freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

DogeDead

$DOGEDEAD

04/02/2022

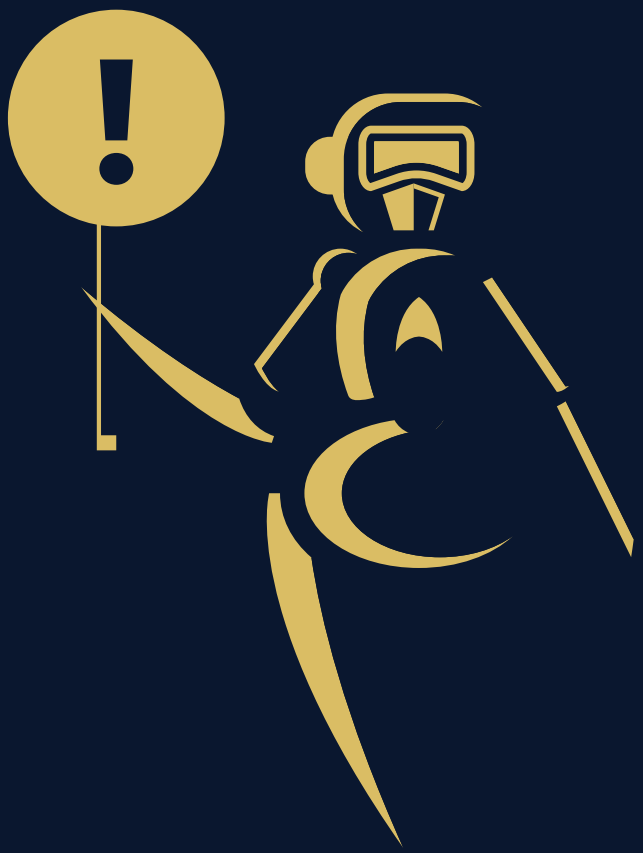
TABLE OF CONTENTS

DISCLAIMER

The information provided on this analysis document is only for general information and should not be used as a reason to invest.

FreshCoins Team will take no payment for manipulating the results of this audit.

The score and the result will stay on this project page information on our website https://freshcoins.io
FreshCoins Team does not guarantees that a project will not sell off team supply, or any other scam strategy (RUG or Honeypot etc)



INTRODUCTION

FreshCoins (Consultant) was contracted by DogeDead (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0xd890fea1da0348d3981af8ae200fa575068f3846

Network: Binance Smart Chain (BSC)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 04/02/2022

WEBSITE DIAGNOSTIC

https://dogedead.com/

0-49 50-89 90-100

Performance	Accessibility	Best Practices	SEO	Progressive Web App
72	90	92	90	NA

Metrics

First Contentful Paint
3.4 s

Time to interactive
3.2 s

Speed Index
8.7 s

Total Blocking Time
170 ms

Large Contentful Paint
5.1 s

Cumulative Layout Shift
0

WEBSITE IMPROVEMENTS

Use video formats for animated content

Reduce unused CSS

Reduce unused JavaScript

Reduce JavaScript execution time 3.4 s

Avoid enormous network payloads Total size was 5,382 KiB

Links do not have a discernible name

Buttons do not have an accessible name

AUDIT OVERVIEW

93

Security Score

91

Static Scan
Automatic scanning for common vulnerabilities

96

ERC Scan
Automatic checks for ERC's conformance

0 High

0 Medium

1 Low

0 Optimizations

0 Informational

No.	Issue description	Checking Status
1	Compiler Errors / Warnings	Passed
2	Reentrancy and Cross-function	Passed
3	Front running	Passed
4	Timestamp dependence	Passed
5	Integer Overflow and Underflow	Passed
6	Reverted DoS	Passed
7	DoS with block gas limit	Low
8	Methods execution permissions	Passed
9	Exchange rate impact	Passed
10	Malicious Event	Passed
11	Scoping and Declarations	Passed
12	Uninitialized storage pointers	Passed
13	Design Logic	Passed
14	Safe Zeppelin module	Passed

OWNER PRIVILEGES

Contract owner can't mint tokens after initial contract deploy.

SafeToken and LockToken lib included

Contract owner can exclude/include wallet(s) from fees

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
      _isExcludedFromFees[account] = excluded;
      emit ExcludeFromFees(account, excluded);
}
```

```
function excludeMultipleAccountsFromFees(address[] calldata accounts, bool excluded) public onlyOwner {
      for(uint256 i = 0; i < accounts.length; i++)
      {
         _isExcludedFromFees[accounts[i]] = excluded;
      }
      emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

Contract owner can exclude/include wallet from rewards

```
function excludeFromRewards(address account, bool value) external onlyOwner {
            excludedFromRewards[account] = value;
            _setBalance(account, 0);
            tokenHoldersMap.remove(account);
            emit ExcludeFromRewards(account);
}
```

Contract owner can exclude/include wallet from max tx limitations

```
function excludeFromMaxTx(address _address, bool value) public onlyOwner {
      _isExcludedFromMaxTx[_address] = value;
}
```

Contract owner can exclude wallet from rewards, fees and max tx limitations

```
function excludeFromAll(address _address) public onlyOwner {
      _isExcludedFromMaxTx[_address] = true;
      _isExcludedFromFees[_address] = true;
      rewardTracker.excludeFromRewards(_address,true);
}
```

Contract owner can include wallet in whitelist

```
function includeToWhiteList(address[] memory _users) external onlyOwner {
    for(uint8 i = 0; i < _users.length; i++) {
        _whiteList[_users[i]] = true;
    }
}
```

Contract owner can change the fees up to 10%

```
function setFee(uint256 _bnbRewardFee, uint256 _buybackFee, uint256 _marketingFee) public onlyOwner {
    BNBRewardsFee = _bnbRewardFee;
    buybackFee = _buybackFee;
    marketingFee = _marketingFee;
    totalFees = BNBRewardsFee.add(buybackFee).add(marketingFee); // total fee transfer and buy
    require(totalFees <= 10, "Total fee can't be over 10% initially established");
}
```

Contract owner can change sell fees

```
function setExtraFeeOnSell(uint256 _extraFeeOnSell) public onlyOwner {
    require(_extraFeeOnSell <= 10, "Extra fee can't be over 10%");
    extraFeeOnSell = _extraFeeOnSell; // extra fee on sell
}
```

Contract owner can change marketingWallet address

Current address:

marketingWallet: 0x64c851f870c3cc03801fbb23d3c1f397a5ff7065

```
function setMarketingWallet(address newAccount) public onlyOwner {
    marketingWallet = payable(newAccount);
}
```

Contract owner can change max sell amount

```
function setMaxSellTransactionAmount(uint256 newAmount) public onlyOwner {
    require(newAmount >= 1_000_000_000 * (10**18), "MaxSellTransactionAmount has to be over 0.1% of the
total supply");
    maxSellTransactionAmount = newAmount;
}
```

Contract owner can change swap settings and amount

```
function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}
```

```
function setSwapTokensAtAmount(uint256 newAmount) public onlyOwner {
    require(newAmount <= 1_000_000 * (10**18), "swapTokensAtAmount can't to be more than 1.000.000
Tokens");
    require(newAmount >= 1_000 * (10**18), "swapTokensAtAmount can't to be less than 1.000 Tokens");
    swapTokensAtAmount = newAmount;
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    _setOwner(address(0));
}
```

Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(newOwner != address(0), "Ownable: new owner is the zero address");
    _setOwner(newOwner);
}
```

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. The risk can be prevented by temporarily locking the contract or renouncing ownership.

CONCLUSION AND ANALYSIS

Smart Contracts within the scope were manually reviewed and analyzed with static tools.

Audit report overview contains all found security vulnerabilities and other issues in the reviewed code.

Found 1 LOW issue during the first review.

TOKEN DETAILS

Details

Buy fees:	10%
Sell fees:	15%
Max TX:	N/A
Max Sell:	10,000,000,000

Honeypot Risk

Ownership:	Owned
Blacklist:	Not detected
Modify Max TX:	Detected with threshold
Modify Max Sell:	Detected with threshold
Disable Trading:	Not detected

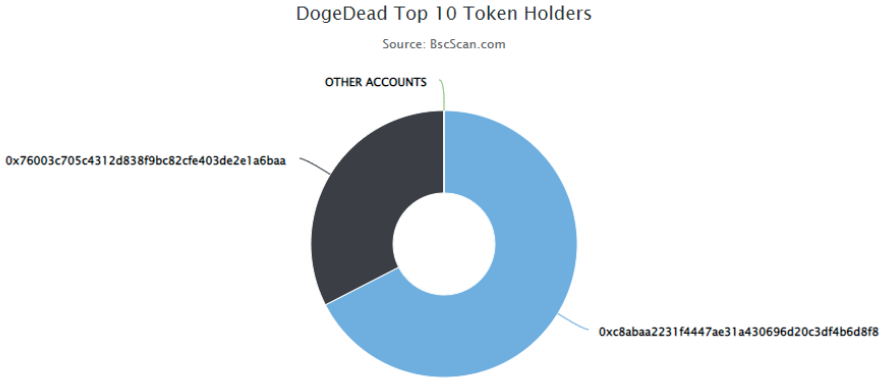
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

DOGEDEAD TOKEN DISTRIBUTION & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (1,000,000,000,000.00 Tokens) of DogeDead

Token Total Supply: 1,000,000,000,000.00 Token | Total Token Holders: 2



(A total of 1,000,000,000,000.00 tokens held by the top 10 accounts from the total supply of 1,000,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0xc8abaa2231f4447ae31a430696d20c3df4b6d8f8	674,600,000,172.48	67.4600%
2	0x76003c705c4312d838f9bc82cfe403de2e1a6baa	325,399,999,827.52	32.5400%

TECHNICAL DISCLAIMER

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. The audit can't guarantee the explicit security of the audited project / smart contract.