Cyberscope

Audit Report

SLUMDOGz

July 2022

Type	BEP20
Network	BSC
Address	0x7df3a788adf3ace4f7fd272fa6933ba1f5edcbca
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	TOKEN
Compiler Version	v0.8.12+commit.f00d7308
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0x7df3a788adf3ace4f7fd272fa6933ba1f5edcbca
Symbol	SDT
Decimals	18
Total Supply	1,000,000,000
Domain	slumdogz-token.com

Source Files

Filename	**SHA256**
contract.sol	4c1912b9f165b175135ca43fbcebbbe76085cf02d94a0e3c6a3e4fe5cb84812a

Audit Updates

Initial Audit	13th July 2022
Corrected	

Contract Analysis

Critical Medium Minor Pass

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

ST - Stop Transactions

Criticality	critical
Location	contract.sol#L1009

Description

The contract owner has the authority to stop selling transactions for all users excluding the owner. The owner may take advantage of it by setting the selling taxing fees to the maximum amount and as a result the contract will become honeypot.

```
function _transferStandard(
        address sender,
        address recipient,
        uint256 tAmount
    ) private {
        (
            uint256 rAmount,
            uint256 rTransferAmount,
            uint256 rFee,
            uint256 tTransferAmount,
            uint256 tFee,
            uint256 tLiquidity,
            uint256 tDev
        ) = _getValues(tAmount);
```

Recommendation

The contract could embody a check for not allowing setting the selling taxing fees less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply.

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

OCTD - Owner Contract Tokens Drain

Criticality	minor
Location	contract.sol#L1067,L1074

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `withdrawStuckedFunds` and `withdrawStuckedTokens` function.

```
function withdrawStuckedFunds(uint256 amount) external onlyOwner {
    // This is the current recommended method to use.
    (bool sent, ) = _owner.call{value: amount}("");
    require(sent, "Failed to withdraw BNB");
}

function withdrawStuckedTokens(address tokenAddress, uint256 tokens) external onlyOwner
returns (bool success){
    return IERC20(tokenAddress).transfer(msg.sender, tokens);
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L1015,L1029

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the setTaxFeePercent function with a high percentage value.

```
function setSellFeePercent(
    uint256 tFee,
    uint256 lFee,
    uint256 mFee,
    uint256 dFee,
    uint256 bFee
) external onlyOwner {
    _sellTaxFee = tFee;
    _sellLiquidityFee = lFee;
    _sellMarketingFee = mFee;
    _sellDevFee = dFee;
    _sellBurnFee = bFee;
}

function setBuyFeePercent(
    uint256 tFee,
    uint256 lFee,
    uint256 mFee,
    uint256 dFee,
    uint256 bFee
) external onlyOwner {
    _buyTaxFee = tFee;
    _buyLiquidityFee = lFee;
    _buyMarketingFee = mFee;
    _buyDevFee = dFee;
    _buyBurnFee = bFee;
}
```

Recommendation

The contract could embody a check for the maximum acceptable value.

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	ZD	Zero Division
	STC	Succeeded Transfer Check
	CO	Code Optimization
	L01	Public Function could be Declared External
	L02	State Variables could be Declared Constant
	L04	Conformance to Solidity Naming Conventions
	L07	Missing Events Arithmetic
	L09	Dead Code Elimination
	L13	Divide before Multiply Operation

ZD - Zero Division

Criticality	critical
Location	contract.sol#L1330

Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert.

In multiple code segments there is division with _totalFees. Total fees can be set to zero due to the lack of check and as a result zero division will be provoked.

```
function swapAndLiquify(uint256 contractTokenBalance) private lockTheSwap {
        _totalFees = _marketingFee.add(_liquidityFee).add(_devFee).add(_burnFee);
        burnTokens = contractTokenBalance.div(_totalFees).mul(_burnFee);
```

Recommendation

The contract should prevent those variables to be set to zero or should not allow to execute the corresponding statements.

STC - Succeeded Transfer Check

Criticality	minor
Location	contract.sol#L1075

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
function withdrawStuckedTokens(address tokenAddress, uint256 tokens) external onlyOwner
returns (bool success){
    return IERC20(tokenAddress).transfer(msg.sender, tokens);
}
```

Recommendation

The contract should check if the result of the transfer methods is successful.

CO - Code Optimization

Criticality	minor
Location	contract.sol#L1298

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

This code segment can be optimized. The buy and transfer fees are the same.

```
if (from == uniswapV2Pair) {
                // Buy
                _taxFee = _buyTaxFee;
                _liquidityFee = _buyLiquidityFee;
                _marketingFee = _buyMarketingFee;
                _devFee = _buyDevFee;
                _burnFee = _buyBurnFee;

 } else if (to == uniswapV2Pair) {
                // Sell
                _taxFee = _sellTaxFee;
                _liquidityFee = _sellLiquidityFee;
                _marketingFee = _sellMarketingFee;
                _devFee = _sellDevFee;
                _burnFee = _sellBurnFee;

} else {
                // Transfer
                _taxFee = _buyTaxFee;
                _liquidityFee = _buyLiquidityFee;
                _marketingFee = _buyMarketingFee;
                _devFee = _buyDevFee;
                _burnFee = _buyBurnFee;
 }
```

Recommendation

Rewrite some code segments so the runtime will be more performant.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L815,885,999,318,921,949,837,819,855,828,909,323,807,811,846,341,872,901,905,333,995,1250

Description

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

```
isExcludedFromFee
excludeFromFee
lock
totalFees
isExcludedFromReward
increaseAllowance
unlock
approve
symbol
...
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L723,710,724

Description

Constant state variables should be declared constant to save gas.

```
_maxTxAmount
_burnAddress
_maxWalletBalance
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L1208,744,723,724,297,743,1212,417,448,741,415,1054,710,742,295,1003,708,706,709,1220,1007,494,740

Description

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

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

```
_taxFee
WETH
_addr
_amount
_devWalletAddress
_isBlacklisted
_marketingWalletAddress
_owner
_marketingFee
...
```

Recommendation

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

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L1011,1025,1040

Description

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

```
numTokensSellToAddToLiquidity = amount * 10 ** _decimals
_buyTaxFee = tFee
_sellTaxFee = tFee
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L171,198,264,252,212,229,274,163,190,242,183

Description

Functions that are not used in the contract, and make the code's size bigger.

```
functionCall
functionStaticCall
isContract
_verifyCallResult
functionCallWithValue
functionDelegateCall
sendValue
...
```

Recommendation

Remove unused functions.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L1324

Description

Performing divisions before multiplications may cause lose of prediction.

```
marketingFunds = newBalance.div(_totalFees).mul(_marketingFee)
halfFunds = newBalance.div(_totalFees).mul(_liquidityFee.div(2))
marketingTokens = contractTokenBalance.div(_totalFees).mul(_marketingFee)
devFunds = newBalance.div(_totalFees).mul(_devFee)
devTokens = contractTokenBalance.div(_totalFees).mul(_devFee)
burnTokens = contractTokenBalance.div(_totalFees).mul(_burnFee)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
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		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Address	Library			
	isContract	Internal		

	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	_verifyCallResult	Private		
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	lock	Public	✓	onlyOwner
	unlock	Public	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-

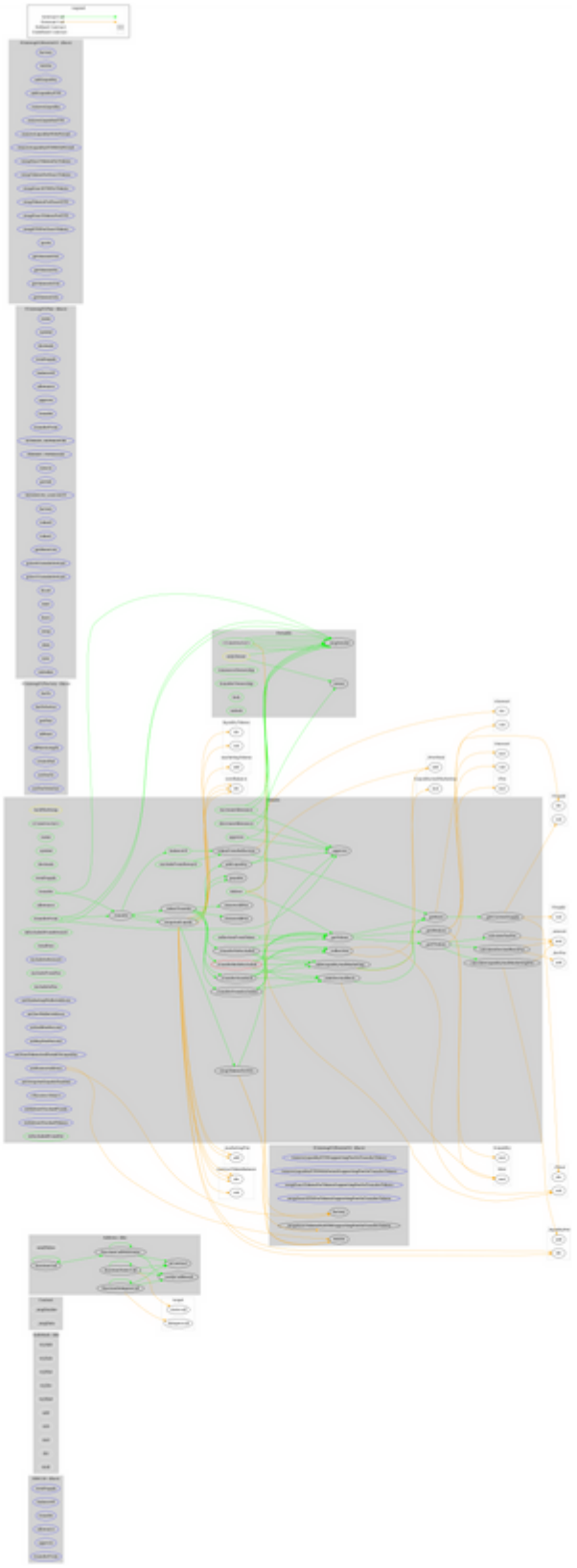
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-

	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
TOKEN	Implementation	Context, IERC20, Ownable		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-

	isExcludedFromReward	Public		-
	totalFees	Public		-
	deliver	Public	✓	-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-
	excludeFromReward	Public	✓	onlyOwner
	includeInReward	External	✓	onlyOwner
	_transferBothExcluded	Private	✓	
	excludeFromFee	Public	✓	onlyOwner
	includeInFee	Public	✓	onlyOwner
	setMarketingWalletAddress	External	✓	onlyOwner
	setDevWalletAddress	External	✓	onlyOwner
	setSellFeePercent	External	✓	onlyOwner
	setBuyFeePercent	External	✓	onlyOwner
	setNumTokensSellToAddToLiquidity	External	✓	onlyOwner
	setRouterAddress	External	✓	onlyOwner
	setSwapAndLiquifyEnabled	External	✓	onlyOwner
	<Receive Ether>	External	Payable	-
	withdrawStuckedFunds	External	✓	onlyOwner
	withdrawStuckedTokens	External	✓	onlyOwner
	_reflectFee	Private	✓	
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidityAndMarketing	Private	✓	
	_takeDevAndBurn	Private	✓	
	calculateTaxFee	Private		
	calculateDevAndBurnFee	Private		
	calculateLiquidityAndMarketingFee	Private		
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	
	isExcludedFromFee	Public		-
	_approve	Private	✓	

	_transfer	Private	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	

Contract Flow



Domain Info

Domain Name	slumdogz-token.com
Registry Domain ID	2682308114_DOMAIN_COM-VRSN
Creation Date	2022-03-17T00:00:00Z
Updated Date	2022-03-17T00:00:00Z
Registry Expiry Date	2023-03-17T00:00:00Z
Registrar WHOIS Server	whois.cronon.net
Registrar URL	http://www.cronon.net
Registrar	Cronon AG
Registrar IANA ID	141

The domain has been created in 8 months before the creation of the audit.

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

Summary

There are some functions that can be abused by the owner like transferring tokens to the team’s wallet and manipulating fees. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

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

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

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

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

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

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

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

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

The Cyberscope team

https://www.cyberscope.io