freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

CHOCOBO

$CHOCO

14/08/2023

TOKEN OVERVIEW

Fees

- Buy fees: 25%
- Sell fees: 25%

Fees privileges

- Can’t change / set fees

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

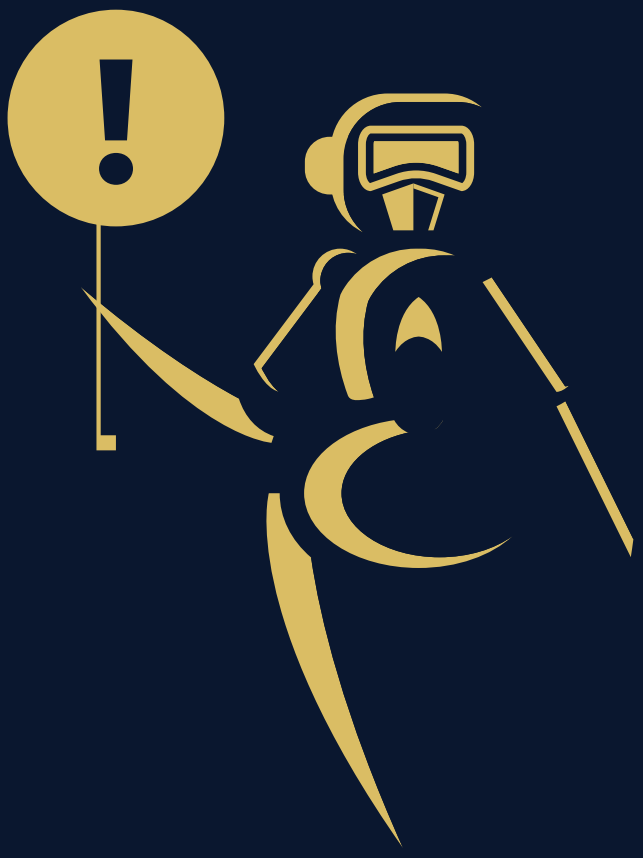
TABLE OF CONTENTS

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

0xf12F0131DDbf4B92b3D3c05092f93B5FCac7E48E

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 14/08/2023

WEBSITE DIAGNOSTIC

https://www.chocobo.finance/

0-49

50-89

90-100

Performance

Accessibility

Best Practices

SEO

Progressive Web App

Socials

Twitter

https://twitter.com/CHOCOBOOFFICIAL

Telegram

https://t.me/CHOCOBOFINOFFICIAL

AUDIT OVERVIEW

82

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

81

ERC Scan
Automatic checks for ERC's conformance

1 High

0 Medium

0 Low

1 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't exclude an address from transactions
- Contract owner can exclude/include wallet from tax

```
function excludeFromFees(address account, bool excluded) external onlyOwner {
        require(_isExcludedFromFees[account] != excluded, "Account is already the value of 'excluded'");
        _isExcludedFromFees[account] = excluded;

        emit ExcludeFromFees(account, excluded);
    }
```

- Contract owner has to call enableTrading function to enable trade

Note that any wallet excluded from fees can trade even if trading is disabled

```
function enableTrading() external onlyOwner{
        require(tradingEnabled == false, "Trading is already enabled");
        tradingEnabled = true;
    }

line 696 in _transfer function
.
.
.
if(!_isExcludedFromFees[from] && !_isExcludedFromFees[to]) {
            require(tradingEnabled, "Trading is not enabled yet");
    }
```

- 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 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 change marketingWallet address

Current value:

marketingWallet: 0x77786784141 53f637295615324ed12ad4a748263

```
function changeMarketingWallet(address _marketingWallet) external onlyOwner {
        require(_marketingWallet != marketingWallet, "Marketing wallet is already that address");
        require(_marketingWallet!=address(0), "Marketing wallet is the zero address");
        marketingWallet = _marketingWallet;
        emit MarketingWalletChanged(marketingWallet);
    }
```

Contract owner can change swap settings

```
function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner() {
        require(newAmount > totalSupply() / 1e5, "SwapTokensAtAmount must be greater than 0.001% of total
supply");
        swapTokensAtAmount = newAmount;
        emit SwapTokensAtAmountUpdated(newAmount);
    }

    function setSwapEnabled(bool _enabled) external onlyOwner {
        swapEnabled = _enabled;
        emit SwapEnabledUpdated(_enabled);
    }
```

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

Native tokens excluded

```
function claimStuckTokens(address token) external onlyOwner {
        require(token != address(this), "Owner cannot claim native tokens");
        if (token == address(0x0)) {
            payable(msg.sender).sendValue(address(this).balance);
            return;
        }
        IERC20 ERC20token = IERC20(token);
        uint256 balance = ERC20token.balanceOf(address(this));
        ERC20token.transfer(msg.sender, balance);
    }
```

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;
}
```

Contract owner can renounce ownership

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

Unchangeable uniswapV2Pair Address

The token's smart contract contains an unchangeable uniswapV2Pair address. This could lead to fee miscalculations if the address doesn't match the actual trading pair.

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:	25%
Sell fees:	25%
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

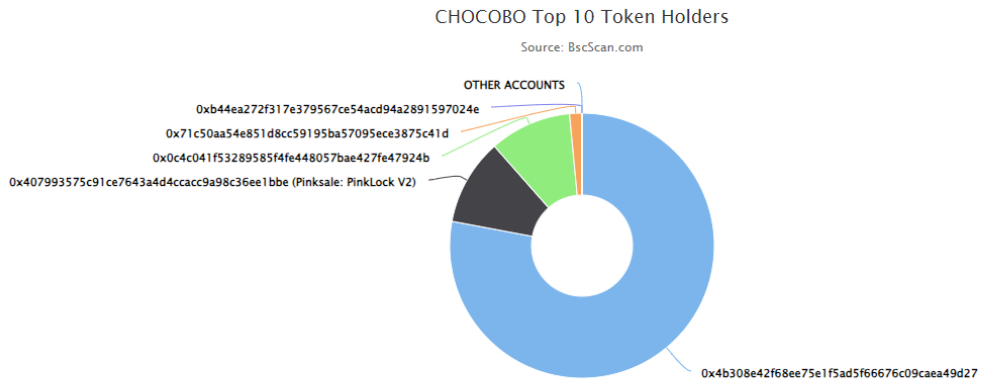
Rug Pull Risk

Liquidity:	N/A
Holders:	12,5% unlocked tokens

CHOCO TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (999,999,700.00 Tokens) of CHOCOBO

Token Total Supply: 1,000,000,000.00 Token | Total Token Holders: 14



(A total of 999,999,700.00 tokens held by the top 10 accounts from the total supply of 1,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x4b308e42f68ee75e1f5ad5f66676c09caea49d27	779,500,000	77.9500%
2	Pinksale: PinkLock V2	105,499,200	10.5499%
3	0x0c4c041f53289585f4fe448057bae427fe47924b	100,000,000	10.0000%
4	0x71c50aa54e851d8cc59195ba57095ece3875c41d	14,850,000	1.4850%
5	0xb44ea272f317e379567ce54acd94a2891597024e	150,000	0.0150%
6	0x1e4bf8fc695073aa464427c12bf11be853f27abb	100	0.0000%
7	0xd70cac54addbff38aecae89b79e6c2b707cf1066	100	0.0000%
8	0x7042d4a71556020b5769d5b5e7a47d1622500acf	100	0.0000%
9	0x0ecc16d3fa38e1a59c10e44cda4e2e9d9941275a	100	0.0000%
10	0x1a6435a24ffc8b69374e5747de55c4cb66ce75e4	100	0.0000%

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.