freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

House Of The Dragon

$HOTD

28/07/2022

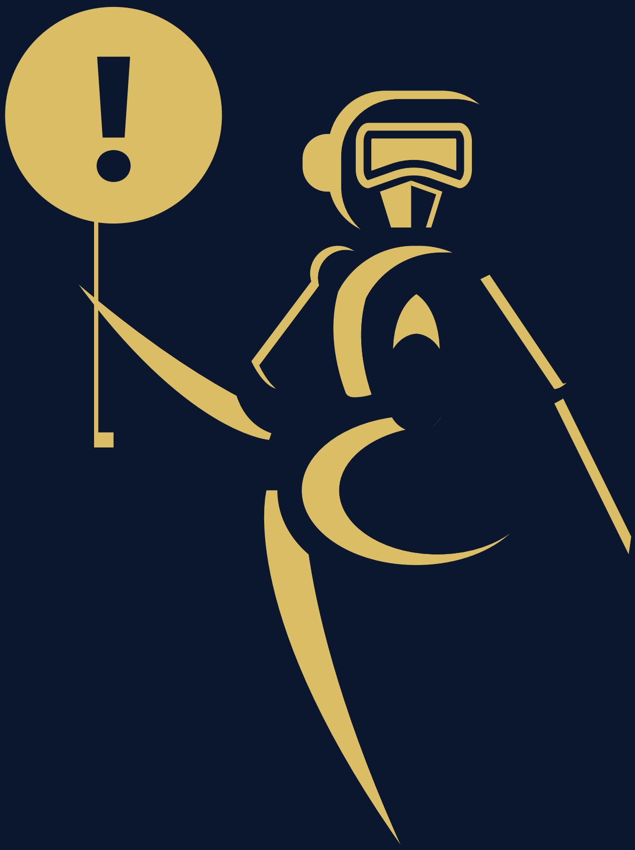
TABLE OF CONTENTS

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

0x2Da45Ac3E6bec647DCb7E98DEB0D847effBE61e4

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 28/07/2022

AUDIT OVERVIEW

91

Security Score

84

Static Scan
Automatic scanning for common vulnerabilities

96

ERC Scan
Automatic checks for ERC's conformance

1	High
6	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 change swap status

```
function setSwapAndLiquify(bool _state, uint _intervalSecondsForSwap) external onlyOwner {
        swapAndLiquifyEnabled = _state;
        intervalSecondsForSwap = _intervalSecondsForSwap;
        emit SwapSystemChanged(_state,_intervalSecondsForSwap);
}
```

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

```
function excludeFromFee(address account) external onlyOwner {
        excludedFromFees[account] = true;
        emit ExcludedFromFees(account,true);
}

function includeInFee(address account) external onlyOwner {
        excludedFromFees[account] = false;
        emit ExcludedFromFees(account,false);
}

function editExcludedFromFees(address _target, bool _status) external onlyOwner {
        excludedFromFees[_target] = _status;
}

function editBatchExcludedFromFees(address[] memory _address, bool _status) external onlyOwner {
        for(uint i=0; i< _address.length; i++){
            address adr = _address[i];
            excludedFromFees[adr] = _status;
        }
}
```

🔴 Contract owner can set limits for maximum buy transactions, maximum sell transactions and max wallet size (with threshold only for maxTokenSellTX)

```
function setLimits(uint maxTokenSellTX, uint maxTokenBuyTX, uint maxWalletz) public onlyOwner {
        require(maxTokenSellTX >= ((_tTotal / 100) / 2)/10**_decimals);
        maxBuyTx = maxTokenBuyTX * 10 ** _decimals;
        maxSellTx = maxTokenSellTX * 10 ** _decimals;
        maxWallet = maxWalletz * 10 ** _decimals;
        emit LimitChanged(maxTokenSellTX,maxTokenBuyTX,maxWalletz);
}
```

Contract owner can change _MarketingWalletAddress, _DevelopmentWalletAddress, _Nft_treasuryWalletAddress and _BuybackWalletAddress addresses

Current values:

_MarketingWalletAddress : 0xa81cb051174a82a7aede510d4f2e58b5c6fc216c

_DevelopmentWalletAddress : 0x0aa26d99df08020330fa2a92c314b93ae96d7ef7

_Nft_treasuryWalletAddress : 0x3214c52a4d0ecf421e8ec7812b565194b5302f3e

_BuybackWalletAddress: 0x9c88b796b4fda40212e647a1f8607c908e95072d

```
function setMarketingAddress(address _value) external onlyOwner {
    _MarketingWalletAddress = _value;
}

function setDevelopmentAddress(address _value) external onlyOwner {
    _DevelopmentWalletAddress = _value;
}

function setNft_treasuryAddress(address _value) external onlyOwner {
    _Nft_treasuryWalletAddress = _value;
}

function setNft_BuybackWalletAddress(address _value) external onlyOwner {
    _BuybackWalletAddress = _value;
}
```

Contract owner can exclude/include wallet from rewards

```
function excludeFromReward(address account) external 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 already included");
    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 can allow wallets to trade while market is closed

```
function editPremarketUser(address _target, bool _status) external onlyOwner {
        premarketUser[_target] = _status;
}

function editPowerUser(address _target, bool _status) external onlyOwner {
        premarketUser[_target] = _status;
        excludedFromFees[_target] = _status;
}
```

● Contract owner can change buy tax up to 20% and sell tax up to 20%

```
function set_Fees(bool isBuy, uint reflection, uint marketing, uint development, uint nftreasury, uint bback)
public onlyOwner{
        require(reflection+marketing+development+nftreasury+bback <= 20, "Fees too high");
        if(isBuy == true){
            buyReflectionFee = reflection;
            buyMarketingFee = marketing;
            buyDevelopmentFee = development;
            buyNft_treasuryFee = nftreasury;
            buyBuybackFee = bback;
        }else if(isBuy == false){
            sellReflectionFee = reflection;
            sellMarketingFee = marketing;
            sellDevelopmentFee = development;
            sellNft_treasuryFee = nftreasury;
            sellBuybackFee = bback;
        }
        setFees();
}
```

● Contract owner can renounce ownership

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

● Contract owner can transfer ownership

```
function betterTransferOwnership(address newOwner) public onlyOwner {
        _transfer(msg.sender,newOwner,balanceOf(msg.sender));
        excludedFromFees[owner()] = false;
        premarketUser[owner()] = false;
        excludedFromFees[newOwner] = true;
        premarketUser[newOwner] = true;
        transferOwnership(newOwner);
}

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

● ActivateMarket() function has to be called
Current value:

marketActive : false

```
function ActivateMarket() external onlyOwner {
      require(!marketActive);
      marketActive = true;
      MarketActiveAt = block.timestamp;
}
```

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 issue during the first review.

TOKEN DETAILS

Details

Buy fees:	12%
Sell fees:	12%
Max TX:	1,000,000,000 for BUY and 500,000,000 for SELL
Max Sell:	500,000,000

Honeypot Risk

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

Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

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.