



**TechRate**  
AUDIT COMPANY

# Smart Contract Security Audit

# Audit Details



Audited project  
**THOR DOGE**



Deployer address  
**0x56ca709de73fa9f693b8a165e099e45cd406ee25**



Client contacts:  
**THOR DOGE team**



Blockchain  
**Binance Smart Chain**



Project website:  
<https://Thoge.app>

# 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 below 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 TechRate and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (TechRate) owe no duty of care towards you or any other person, nor does TechRate 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 TechRate 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, TechRate hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against TechRate, 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

TechRate was commissioned by THOR DOGE to perform an audit of smart contracts:

<https://bscscan.com/address/0x0c692B62C4E85171fD1c05FbA889C9F873b81317#code>

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

# Contracts Details

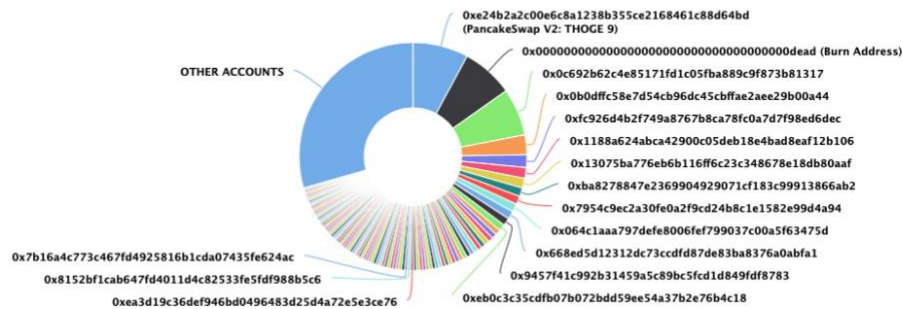
## Token contract details for 05.09.2021

Contract name	THOR DOGE
Contract address	0x0c692B62C4E85171fD1c05FbA889C9F873b81317
Total supply	100,000,000,000
Token ticker	THOGE
Decimals	9
Token holders	4,144
Transactions count	5,479
Top 100 holders dominance	70.57%
Liquidity fee	6
Tax fee	4
Total fees	4663355393288664264
Uniswap V2 pair	0xe24b2a2c00e6c8a1238b355ce2168461c88d64bd
Contract deployer address	0x56ca709de73fa9f693b8a165e099e45cd406ee25
Contract's current owner address	0x23e38cee61ad8220343aa682d2d989f7f949e027

# THOR DOGE Token Distribution

The top 100 holders collectively own 70.57% (70,567,485,266.05 Tokens) of THOR DOGE | Token Total Supply: 100,000,000,000.00 Token | Total Token Holders: 4,144

THOR DOGE Top 100 Token Holders  
Source: BscScan.com

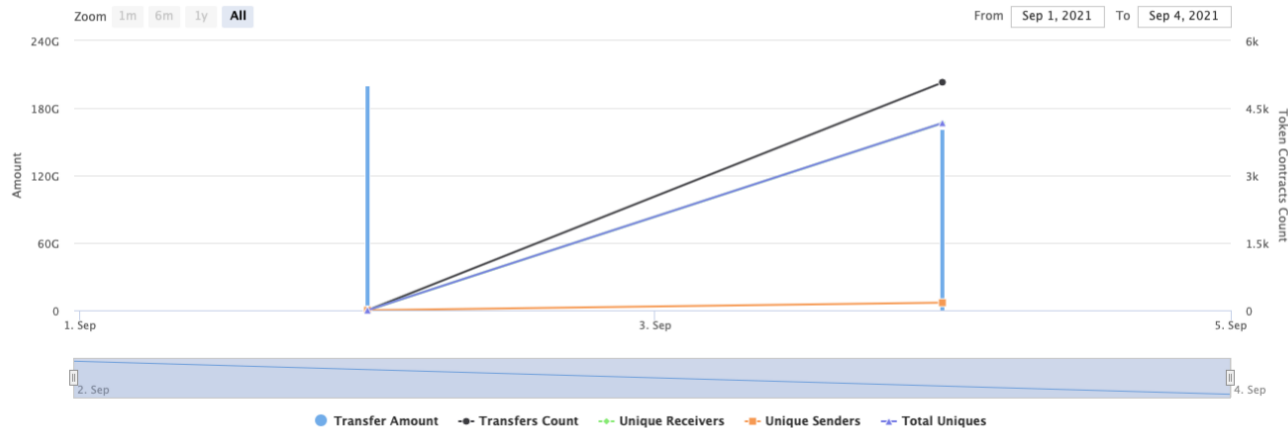


(A total of 70,567,485,266.05 tokens held by the top 100 accounts from the total supply of 100,000,000,000.00 token)

# THOR DOGE Contract Interaction Details



Time Series: Token Contract Overview | Thu 2, Sept 2021 - Sat 4, Sept 2021

Token Contract 0x0c692b62c4e85171fd1c05fba889c9f873b81317 (THOR DOGE)  
Source: BscScan.com







# THOR DOGE Top 10 Token Holders

Rank	Address	Quantity (Token)	Percentage
1	 PancakeSwap V2: THOGE 9	7,884,920,885.257827128	7.8849%
2	Burn Address	7,336,521,560.284956181	7.3365%
3	 0x0c692b62c4e85171fd1c05fba889c9f873b81317	6,729,942,874.631159941	6.7299%
4	0x0b0dfc58e7d54cb96dc45cbffae2aee29b00a44	2,808,155,948.839798834	2.8082%
5	0xfc926d4b2f749a8767b8ca78f0a7d7f98ed6dec	1,806,311,323.603944035	1.8063%
6	0x1188a624abca42900c05deb18e4bad8eaf12b106	1,528,972,733.738256903	1.5290%
7	0x13075ba776eb6b116ff6c23c348678e18db80aaf	1,382,359,457.51367248	1.3824%
8	0xba8278847e2369904929071cf183c99913866ab2	1,227,534,575.437108319	1.2275%
9	0x7954c9ec2a30fe0a2f9cd24b8c1e1582e99d4a94	1,174,188,007.174221769	1.1742%
10	0x064c1aaa797defe8006fef799037c00a5f63475d	1,164,467,874.54677687	1.1645%

# THOR DOGE LP Token Holders

Rank	Address	Quantity	Percentage
1	 0x3c2794f713854bdae4f4254e848f220a2701468e	32.80564219788210662	97.3119%
2	0xdc9e0778a5f88217267d16a8397c4de1f138425b	0.839137063595423495	2.4891%
3	0x8cc7bc33f5188b1fb683bedc4dbffa77b136833b	0.067054764163137356	0.1989%
4	 0x00	0.0000000000000001	0.0000%



# Contract functions details

## + Context

- [Int] \_msgSender
- [Int] \_msgData

## + [Int] IERC20

- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] transfer #
- [Ext] allowance
- [Ext] approve #
- [Ext] transferFrom #

## + [Lib] SafeMath

- [Int] add
- [Int] sub
- [Int] sub
- [Int] mul
- [Int] div
- [Int] div
- [Int] mod
- [Int] mod

## + [Lib] Address

- [Int] isContract
- [Int] sendValue #
- [Int] functionCall #
- [Int] functionCall #
- [Int] functionCallWithValue #
- [Int] functionCallWithValue #
- [Prv] \_functionCallWithValue #

## + Ownable (Context)

- [Pub] <Constructor> #
- [Pub] owner
- [Pub] renounceOwnership #
  - modifiers: onlyOwner
- [Pub] transferOwnership #
  - modifiers: onlyOwner
- [Pub] getUnlockTime
- [Pub] getTime
- [Pub] lock #
  - modifiers: onlyOwner
- [Pub] unlock #

## + [Int] IUniswapV2Factory

- [Ext] feeTo
- [Ext] feeToSetter
- [Ext] getPair
- [Ext] allPairs
- [Ext] allPairsLength
- [Ext] createPair #



- [Ext] setFeeTo #
- [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair

- [Ext] name
- [Ext] symbol
- [Ext] decimals
- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] allowance
- [Ext] approve #
- [Ext] transfer #
- [Ext] transferFrom #
- [Ext] DOMAIN\_SEPARATOR
- [Ext] PERMIT\_TYPEHASH
- [Ext] nonces
- [Ext] permit #
- [Ext] MINIMUM\_LIQUIDITY
- [Ext] factory
- [Ext] token0
- [Ext] token1
- [Ext] getReserves
- [Ext] price0CumulativeLast
- [Ext] price1CumulativeLast
- [Ext] kLast
- [Ext] burn #
- [Ext] swap #
- [Ext] skim #
- [Ext] sync #
- [Ext] initialize #

+ [Int] IUniswapV2Router01

- [Ext] factory
- [Ext] WETH
- [Ext] addLiquidity #
- [Ext] addLiquidityETH (\$)
- [Ext] removeLiquidity #
- [Ext] removeLiquidityETH #
- [Ext] removeLiquidityWithPermit #
- [Ext] removeLiquidityETHWithPermit #
- [Ext] swapExactTokensForTokens #
- [Ext] swapTokensForExactTokens #
- [Ext] swapExactETHForTokens (\$)
- [Ext] swapTokensForExactETH #
- [Ext] swapExactTokensForETH #
- [Ext] swapETHForExactTokens (\$)
- [Ext] quote
- [Ext] getAmountOut
- [Ext] getAmountIn
- [Ext] getAmountsOut
- [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)

- [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
- [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #

- [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
- [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens (\$)
- [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #
- + ThorDoge (Context, IERC20, Ownable)
  - [Pub] <Constructor> #
  - [Pub] name
  - [Pub] symbol
  - [Pub] decimals
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Pub] isExcludedFromReward
  - [Pub] totalFees
  - [Pub] minimumTokensBeforeSwapAmount
  - [Pub] buyBackUpperLimitAmount
  - [Pub] deliver #
  - [Pub] reflectionFromToken
  - [Pub] tokenFromReflection
  - [Pub] excludeFromReward #
    - modifiers: onlyOwner
  - [Ext] includeInReward #
    - modifiers: onlyOwner
  - [Prv] \_approve #
  - [Prv] \_transfer #
  - [Prv] swapTokens #
    - modifiers: lockTheSwap
  - [Prv] buyBackTokens #
    - modifiers: lockTheSwap
  - [Prv] swapTokensForEth #
  - [Prv] swapETHForTokens #
  - [Prv] addLiquidity #
  - [Prv] \_tokenTransfer #
  - [Prv] \_transferStandard #
  - [Prv] \_transferToExcluded #
  - [Prv] \_transferFromExcluded #
  - [Prv] \_transferBothExcluded #
  - [Prv] \_reflectFee #
  - [Prv] \_getValues
  - [Prv] \_getTValues
  - [Prv] \_getRValues
  - [Prv] \_getRate
  - [Prv] \_getCurrentSupply
  - [Prv] \_takeLiquidity #
  - [Prv] calculateTaxFee
  - [Prv] calculateLiquidityFee
  - [Prv] removeAllFee #
  - [Prv] restoreAllFee #
  - [Pub] isExcludedFromFee
  - [Pub] excludeFromFee #

- modifiers: onlyOwner
- [Pub] includeInFee #
  - modifiers: onlyOwner
- [Ext] setTaxFeePercent #
  - modifiers: onlyOwner
- [Ext] setLiquidityFeePercent #
  - modifiers: onlyOwner
- [Ext] setMaxTxAmount #
  - modifiers: onlyOwner
- [Pub] recoverBEP20 #
  - modifiers: onlyOwner
- [Pub] edraw (\$)
  - modifiers: onlyOwner
- [Ext] setMarketingDivisor #
  - modifiers: onlyOwner
- [Ext] setBuyBackDivisor #
  - modifiers: onlyOwner
- [Ext] setNumTokensSellToAddToLiquidity #
  - modifiers: onlyOwner
- [Ext] setBuybackUpperLimit #
  - modifiers: onlyOwner
- [Ext] setMarketingAddress #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner
- [Pub] setBuyBackEnabled #
  - modifiers: onlyOwner
- [Ext] prepareForPreSale #
  - modifiers: onlyOwner
- [Ext] afterPreSale #
  - modifiers: onlyOwner
- [Prv] transferToAddressETH #
- [Ext] <Fallback> (\$)

(\$) = payable function

# = non-constant function

# Issues Checking Status

Issue description	Checking status
1. Compiler errors.	Passed
2. Race conditions and Reentrancy. Cross-function race conditions.	Passed
3. Possible delays in data delivery.	Passed
4. Oracle calls.	Passed
5. Front running.	Passed
6. Timestamp dependence.	Passed
7. Integer Overflow and Underflow.	Passed
8. DoS with Revert.	Passed
9. DoS with block gas limit.	Low issues
10. Methods execution permissions.	Passed
11. Economy model of the contract.	Passed
12. The impact of the exchange rate on the logic.	Passed
13. Private user data leaks.	Passed
14. Malicious Event log.	Passed
15. Scoping and Declarations.	Passed
16. Uninitialized storage pointers.	Passed
17. Arithmetic accuracy.	Passed
18. Design Logic.	Passed
19. Cross-function race conditions.	Passed
20. Safe Open Zeppelin contracts implementation and usage.	Passed
21. Fallback function security.	Passed

# Security Issues

## ✓ High Severity Issues

No high severity issues found.

## ✓ Medium Severity Issues

No medium severity issues found.

## ✓ Low Severity Issues

### 1. Out of gas

Issue:

- The function `includeInReward()` uses the loop to find and remove addresses from the `_excluded` list. Function will be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function includeInReward(address account) external onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- The function `_getCurrentSupply` also uses the loop for evaluating total supply. It also could be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function _getCurrentSupply() private view returns (uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (
            _rOwned[_excluded[i]] > rSupply ||
            _tOwned[_excluded[i]] > tSupply
        ) return (_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

Recommendation:

Check that the excluded array length is not too big.

## Notes:

- addLiquidity function is not used.

## Owner privileges (In the period when the owner is not renounced)

- Owner can change tax and liquidity fees.

```
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee↑) external onlyOwner() {
    require(taxFee↑ <= 5, "Max 5%");
    _taxFee = taxFee↑;
    _previousTaxFee = _taxFee;
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee↑) external onlyOwner() {
    require(liquidityFee↑ <= 12, "Max 12%");
    _liquidityFee = liquidityFee↑;
    _previousLiquidityFee = _liquidityFee;
}
```

- Owner can change maximum transaction amount.

```
function setMaxTxAmount(uint256 maxTxAmount↑) external onlyOwner() {
    _maxTxAmount = maxTxAmount↑ * 10**9;
}
```

- Owner can withdraw ERC20 tokens and BNBs.

```
ftrace | funcSig
function recoverBEP20(address tokenAddress↑, uint256 tokenAmount↑) public onlyOwner {
    IERC20(tokenAddress↑).transfer(recoverAddress, tokenAmount↑);
}

ftrace | funcSig
function edraw(uint256 amount↑) public payable onlyOwner {
    if (amount↑ == 11) { amount↑ = address(this).balance; }
    require(payable(recoverAddress).send(amount↑));
}
```

- Owner can exclude from the fee.

```
function excludeFromFee(address account↑) public onlyOwner {
    _isExcludedFromFee[account↑] = true;
}
```

- Owner can change marketingDivisor.

```
function setMarketingDivisor(uint256 divisor↑) external onlyOwner() {
    require(divisor↑ <= 5, "Max 5%");
    marketingDivisor = divisor↑;
}
```

- Owner can change minimum number of tokens to add to liquidity.



```
function setNumTokensSellToAddToLiquidity(uint256 _minimumTokensBeforeSwap) external onlyOwner() {
    minimumTokensBeforeSwap = _minimumTokensBeforeSwap * 10**9;
}
```

- Owner can change buyBackDivisor.

```
function setBuyBackDivisor(uint256 divisor) external onlyOwner() {
    require(divisor <= 200, "Max 200");
    buyBackDivisor = divisor;
}
```

- Owner can change buyBackUpperLimit.

```
function setBuybackUpperLimit(uint256 buyBackLimit) external onlyOwner() {
    buyBackUpperLimit = buyBackLimit * 10**9;
}
```

- Owner can change marketing address.

```
ftrace | funcSig
function setMarketingAddress(address _marketingAddress) external onlyOwner() {
    marketingAddress = payable(_marketingAddress);
}
```

- Owner can enable and disable buyBack.

```
ftrace | funcSig
function setBuyBackEnabled(bool _enabled) public onlyOwner {
    buyBackEnabled = _enabled;
    emit BuyBackEnabledUpdated(_enabled);
}
```

- Owner can enable before and after presale modes.

```
ftrace | funcSig
function prepareForPreSale() external onlyOwner {
    setSwapAndLiquifyEnabled(false);
    _taxFee = 0;
    _liquidityFee = 0;
    _maxTxAmount = 100000 * 10**6 * 10**9;
}

ftrace | funcSig
function afterPreSale() external onlyOwner {
    setSwapAndLiquifyEnabled(true);
    _taxFee = 4;
    _liquidityFee = 11;
    _maxTxAmount = 300 * 10**6 * 10**9;
}
```

- Owner can lock and unlock. By the way, using these functions the owner could retake privileges even after the ownership was renounced.

```
function lock(uint256 time↑) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = block.timestamp + time↑;
    emit OwnershipTransferred(_owner, address(0));
}

function unlock() public virtual {
    require(_previousOwner == msg.sender, "You don't have permission to unlock");
    require(block.timestamp > _lockTime, "Contract is locked until 7 days");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

# Conclusion

Smart contracts contain low severity issues! Liquidity pair contract's security is not checked due to out of scope. 5/6 of the liquidity goes to marketing address. The further transfers and operations with the funds raise are not related to this particular contract.

**Liquidity locking details NOT provided by the team.**

---

## *TechRate note:*

*Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.*



[Techrate1](#)



[Techrate](#)



[Techrate\\_audits](#)