L	getMultipliedFee	Public 		NO 
L	takeFee	Internal 		
L	isSell	Internal 		
L	shouldSwapBack	Internal 		
L	swapBack	Internal 		swapping
L	triggerBuyback	External 		authorizedFor
L	clearBuybackMultiplier	External 		authorizedFor
L	buyTokens	Internal 		swapping
L	setBuybackMultiplierSettings	External 		authorizedFor
L	launched	Internal 		
L	launch	Internal 		
L	setTxLimit	External 		authorizedFor
L	setIsDividendExempt	External 		authorizedFor
L	setIsFeeExempt	External 		authorizedFor
L	setIsTxLimitExempt	External 		authorizedFor
L	setFees	External 		authorizedFor

L	setFeeReceivers	External ¶	⬢	authorizedFor
L	setSwapBackSettings	External ¶	⬢	authorizedFor
L	setTargetLiquidity	External ¶	⬢	authorizedFor
L	setDistributionCriteria	External ¶	⬢	authorizedFor
L	setDistributorSettings	External ¶	⬢	authorizedFor
L	getCirculatingSupply	Public ¶		NO ¶
L	getLiquidityBacking	Public ¶		NO ¶
L	isOverLiquified	Public ¶		NO ¶
L	claimDividend	External ¶	⬢	NO ¶
L	addPair	External ¶	⬢	authorizedFor
L	removeLastPair	External ¶	⬢	authorizedFor
L	setFeesOnNormalTransfers	External ¶	⬢	authorizedFor
L	setIsBlacklisted	External ¶	⬢	authorizedFor
L	setLaunchedAt	External ¶	⬢	authorizedFor

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

- The Owner can change tax fees without any max limit, they have added validation with feeDenominator but the owner can change feeDenominator too. So still the owner can change fees without any limit.

```
ftrace | funcSig
function setFees(
    uint256 _liquidityFee↑,
    uint256 _buybackFee↑,
    uint256 _reflectionFee↑,
    uint256 _marketingFee↑,
    uint256 _feeDenominator↑,
    uint256 _totalSellFee↑
) external authorizedFor(Permission.AdjustContractVariables) {
    liquidityFee = _liquidityFee↑;
    buybackFee = _buybackFee↑;
    reflectionFee = _reflectionFee↑;
    marketingFee = _marketingFee↑;
    totalBuyFee = _liquidityFee↑.add(_buybackFee↑).add(_reflectionFee↑).add(
        _marketingFee↑
    );
    feeDenominator = _feeDenominator↑;
    totalSellFee = _totalSellFee↑;
    require(totalBuyFee <= feeDenominator / 10, "Buy fee too high");
    require(totalSellFee <= feeDenominator / 5, "Sell fee too high");
}
```


Owner privileges

- ❖ The owner can change buy back multiplier settings.

```
ftrace | funcSig
function setBuybackMultiplierSettings(
    uint256 numerator↑,
    uint256 denominator↑,
    uint256 length↑
) external authorizedFor(Permission.AdjustContractVariables) {
    require(numerator↑ / denominator↑ <= 3 && numerator↑ > denominator↑);
    buybackMultiplierNumerator = numerator↑;
    buybackMultiplierDenominator = denominator↑;
    buybackMultiplierLength = length↑;
}
```

- ❖ The owner can change the max transaction limit.

```
ftrace | funcSig
function setTxLimit(uint256 amount↑)
    external
    authorizedFor(Permission.AdjustContractVariables)
{
    require(amount↑ >= totalSupply / 2000);
    maxTxAmount = amount↑;
}
```

- ❖ The owner can exempt wallets from dividends.

```
ftrace | funcSig
function setIsDividendExempt(address holder↑, bool exempt↑)
    external
    authorizedFor(Permission.ExcludeInclude)
{
    require(holder↑ != address(this) && holder↑ != pancakeV2BNBPair);
    isDividendExempt[holder↑] = exempt↑;
    if (exempt↑) {
        distributor.setShare(holder↑, 0);
    } else {
        distributor.setShare(holder↑, balances[holder↑]);
    }
}
```

- ❖ The owner can exempt wallets from fees.

```
ftrace | funcSig
function setIsFeeExempt(address holder↑, bool exempt↑)
    external
    authorizedFor(Permission.ExcludeInclude)
{
    isFeeExempt[holder↑] = exempt↑;
}
```

- ❖ The owner can exempt wallets from max transaction fees.

```
ftrace | funcSig
function setIsTxLimitExempt(address holder↑, bool exempt↑)
    external
    authorizedFor(Permission.ExcludeInclude)
{
    isTxLimitExempt[holder↑] = exempt↑;
}
```

- ❖ The owner can change all fees.

```
ftrace | funcSig
function setFees(
    uint256 _liquidityFee↑,
    uint256 _buybackFee↑,
    uint256 _reflectionFee↑,
    uint256 _marketingFee↑,
    uint256 _feeDenominator↑,
    uint256 _totalSellFee↑
) external authorizedFor(Permission.AdjustContractVariables) {
    liquidityFee = _liquidityFee↑;
    buybackFee = _buybackFee↑;
    reflectionFee = _reflectionFee↑;
    marketingFee = _marketingFee↑;
    totalBuyFee = _liquidityFee↑.add(_buybackFee↑).add(_reflectionFee↑).add(
        _marketingFee↑
    );
    feeDenominator = _feeDenominator↑;
    totalSellFee = _totalSellFee↑;
    require(totalBuyFee <= feeDenominator / 10, "Buy fee too high");
    require(totalSellFee <= feeDenominator / 5, "Sell fee too high");
}
```

- ❖ The owner can change marketing and liquidity wallets.

```
ftrace | funcSig
function setFeeReceivers(
    address _autoLiquidityReceiver↑,
    address _marketingFeeReceiver↑
) external authorizedFor(Permission.AdjustContractVariables) {
    autoLiquidityReceiver = _autoLiquidityReceiver↑;
    marketingFeeReceiver = _marketingFeeReceiver↑;
}
ftrace | funcSig
```

- ❖ The owner can change swap back settings.

```
ftrace | funcSig
function setSwapBackSettings(bool _enabled↑, uint256 _amount↑)
    external
    authorizedFor(Permission.AdjustContractVariables)
{
    swapEnabled = _enabled↑;
    swapThreshold = _amount↑;
}
ftrace | funcSig
```

- ❖ The owner can change the target liquidity amount.

```
ftrace | funcSig
function setTargetLiquidity(uint256 _target↑, uint256 _denominator↑)
    external
    authorizedFor(Permission.AdjustContractVariables)
{
    targetLiquidity = _target↑;
    targetLiquidityDenominator = _denominator↑;
}
ftrace | funcSig
```

- ❖ The owner can change the minimum distribution amount and time.

```
ftrace | funcSig
function setDistributionCriteria(
    uint256 _minPeriod↑,
    uint256 _minDistribution↑
) external authorizedFor(Permission.AdjustContractVariables) {
    distributor.setDistributionCriteria(_minPeriod↑, _minDistribution↑);
}
ftrace | funcSig
```

- ❖ The owner can change the distribution gas fee maximum up to 1000000.

```
ftrace | funcSig
function setDistributorSettings(uint256 gas↑)
    external
    authorizedFor(Permission.AdjustContractVariables)
{
    require(gas↑ <= 1000000);
    distributorGas = gas↑;
}
```

- ❖ The owner can enable/disable fees on normal transactions.

```
ftrace | funcSig
function setFeesOnNormalTransfers(bool _enabled↑)
    external
    authorizedFor(Permission.AdjustContractVariables)
{
    feesOnNormalTransfers = _enabled↑;
}
```

- ❖ The owner can blacklist wallets.

```
ftrace | funcSig
function setIsBlacklisted(address adr↑, bool blacklisted↑)
    external
    authorizedFor(Permission.Blacklist)
{
    isBlacklisted[adr↑] = blacklisted↑;
}
```

- ❖ The owner can change launch time.

```
ftrace | funcSig
function setLaunchedAt(uint256 launched_↑)
    external
    authorizedFor(Permission.AdjustContractVariables)
{
    launchedAt = launched_↑;
}
```

- ❖ The owner can add permissions for authorized persons.

```
ftrace | funcSig
function authorizeFor(address adr↑, string memory permissionName↑)
    public
    authorizedFor(Permission.Authorize)
{
    uint256 permIndex = permissionNameToIndex[permissionName↑];
    authorizations[adr↑][permIndex] = true;
    emit AuthorizedFor(adr↑, permissionName↑, permIndex);
}

ftrace | funcSig
function authorizeForMultiplePermissions(
    address adr↑,
    string[] calldata permissionNames↑
) public authorizedFor(Permission.Authorize) {
    for (uint256 i; i < permissionNames↑.length; i++) {
        uint256 permIndex = permissionNameToIndex[permissionNames↑[i]];
        authorizations[adr↑][permIndex] = true;
        emit AuthorizedFor(adr↑, permissionNames↑[i], permIndex);
    }
}
```


- ❖ The owner can remove permissions for authorized persons.

```
ftrace | funcSig
function unauthorizedFor(address adr↑, string memory permissionName↑)
    public
    authorizedFor(Permission.Unauthorize)
{
    require(adr↑ != owner, "Can't unauthorize owner");

    uint256 permIndex = permissionNameToIndex[permissionName↑];
    authorizations[adr↑][permIndex] = false;
    emit UnauthorizedFor(adr↑, permissionName↑, permIndex);
}

ftrace | funcSig
function unauthorizedForMultiplePermissions(
    address adr↑,
    string[] calldata permissionNames↑
) public authorizedFor(Permission.Unauthorize) {
    require(adr↑ != owner, "Can't unauthorize owner");

    for (uint256 i; i < permissionNames↑.length; i++) {
        uint256 permIndex = permissionNameToIndex[permissionNames↑[i]];
        authorizations[adr↑][permIndex] = false;
        emit UnauthorizedFor(adr↑, permissionNames↑[i], permIndex);
    }
}
```

- ❖ The owner can transfer ownership.

```
ftrace | funcSig
function transferOwnership(address payable adr↑) public onlyOwner {
    address oldOwner = owner;
    owner = adr↑;
    for (uint256 i; i < NUM_PERMISSIONS; i++) {
        authorizations[oldOwner][i] = false;
        authorizations[owner][i] = true;
    }
    emit OwnershipTransferred(oldOwner, owner);
}
```

- ❖ The owner can lock/unlock permissions from authorized persons.

```
ftrace | funcSig
function lockPermission(string memory permissionName↑, uint64 time↑)
    public
    virtual
    authorizedFor(Permission.LockPermissions)
{
    uint256 permIndex = permissionNameToIndex[permissionName↑];
    uint64 expiryTime = uint64(block.timestamp) + time↑;
    lockedPermissions[permIndex] = PermissionLock(true, expiryTime);
    emit PermissionLocked(permissionName↑, permIndex, expiryTime);
}

ftrace | funcSig
function unlockPermission(string memory permissionName↑) public virtual {
    require(
        block.timestamp > getPermissionUnlockTime(permissionName↑),
        "Permission is locked until the expiry time."
    );
    uint256 permIndex = permissionNameToIndex[permissionName↑];
    lockedPermissions[permIndex].isLocked = false;
    emit PermissionUnlocked(permissionName↑, permIndex);
}
```

Audit conclusion

While conducting the audit of the Bit Doge smart contract, it was observed that there is nothing alarming with the code and it only contains an informational concern.