Cyberscope

Audit Report

DogeSQ

June 2022

Type	BEP20
Network	BSC
Address	0x219a756d08694cbe0b8f4d0298094104a2ed1357
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	DOGESQ
Compiler Version	v0.8.3+commit.8d00100c
Optimization	200 runs
Licence	None
Explorer	https://bscscan.com/token/0x219A756D08694Cbe0b8f4d0298094104A2ED1357
Symbol	DogeSQ
Decimals	9
Total Supply	100,000,000,000
Domain	dogesq.io

Source Files

Filename	**SHA256**
contract.sol	8505b7a659332bab15c3e23113d2ee98e194b68ec1be4dd9d6b0f1b5008c044d

Audit Updates

Initial Audit	15th June 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	medium
Location	contract.sol#L767, 797

Description

The contract owner has the authority to stop transactions for all users excluding the owner. The owner may take advantage of it by setting the fees (_marketingFee , _liquidityFee , _taxFee , _BurnFee) to a high percentage value.

```
if(!isExcludedFromFee[sender] && !isExcludedFromFee[recipient]){
    transferAmount = collectFee(sender,amount,rate);
}
```

```
function collectFee(address account, uint256 amount, uint256 rate) private
returns (uint256) {

        uint256 transferAmount = amount;

        uint256 marketingFee = amount.mul(_marketingFee).div(10000);
        uint256 liquidityFee = amount.mul(_liquidityFee).div(10000);
        uint256 taxFee = amount.mul(_taxFee).div(10000);
        uint256 BurnFee = amount.mul(_BurnFee).div(10000);
        //@dev for holders distribution
        if (taxFee > 0) {
            transferAmount = transferAmount.sub(taxFee);
            _reflectionTotal = _reflectionTotal.sub(taxFee.mul(rate));
            _taxFeeTotal = _taxFeeTotal.add(taxFee);
            emit RewardsDistributed(taxFee);
        }

        //@dev Marketing fee
        if(marketingFee > 0){
            transferAmount = transferAmount.sub(marketingFee);
            _reflectionBalance[marketingAddress] =
_reflectionBalance[marketingAddress].add(marketingFee.mul(rate));
            _marketingFeeTotal = _marketingFeeTotal.add(marketingFee);
            emit Transfer(account,marketingAddress,marketingFee);
        }
        //@dev Burn fee
        if(BurnFee > 0){
            transferAmount = transferAmount.sub(BurnFee);
            _reflectionBalance[BurnAddress] =
_reflectionBalance[BurnAddress].add(BurnFee.mul(rate));
            _BurnFeeTotal = _BurnFeeTotal.add(BurnFee);
            emit Transfer(account,BurnAddress,BurnFee);
        }

        //@dev Liquidity fee
        if(liquidityFee > 0){
            transferAmount = transferAmount.sub(liquidityFee);
            _reflectionBalance[liquidityAddress] =
_reflectionBalance[liquidityAddress].add(liquidityFee.mul(rate));
            _liquidityFeeTotal = _liquidityFeeTotal.add(liquidityFee);
            emit Transfer(account,liquidityAddress,liquidityFee);
        }
```

Recommendation

The contract could embody a check for not allowing setting the `_marketingFee` , `_liquidityFee` , `_taxFee` and `_BurnFee` more than a reasonable amount.

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#L1045, 1049, 1054, 1060

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 setReflectionFee , setLiquidityFee, setMarketingFee, and setBurnPercent functions with a high percentage value.

```
function setReflectionFee(uint256 fee) public onlyOwner {
    _taxFee = fee;
}
```

```
function setLiquidityFee(uint256 fee) public onlyOwner {
    _liquidityFee = fee;
}
```

```
function setMarketingFee(uint256 fee) public onlyOwner {
    _marketingFee = fee;
}
```

```
function setBurnPercent(uint256 fee) public onlyOwner {
    _BurnFee = fee;
}
```

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
●	BLC	Business Logic Concern
●	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
●	L11	Unnecessary Boolean equality
●	L13	Divide before Multiply Operation

BLC - Business Logic Concern

Criticality	minor
Location	contract.sol#L588, 725, 1066

Description

The business logic seems peculiar. The implementation may not follow the expected behavior. In the following code segments, the contract checks if the account address that is passed in the excludeAccount function is different from the initial value of the liquidityAddress.

```
address public liquidityAddress = 0x10ED43C718714eb63d5aA57B78B54704E256024E;
// PancakeSwapRouterV2
```

```
function excludeAccount(address account) external onlyOwner() {
    require(account != 0x10ED43C718714eb63d5aA57B78B54704E256024E,"CHAR:
Uniswap router cannot be excluded.");
```

But the user has the authority to change this address by calling the setLiquidityAddress.

```
function setLiquidityAddress(address _Address) public onlyOwner {
    require(_Address != liquidityAddress);

    liquidityAddress = _Address;
}
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

CO - Code Optimization

Criticality	minor
Location	contract.sol#L555, 607, 616, 626, 634, 643, 648

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. The contract does not use the blacklisted variable or any of the functions and modifiers that are in the code segments below.

```
address public blacklister;
mapping(address => bool) internal blacklisted;
event Blacklisted(address indexed _account);
event UnBlacklisted(address indexed _account);
event BlacklisterChanged(address indexed newBlacklister);
```

```
modifier onlyBlacklister() {
    require(msg.sender == blacklister);
    _;
}
```

```
modifier notBlacklisted(address _account) {
    require(blacklisted[_account] == false);
    _;
}
```

```
function isBlacklisted(address _account) public view returns (bool) {
    return blacklisted[_account];
}
```

```
function blacklist(address _account) public onlyBlacklister {
    blacklisted[_account] = true;
    emit Blacklisted(_account);
}
```

```
function unBlacklist(address _account) public onlyBlacklister {
     blacklisted[_account] = false;
     emit UnBlacklisted(_account);
}
```

```
function updateBlacklister(address _newBlacklister) public onlyOwner {
    require(_newBlacklister != address(0));
    blacklister = _newBlacklister;
    emit BlacklisterChanged(blacklister);
}
```

Recommendation

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

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L510,529,538,626,634,643,648,658,662,666,675,680,684,689,696,701,706,710,788,1037,1041,1045,1049,1053,1056,1060,1066

Description

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

```
setLiquidityAddress
setMarketingAddress
setBurnPercent
setMarketingFee
setLiquidityFee
setReflectionFee
IncludeFromFee
ExcludedFromFee
_burn
...
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L589,563,561,562

Description

Constant state variables should be declared constant to save gas.

```
_symbol
_name
_decimals
BurnAddress
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L626,634,643,648,788,1037,1041,1060,1066,565,566,567,570,571,574,575,577,578,579,580,582,583,584,585,589,864

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.

```
_delegates
BurnAddress
_liquidityFeeTotal
_taxFeeTotal
_BurnFeeTotal
_marketingFeeTotal
_liquidityFee
_BurnFee
_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#L1045,1049,1053,1056

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.

```
_BurnFee = fee
_marketingFee = fee
_liquidityFee = fee
_taxFee = fee
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L447,382,395,414,434,318,350

Description

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

```
sendValue
isContract
functionCallWithValue
functionCall
_functionCallWithValue
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor
Location	contract.sol#L616

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(bool)(blacklisted[_account] == false)
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L797

Description

Performing divisions before multiplications may cause lose of prediction.

```
liquidityFee = amount.mul(_liquidityFee).div(10000)
BurnFee = amount.mul(_BurnFee).div(10000)
marketingFee = amount.mul(_marketingFee).div(10000)
taxFee = amount.mul(_taxFee).div(10000)
```

Recommendation

The multiplications should be prior to the divisions.

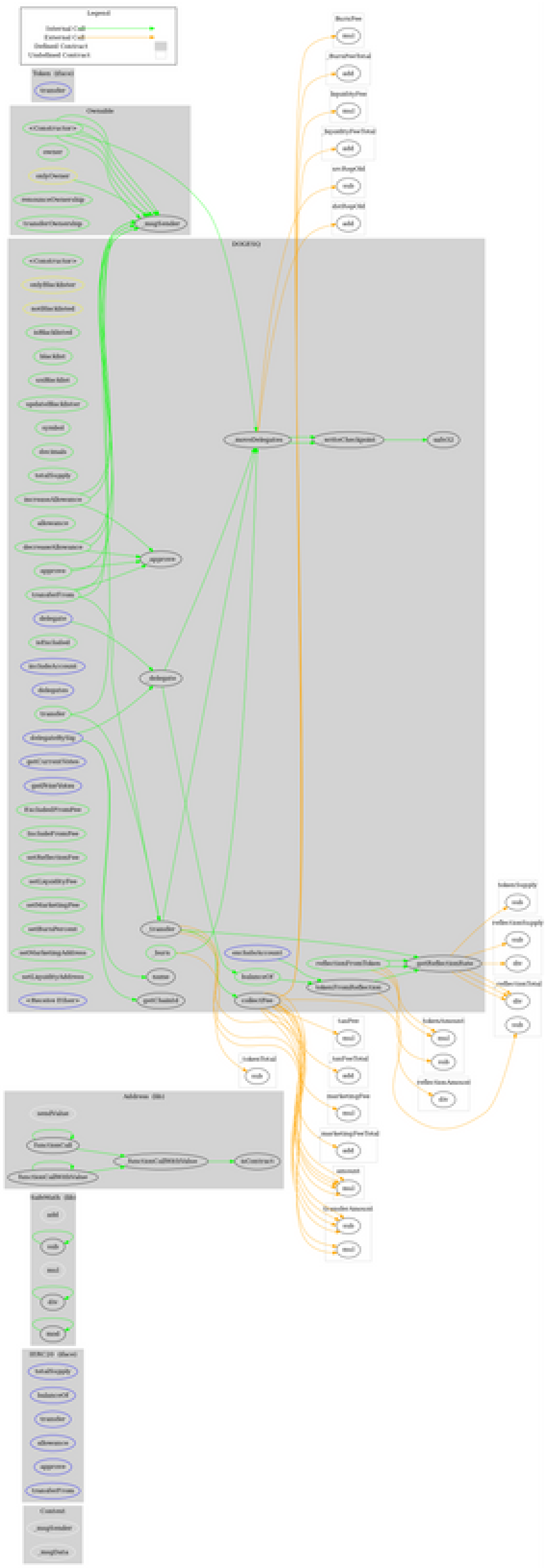
Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	

	functionCallWithValue	Internal	✓	
	_functionCallWithValue	Private	✓	
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
Token	Interface			
	transfer	External	✓	-
DOGESQ	Implementation	Context, IERC20, Ownable		
	<Constructor>	Public	✓	-
	isBlacklisted	Public		-
	blacklist	Public	✓	onlyBlacklister
	unBlacklist	Public	✓	onlyBlacklister
	updateBlacklister	Public	✓	onlyOwner
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	isExcluded	Public		-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-
	excludeAccount	External	✓	onlyOwner

	includeAccount	External	✓	onlyOwner
	_approve	Private	✓	
	_transfer	Private	✓	
	_burn	Public	✓	onlyOwner
	collectFee	Private	✓	
	_getReflectionRate	Private		
	delegates	External		-
	delegate	External	✓	-
	delegateBySig	External	✓	-
	getCurrentVotes	External		-
	getPriorVotes	External		-
	_delegate	Internal	✓	
	_moveDelegates	Internal	✓	
	_writeCheckpoint	Internal	✓	
	safe32	Internal		
	getChainId	Internal		
	ExcludedFromFee	Public	✓	onlyOwner
	IncludeFromFee	Public	✓	onlyOwner
	setReflectionFee	Public	✓	onlyOwner
	setLiquidityFee	Public	✓	onlyOwner
	setMarketingFee	Public	✓	onlyOwner
	setBurnPercent	Public	✓	onlyOwner
	setMarketingAddress	Public	✓	onlyOwner
	setLiquidityAddress	Public	✓	onlyOwner
	<Receive Ether>	External	Payable	-

Contract Flow



Domain Info

Domain Name	dogesq.io
Registry Domain ID	8fdebda390c446c49d7036c615c4cd1f-DONUTS
Creation Date	2022-04-07T21:13:18Z
Updated Date	2022-04-12T21:14:01Z
Registry Expiry Date	2023-04-07T21:13:18Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	https://www.namecheap.com/
Registrar	NameCheap, Inc.
Registrar IANA ID	1068

The domain has been created 2 months before the creation of the audit. It will expire in 10 months.

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 stopping transactions 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