Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

November, 2021

Audit Details

Audited project

Christmas Doge

Deployer address

0xd846784e2647B432F16c445606E9b3B31a43684D

Client contacts:

Christmas Doge team

Blockchain

Binance Smart Chain

Project website:

Not provided

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 Christmas Doge to perform an audit of smart contracts:

https://bscscan.com/address/0xBD2E15a846F945BdaA9Af20eDf2434cF182D16dD#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

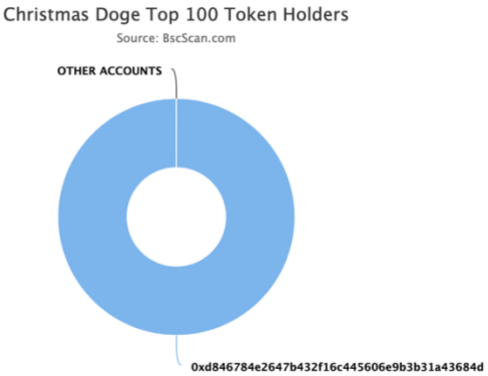
Token contract details for 25.11.2021

Contract name	Christmas Doge
Contract address	0xBD2E15a846F945BdaA9Af20eDf2434cF182D16dD
Total supply	100,000,000,000
Token ticker	ChristmasDoge
Decimals	9
Token holders	1
Transactions count	1
Top 100 holders dominance	100.00%
Total fee	1400
Autoliquidity fee receiver	0xd846784e2647B432F16c445606E9b3B31a43684D
Marketing fee receiver	0xceCef5C685847C594B40C0A2ec7Ae702C2a1A598
Pair	0x424d20A18b3938Fa23a87756d7C773076c361710
Contract deployer address	0xd846784e2647B432F16c445606E9b3B31a43684D
Contract's current owner address	0xd846784e2647B432F16c445606E9b3B31a43684D

Christmas Doge Token Distribution

The top 100 holders collectively own 100.00% (100,000,000,000.00 Tokens) of Christmas Doge

Token Total Supply: 100,000,000,000.00 Token | Total Token Holders: 1



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

Christmas Doge Contract Interaction Details

Time Series: Token Contract Overview

Tue 23, Nov 2021 - Tue 23, Nov 2021

Christmas Doge Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	0xd846784e2647b432f16c445606e9b3b31a43684d	100,000,000,000	100.0000%

Contract functions details

+ [Lib] SafeMath
  - [Int] tryAdd
  - [Int] trySub
  - [Int] tryMul
  - [Int] tryDiv
  - [Int] tryMod
  - [Int] add
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] mod
  - [Int] sub
  - [Int] div
  - [Int] mod

+ [Int] IBEP20
  - [Ext] totalSupply
  - [Ext] decimals
  - [Ext] symbol
  - [Ext] name
  - [Ext] getOwner
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #

+ Auth
  - [Pub] <Constructor> #
  - [Pub] authorize #
    - modifiers: onlyOwner
  - [Pub] isOwner
  - [Pub] isAuthorized
  - [Pub] transferOwnership #
    - modifiers: onlyOwner

+ [Int] IDEXFactory
  - [Ext] createPair #

+ [Int] IDEXRouter
  - [Ext] factory
  - [Ext] WETH
  - [Ext] addLiquidity #
  - [Ext] addLiquidityETH ($)
  - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
  - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
  - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ [Int] IDividendDistributor
  - [Ext] setDistributionCriteria #
  - [Ext] setShare #

- [Ext] deposit ($)
- [Ext] process #

+ DividendDistributor (IDividendDistributor)
- [Pub] <Constructor> #
- [Ext] setDistributionCriteria #
  - modifiers: onlyToken
- [Ext] setShare #
  - modifiers: onlyToken
- [Ext] deposit ($)
  - modifiers: onlyToken
- [Ext] process #
  - modifiers: onlyToken
- [Int] shouldDistribute
- [Int] distributeDividend #
- [Ext] claimDividend #
- [Pub] getUnpaidEarnings
- [Int] getCumulativeDividends
- [Int] addShareholder #
- [Int] removeShareholder #

+ ChristmasDoge (IBEP20, Auth)
- [Pub] <Constructor> #
  - modifiers: Auth
- [Ext] <Fallback> ($)
- [Ext] totalSupply
- [Ext] decimals
- [Ext] symbol
- [Ext] name
- [Ext] getOwner
- [Pub] balanceOf
- [Ext] allowance
- [Pub] approve #
- [Ext] approveMax #
- [Ext] transfer #
- [Ext] transferFrom #
- [Ext] setMaxWalletPercent #
  - modifiers: onlyOwner
- [Int] _transferFrom #
- [Int] _basicTransfer #
- [Int] checkTxLimit
- [Int] shouldTakeFee
- [Int] takeFee #
- [Int] shouldSwapBack
- [Ext] clearStuckBalance #
  - modifiers: onlyOwner
- [Pub] cooldownEnabled #
  - modifiers: onlyOwner
- [Int] swapBack #
  - modifiers: swapping
- [Ext] setTxLimit #
  - modifiers: authorized
- [Ext] setIsDividendExempt #
  - modifiers: authorized
- [Ext] setIsFeeExempt #

  - modifiers: authorized
- [Ext] setIsTxLimitExempt #
  - modifiers: authorized
- [Ext] setIsTimelockExempt #
  - modifiers: authorized
- [Ext] setFees #
  - modifiers: authorized
- [Ext] setFeeReceivers #
  - modifiers: authorized
- [Ext] setSwapBackSettings #
  - modifiers: authorized
- [Ext] setTargetLiquidity #
  - modifiers: authorized
- [Ext] setDistributionCriteria #
  - modifiers: authorized
- [Ext] setDistributorSettings #
  - modifiers: authorized
- [Pub] getCirculatingSupply
- [Pub] getLiquidityBacking
- [Pub] isOverLiquified

($) = 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.	Passed
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

No low severity issues found.

Notes

- The function swapBack send reflection BNBs to marketing wallet address and send marketing BNBs to distributor.

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

- Owner can authorize addresses.

```
function authorize(address adr) public onlyOwner {
    authorizations[adr] = true;
}
```

- Owner can change maximum token amount per wallet.

```
 function setMaxWalletPercent(uint256 maxWallPercent) external onlyOwner() {
    _maxWalletToken = (_totalSupply * maxWallPercent ) / 100;
}
```

- Owner can withdraw BNBs to the marketing receiver address.

```
function clearStuckBalance(uint256 amountPercentage) external onlyOwner {
    uint256 amountBNB = address(this).balance;
    payable(marketingFeeReceiver).transfer(amountBNB * amountPercentage / 100);
}
```

- Owner can change cooldown status.

```
function cooldownEnabled(bool _status, uint8 _interval) public onlyOwner {
    buyCooldownEnabled = _status;
    cooldownTimerInterval = _interval;
}
```

- Authorized addresses can change the maximum transaction amount limit.

```
function setTxLimit(uint256 amount) external authorized {
    _maxTxAmount = amount;
}
```

- Authorized addresses can include in and exclude from dividends.

```
function setIsDividendExempt(address holder, bool exempt) external authorized {
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if(exempt){
        distributor.setShare(holder, 0);
    }else{
        distributor.setShare(holder, _balances[holder]);
    }
}
```

- Authorized addresses can include in and exclude from fees.

```
function setIsFeeExempt(address holder, bool exempt) external authorized {
    isFeeExempt[holder] = exempt;
}
```

- Authorized addresses can include in and exclude from transaction amount limit.

```
function setIsTxLimitExempt(address holder, bool exempt) external authorized {
    isTxLimitExempt[holder] = exempt;
}
```

- Authorized addresses can include in and exclude from cooldown interval.

```
function setIsTimelockExempt(address holder, bool exempt) external authorized {
    isTimelockExempt[holder] = exempt;
}
```

- Authorized addresses can change liquidity, reflection and marketing fees.

```
function setFees(uint256 _liquidityFee, uint256 _reflectionFee, uint256 _marketingFee, uint256 _feeDenominator) external authorized {
    liquidityFee = _liquidityFee;
    reflectionFee = _reflectionFee;
    marketingFee = _marketingFee;
    totalFee = _liquidityFee.add(_reflectionFee).add(_marketingFee);
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator/4);
}
```

- Authorized addresses can change liquidity and marketing fee receivers.

```
function setFeeReceivers(address _autoLiquidityReceiver, address _marketingFeeReceiver) external authorized {
    autoLiquidityReceiver = _autoLiquidityReceiver;
    marketingFeeReceiver = _marketingFeeReceiver;
}
```

- Authorized addresses can change swap threshold and enable / disable swap.

```
function setSwapBackSettings(bool _enabled, uint256 _amount) external authorized {
    swapEnabled = _enabled;



    swapThreshold = _amount;
}
```

- Authorized addresses can change target liquidity values.

```
function setTargetLiquidity(uint256 _target, uint256 _denominator) external authorized {
    targetLiquidity = _target;
    targetLiquidityDenominator = _denominator;
}
```

- Authorized addresses can change distribution criteria.

```
function setDistributionCriteria(uint256 _minPeriod, uint256 _minDistribution) external authorized {
    distributor.setDistributionCriteria(_minPeriod, _minDistribution);
}
```

- Authorized addresses can change distribution GAS.

```
function setDistributorSettings(uint256 gas) external authorized {
    require(gas < 750000);
    distributorGas = gas;
}
```

Conclusion

Smart contracts do not contain high severity issues! Smart contracts contain owner privileges. Liquidity pair contract's security is not checked due to out of scope.

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.