freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Dragon

$Dragon

11/01/2024

TOKEN OVERVIEW

Fees

• Buy fees:	5%
• Sell fees:	5%

Fees privileges

• Can change buy fees up to 10% and sell fees up to 10%

Ownership

• Owned

Minting

• No mint function

Max Tx Amount / Max Wallet Amount

• Can't change max tx amount and / or max wallet amount

Blacklist

• Blacklist function not detected

Other privileges

• Can exclude / include from fees

• Contract owner has to call EnableTrading function to enable trade

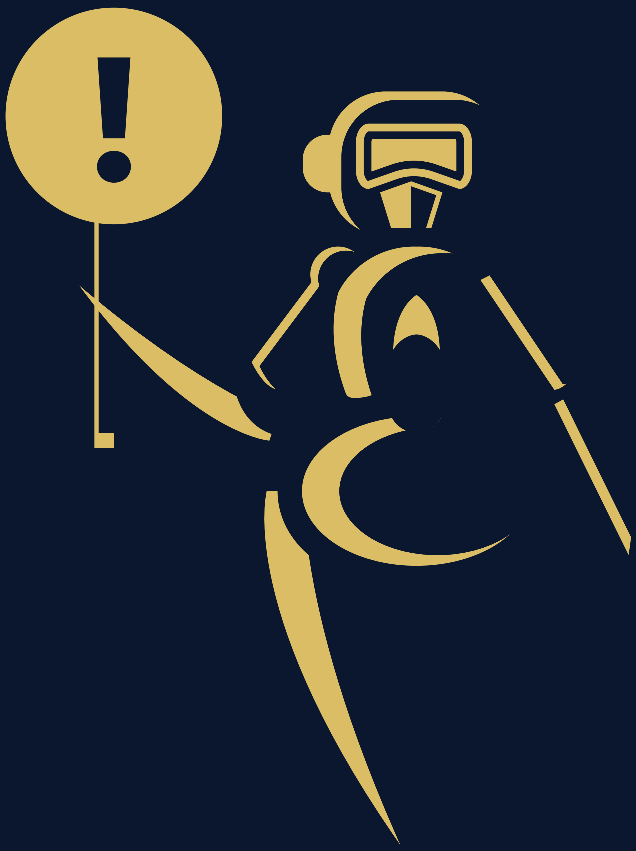
TABLE OF CONTENTS

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

0x0d14F1978119010495A02285fA767bcA92F63De3

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 11/01/2024

WEBSITE DIAGNOSTIC

N/A

0-49	50-89	90-100

Performance	Accessibility	Best Practices	SEO	Progressive Web App
NA	NA	NA	NA	NA

Socials

Twitter	Telegram
https://twitter.com/DragonBNB342687	https://t.me/DragonBNBtg

AUDIT OVERVIEW

84

Security Score

98

Static Scan

Automatic scanning for common vulnerabilities

82

ERC Scan

Automatic checks for ERC's conformance

1	High
0	Medium
0	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

● Contract owner can't exclude an address from transactions

● Contract owner can exclude/include wallet from tax

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

function includeInFee(address account) public onlyOwner {
      _isExcludedFromFee[account] = false;
}
```

● Contract owner can exclude/include wallet from rewards

```
function excludeFromReward(address account) public onlyOwner {
      require(!_isExcluded[account], "Account is already excluded");
      if (_rOwned[account] > 0) {
         _tOwned[account] = tokenFromReflection(_rOwned[account]);
      }
      _isExcluded[account] = true;
      _excluded.push(account);
}

function includeInReward(address account) external onlyOwner {
      require(_isExcluded[account], "Account is not 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;
         }
      }
}
```

● Contract owner has to call EnableTrading function to enable trade

Please note that any wallet excluded from fees retains the ability to engage in trading, even in situations where trading has been disabled

```
function EnableTrading() external onlyOwner {
      require(!tradingEnabled, "Cannot re-enable trading");
      tradingEnabled = true;
      swapEnabled = true;
      genesis_block = block.number;
}
```

● Contract owner has ability to retrieve any token held by the contract

Native tokens excluded

```
function rescueETH() external {
    uint256 contractETHBalance = address(this).balance;
    payable(marketingWallet).transfer(contractETHBalance);
}

function rescueERC20Tokens(address _tokenAddr,address _to, uint256 _amount) public onlyOwner {
    require(_tokenAddr != address(this), "Owner can't claim contract's balance of its own tokens");
    IERC20(_tokenAddr).transfer(_to, _amount);
}
```

● Contract owner can change marketingWallet address

Current value:

marketingWallet: 0x000000000000000000000000000000000000dEaD

```
function updateMarketingWallet(address newWallet) external onlyOwner {
    require(newWallet != address(0),"Fee Address cannot be zero address");
    require(!newWallet.isContract(), "Fee Address cannot be a contract");
    marketingWallet = newWallet;
}
```

● Contract owner can change buy fees up to 10% and sell fees up to 10%

```
function updateBuyTaxes(
    uint256 _rfi,
    uint256 _marketing,
    uint256 _liquidity
  ) public onlyOwner {
    require((_rfi + _marketing + _liquidity) <= 10, "Must keep fees at 10% or less");
    taxes = Taxes(_rfi, _marketing, _liquidity);
    emit FeesChanged();
}

function updateSellTaxes(
    uint256 _rfi,
    uint256 _marketing,
    uint256 _liquidity
  ) public onlyOwner {
    require((_rfi + _marketing + _liquidity) <= 10, "Must keep fees at 10% or less");
    sellTaxes = Taxes(_rfi, _marketing, _liquidity);
    emit FeesChanged();
}
```

● Contract owner can change swap settings

```
function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(amount <= 10000000000, "Cannot set swap threshold amount higher than 1% of tokens");
    require(amount >= 1000000000, "Cannot set swap threshold amount lower than 0.01% of tokens");
    swapTokensAtAmount = amount * 10**_decimals;
}

function updateSwapEnabled(bool _enabled) external onlyOwner {
    swapEnabled = _enabled;
}
```

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);
}

function _setOwner(address newOwner) private {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, 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 HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	5%
Sell fees:	5%
Max TX:	N/A
Max Sell:	N/A

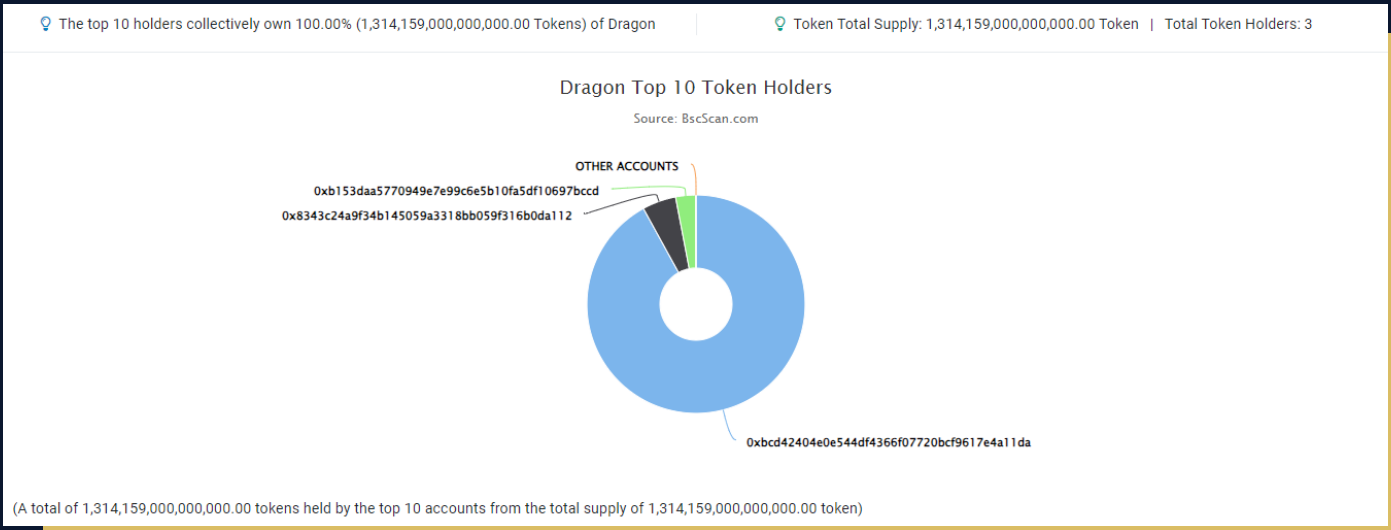
Honeypot Risk

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

Rug Pull Risk

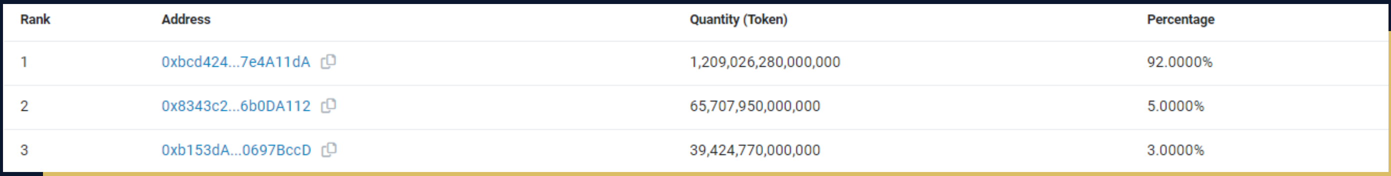
Liquidity:	N/A
Holders:	100% unlocked tokens

DRAGON TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS


The top 10 holders collectively own 100.00% (1,314,159,000,000,000.00 Tokens) of Dragon

Token Total Supply: 1,314,159,000,000,000.00 Token | Total Token Holders: 3

(A total of 1,314,159,000,000,000.00 tokens held by the top 10 accounts from the total supply of 1,314,159,000,000,000.00 token)



Rank	Address	Quantity (Token)	Percentage
1	0xbcd424...7e4A11dA	1,209,026,280,000,000	92.0000%
2	0x8343c2...6b0DA112	65,707,950,000,000	5.0000%
3	0xb153dA...0697BccD	39,424,770,000,000	3.0000%

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.