freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Lucky Dragon

$LUCKY

11/08/2022

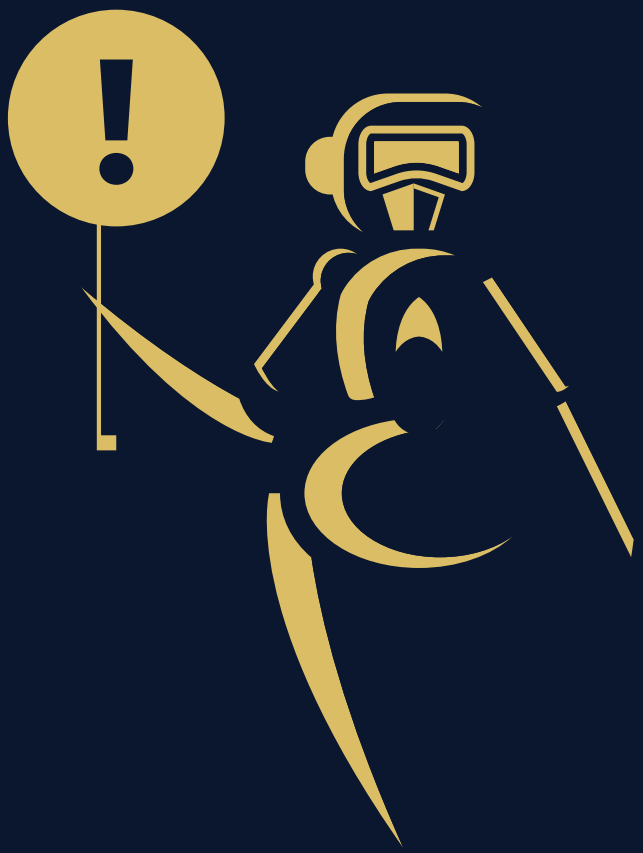
TABLE OF CONTENTS

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
Lucky Dragon (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x52d95492F421425A5A507C0516034041b95fDAa3

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 11/08/2022

AUDIT OVERVIEW

86

Security Score

82

Static Scan
Automatic scanning for common vulnerabilities

90

ERC Scan
Automatic checks for ERC's conformance

1 High

4 Medium

4 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	Passed
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

- Contract owner can exclude an address from transactions

```
function updateIsBlacklisted(address account, bool flag) public onlyOwner {
        require(
          isBlacklisted[account] != flag,
          "You must provide a different exempt address or status other than the current value in order to update
it"
        );
        isBlacklisted[account] = flag;
    }

function bulkUpdateIsBlacklisted(address[] memory accounts, bool flag)
        external
        onlyOwner
    {
        for (uint256 i = 0; i < accounts.length; i++) {
          updateIsBlacklisted(accounts[i], flag);
        }
}
```

- Contract owner can exclude/include wallet(s) from tax

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

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

- Contract owner can exclude/include wallet from tx limitations

```
function excludeFromMaxTransaction(address updAds, bool isEx) public onlyOwner {
        _isExcludedMaxTransactionAmount[updAds] = isEx;
        emit ExcludedMaxTransactionAmount(updAds, isEx);
}
```

- Contract owner can exclude/include wallet from max wallet limitations

```
function excludeFromMaxWallet(address updAds, bool isEx) public onlyOwner {
        _isExcludedMaxWalletAmount[updAds] = isEx;
        emit ExcludedMaxWalletAmount(updAds, isEx);
}
```

enableTrading() function has to be called in order to open trade

```
function enableTrading() external onlyOwner {
        tradingActive = true;
        swapEnabled = true;
        tradingActiveBlock = block.number;
}
```

Contract owner can change swap settings

```
function updateSwapTokensAtAmount(uint256 newNum) external onlyOwner {
        swapTokensAtAmount = newNum * (10**18);
}
```

Contract owner can change max tx amount

```
function updateMaxTransactionAmount(uint256 newNum) external onlyOwner {
        require(newNum > (totalSupply() * 5 / 1000)/1e18, "Cannot set maxTransactionAmount lower than
0.5%");
        maxTransactionAmount = newNum * (10**18);
}
```

Contract owner can change max wallet amount (no threshold)

```
function updateMaxWalletAmount(uint256 newNum) external onlyOwner {
        maxWalletAmount = newNum * (10**18);
}
```

Contract owner can change marketingWallet and devWallet addresses

Current values:

marketingWallet : 0x79fa4a1975870a3268da9af290dead4c863f649e

devWallet : 0x9c93fc51694f13a24aefba249f5b468d1e39549d

```
function updateMarketingWallet(address newMarketingWallet) external onlyOwner {
        excludeFromFees(newMarketingWallet, true);
        emit marketingWalletUpdated(newMarketingWallet, marketingWallet);
        marketingWallet = newMarketingWallet;
}

function updateDevWallet(address newDevWallet) external onlyOwner {
        excludeFromFees(newDevWallet, true);
        emit marketingWalletUpdated(newDevWallet, devWallet);
        devWallet = newDevWallet;
}
```

Contract owner can change buy fees up to 30% and sell fees up to 30%

```
function updateBuyFees(uint256 _marketingFee,uint256 _devFee , uint256 _rewardsFee) external onlyOwner
{
      marketingBuyFee = _marketingFee;
      devBuyFee = _devFee;
      rewardsBuyFee = _rewardsFee;
      totalBuyFees = marketingBuyFee + rewardsBuyFee + devBuyFee;
      require(totalBuyFees <= 30, "Must keep fees at 0% or less");
   }

function updateSellFees(uint256 _marketingFee, uint256 _devFee, uint256 _rewardsFee) external onlyOwner
{
      marketingSellFee = _marketingFee;
      devSellFee = _devFee;
      rewardsSellFee = _rewardsFee;
      totalSellFees = marketingSellFee + rewardsSellFee + devSellFee;
      require(totalSellFees <= 30, "Must keep fees at 30% or less");
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
      emit OwnershipTransferred(_owner, address(0));
      _owner = 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");
      emit OwnershipTransferred(_owner, newOwner);
      _owner = newOwner;
}
```

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 HIGH issue during the first review.

TOKEN DETAILS

Details

Buy fees:	7%
Sell fees:	7%
Max TX:	7,777,777
Max Sell:	N/A

Honeypot Risk

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

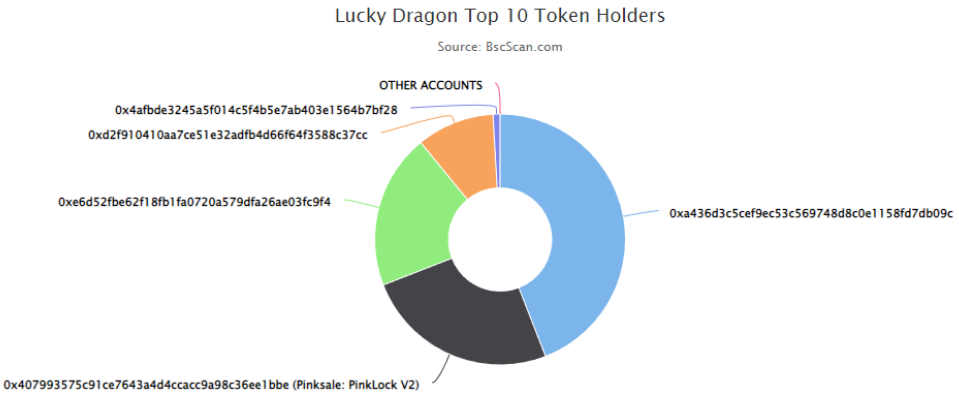
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

LUCKY DRAGON TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (7,777,777.00 Tokens) of Lucky Dragon

Token Total Supply: 7,777,777.00 Token | Total Token Holders: 5



(A total of 7,777,777.00 tokens held by the top 10 accounts from the total supply of 7,777,777.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0xa436d3c5cef9ec53c569748d8c0e1158fd7db09c	3,430,000	44.1000%
2	Pinksale: PinkLock V2	1,944,000	24.9943%
3	0xe6d52fbe62f18fb1fa0720a579dfa26ae03fc9f4	1,555,554	20.0000%
4	0xd2f910410aa7ce51e32adfb4d66f64f3588c37cc	778,227	10.0058%
5	0x4afbde3245a5f014c5f4b5e7ab403e1564b7bf28	69,996	0.8999%

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.