Cyberscope

Audit Report

TDOGE FOOD

December 2022

Type	BEP20
Network	BSC
Address	0x21F9D17e342Ca903a6361fb28257D4f5665390f0
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	TFOOD
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0x21F9D17e342Ca903a6361fb28257D4f5665390f0
Symbol	TDOGE FOOD
Decimals	9
Total Supply	50,000,000,000
Domain	

Source Files

Filename	**SHA256**
contract.sol	fadb61bdcfe384292674db9ee370e257874b0e88c6a4fe64b465c04986ab9ca2

Audit Updates

Initial Audit	13th December 2022
Corrected	

Contract Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Unresolved
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

ST - Stops Transactions

Criticality	medium
Location	contract.sol#L285
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users excluding the owner. The owner may take advantage of it by calling the goAddLP without calling the goMoon method.

```
if (0 == goMoonBlock) {
    require(0 < goAddLPBlock && _swapPairList[to], "!goAddLP");
}
```

Recommendation

The contract could embody a check for not allowing setting the _maxTxAmount 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.

ELFM - Exceeds Fees Limit

Criticality	critical
Location	contract.sol#L323
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by setting the kb variable to a very low value, which will lead to _funTransfer being called and the user will be taxed with 90% fee.

```
function _funTransfer(address sender, address recipient, uint256 tAmount)
private {
    _balances[sender] = _balances[sender] - tAmount;
    uint256 feeAmount = tAmount * 90 / 100;
    _takeTransfer(sender, fundAddress, feeAmount);
    _takeTransfer(sender, recipient, tAmount - feeAmount);
}
```

Recommendation

The contract should lower the tax fee to at least the max allowed limit, which is 25%.

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.

BC - Blacklists Addresses

Criticality	critical
Location	contract.sol#L536
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `manage_bl` function.

```
function manage_bl(address[] calldata addresses, bool status) public onlyOwner
{
    require(addresses.length < 201);
    for (uint256 i; i < addresses.length; ++i) {
        _blackList[addresses[i]] = status;
    }
}
```

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.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	ZD	Zero Division	Unresolved
●	RLC	Redundant Limit Checks	Unresolved
●	UL	Unused Liquidity	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L06	Missing Events Access Control	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved

ZD - Zero Division

Criticality	medium
Location	contract.sol#L377
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert. The variable swapFee is the result of the following math operation: _buyFundFee +_buybuybackfee + _buyLPDividendFee + _sellFundFee + _sellbuybackfee + _sellLPDividendFee + _sellLPFee. All these values can be set to zero.

```
uint256 lpAmount = tokenAmount * lpFee / swapFee;
```

Recommendation

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

RLC - Redundant Limit Checks

Criticality	minor / informative
Location	contract.sol#L352,355,370
Status	Unresolved

Description

The contract is using redundant checks. The variables maxSellAmount, maxBuyAmount, and walletLimit are all equal to the totalSupply, so they can never be reached.

```
require(tAmount <= maxSellAmount,"over max sell amount");
…
require(tAmount <= maxBuyAmount,"over max buy amount");
…
require((balanceOf(recipient) + tAmount - feeAmount) <= walletLimit,"over max wallet limit");
```

Recommendation

The team is advised to remove these checks and therefore the variable declarations, as they are not required.

UL - Unused Liquidity

Criticality	medium
Location	contract.sol#L374
Status	Unresolved

Description

The contract will never add liquidity because of the following condition:

- _sellLPFee = 0 (this value never changes).
- So lpAmount will always be zero.

```
function swapTokenForFund(uint256 tokenAmount, uint256 swapFee) private lockTheSwap {
    swapFee += swapFee;
    uint256 lpFee = _sellLPFee;
    uint256 lpAmount = tokenAmount * lpFee / swapFee;
    …
    if (lpAmount > 0) {
        uint256 lpFist = fistBalance * lpFee / swapFee;
        if (lpFist > 0) {
            _swapRouter.addLiquidity(address(this), _fist, lpAmount, lpFist, 0, 0, fundAddress,
block.timestamp);
        }
    }
}
```

Recommendation

The team is advised to either initialize `_sellLPFee` to a higher value than 0 or add a function that can set a new value to it.

L02 - State Variables could be Declared Constant

Criticality	minor / informative
Location	contract.sol#L144,121
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
_sellLPFee
limitEnable
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contract.sol#L527,130,536,148,144,43,129,141,143,124,139,70,513,142,123,136,140,131,138
Status	Unresolved

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.

```
multiTransfer_fixed
_fist
manage_bl
_mainPair
_sellLPFee
WETH
_swapRouter
_sellLPDividendFee
_sellbuybackfee
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-conventions.

L06 - Missing Events Access Control

Criticality	minor / informative
Location	contract.sol#L423,419
Status	Unresolved

Description

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

```
buybackAddress = addr
fundAddress = addr
```

Recommendation

Emit an event for critical parameter changes.

L14 - Uninitialized Variables in Local Scope

Criticality	minor / informative
Location	contract.sol#L515,281,280,346,538
Status	Unresolved

Description

The are variables that are defined in the local scope and are not initialized.

```
i
isSell
takeFee
feeAmount
```

Recommendation

All the local scoped variables should be initialized.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	decimals	External		-
	symbol	External		-
	name	External		-
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
ISwapRouter	Interface			
	factory	External		-
	WETH	External		-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	addLiquidity	External	✓	-
ISwapFactory	Interface			
	createPair	External	✓	-
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
TokenDistributor	Implementation			

	<Constructor>	Public	✓	-
AbsToken	Implementation	IERC20, Ownable		
	<Constructor>	Public	✓	-
	symbol	External		-
	name	External		-
	decimals	External		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_approve	Private	✓	
	setkb	Public	✓	onlyOwner
	_transfer	Private	✓	
	_funTransfer	Private	✓	
	_tokenTransfer	Private	✓	
	swapTokenForFund	Private	✓	lockTheSwap
	_takeTransfer	Private	✓	
	setFundAddress	External	✓	onlyFunder
	setBuyBackAddress	External	✓	onlyFunder
	setBuyLPDividendFee	External	✓	onlyOwner
	setBuyFundFee	External	✓	onlyOwner
	setBuyBackBuyFee	External	✓	onlyOwner
	setBuyBackSellFee	External	✓	onlyOwner
	setSellLPDividendFee	External	✓	onlyOwner
	setSellFundFee	External	✓	onlyOwner
	goAddLP	External	✓	onlyOwner
	goMoon	External	✓	onlyOwner
	setFeeWhiteList	External	✓	onlyFunder
	setBlackList	External	✓	onlyOwner
	setSwapPairList	External	✓	onlyFunder
	claimBalance	External	✓	onlyFunder
	claimToken	External	✓	onlyFunder

	<Receive Ether>	External	Payable	-
	addHolder	Private	✓	
	manage_wl	Public	✓	onlyOwner
	_basicTransfer	Internal	✓	
	multiTransfer_fixed	External	✓	onlyOwner
	manage_bl	Public	✓	onlyOwner
	processReward	Private	✓	
	setHolderRewardCondition	External	✓	onlyFunder
	setExcludeHolder	External	✓	onlyFunder
TFOOD	Implementation	AbsToken		
	<Constructor>	Public	✓	AbsToken

Contract Flow

Summary

There are some functions that can be abused by the owner like stopping transactions, manipulating fees and massively blacklisting addresses. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. Lastly, a user cannot sell more than 99,99% of his balance.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io