freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Open Network

$OPEN

29/04/2023

TOKEN OVERVIEW

Fees

- Buy fees: 9%
- Sell fees: 9%

Fees privileges

- Can change buy fees up to 99% and sell fees up to 99%

Ownership

- Owned

Minting

- No mint function

Max Tx Amount / Max Wallet Amount

- Can change / set max tx amount or max wallet amount (with threshold)

Blacklist

- Blacklist function not detected

Other privileges

- Can exclude / include from fees

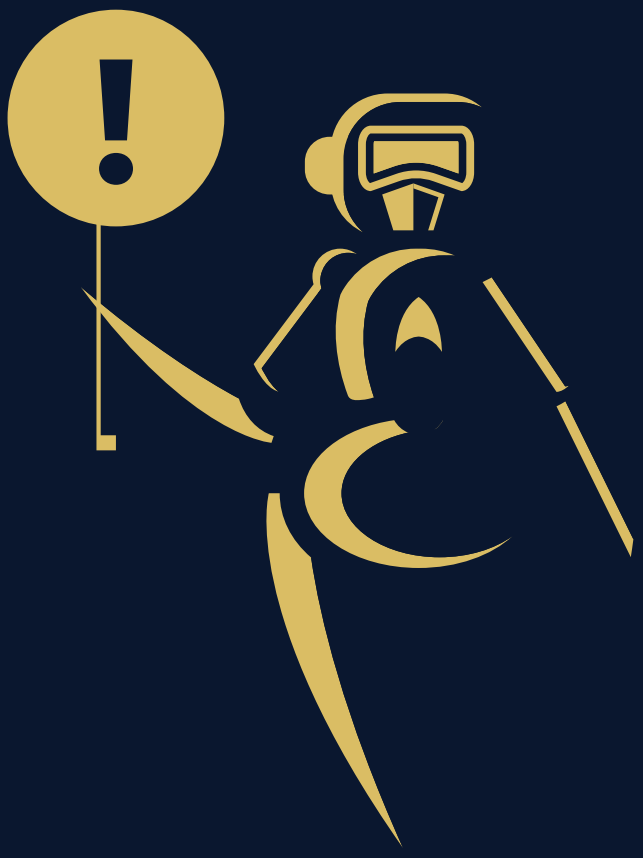
TABLE OF CONTENTS

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

0xE6bD3221fd849523a62EF7dc265Cbc30caA69948

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 29/04/2023

AUDIT OVERVIEW

86

Security Score

94

Static Scan
Automatic scanning for common vulnerabilities

82

ERC Scan
Automatic checks for ERC's conformance

1 High

3 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	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) public onlyOwner {
    _isExcludedFromFees[account] = excluded;
    emit ExcludeFromFees(account, excluded);
}
```

- Contract owner can exclude/include wallet from tx limitations

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

- Contract owner can change swap settings

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

function updateSwapTokensAtAmount(uint256 newAmount)
    external
    onlyOwner
    returns (bool)
  {
    require(
      newAmount >= (totalSupply() * 1) / 100000,
      "Swap amount cannot be lower than 0.001% total supply."
    );
    require(
      newAmount <= (totalSupply() * 5) / 1000,
      "Swap amount cannot be higher than 0.5% total supply."
    );
    swapTokensAtAmount = newAmount;
    return true;
}
```

- Contract owner has to call `enableTrading` function to enable trade

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

Contract owner can change buy fees up to 99% and sell fees up to 99%

```
function updateBuyFees(
        uint256 _marketingFee,
        uint256 _liquidityFee,
        uint256 _devFee
    ) external onlyOwner {
        buyMarketingFee = _marketingFee;
        buyLiquidityFee = _liquidityFee;
        buyDevFee = _devFee;
        buyTotalFees = buyMarketingFee + buyLiquidityFee + buyDevFee;
        require(buyTotalFees <= 99, "Must keep fees at 99% or less");
    }

    function updateSellFees(
        uint256 _marketingFee,
        uint256 _liquidityFee,
        uint256 _devFee
    ) external onlyOwner {
        sellMarketingFee = _marketingFee;
        sellLiquidityFee = _liquidityFee;
        sellDevFee = _devFee;
        sellTotalFees = sellMarketingFee + sellLiquidityFee + sellDevFee;
        require(sellTotalFees <= 99, "Must keep fees at 99% or less");
    }
```

Contract owner can remove all limits (transfers delay, tx limitations, wallet limitations)

```
function removeLimits() external onlyOwner returns (bool) {
        limitsInEffect = false;
        return true;
    }
```

Contract owner can remove delay between trades limitation

Only one purchase per block allowed

```
function disableTransferDelay() external onlyOwner returns (bool) {
        transferDelayEnabled = false;
        return true;
    }
```

Contract owner can change max wallet amount (with threshold)

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

Contract owner can change max tx amount (with threshold)

```
function updateMaxTxnAmount(uint256 newNum) external onlyOwner {
    require(
      newNum >= ((totalSupply() * 1) / 1000) / 1e18,
      "Cannot set maxTransactionAmount lower than 0.1%"
    );
    maxTransactionAmount = newNum * (10**18);
}
```

Contract owner can change marketingWallet and devWallet addresses

Current values:

marketingWallet : 0x255b97ff680341ee5c9bdbce3290a4420758b6d6

devWallet : 0xe447f80c1a1f75b1e812785478c7d19421be9ff0

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

function updateDevWallet(address newWallet) external onlyOwner {
    emit devWalletUpdated(newWallet, devWallet);
    devWallet = newWallet;
}
```

Contract owner can transfer ownership

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

function _transferOwnership(address newOwner) internal virtual {
    address oldOwner = _owner;
    _owner = newOwner;
    emit OwnershipTransferred(oldOwner, newOwner);
}
```

Contract owner can renounce ownership

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

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:	9%
Sell fees:	9%
Max TX:	20,000,000,000,000
Max Sell:	N/A

Honeypot Risk

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

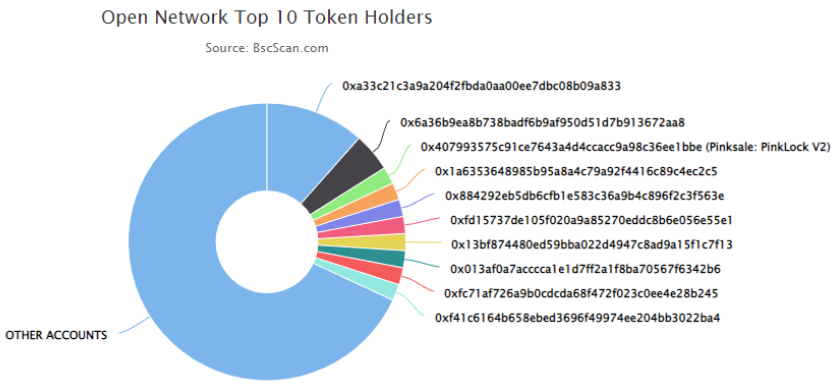
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

OPEN TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 31.91% (319,137,790,471,170.00 Tokens) of Open Network

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 545



(A total of 319,137,790,471,170.00 tokens held by the top 10 accounts from the total supply of 1,000,000,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0xa33c21c3a9a204f2fbda0aa00ee7dbc08b09a833	115,488,735,007,074.494082006123456112	11.5489%
2	0x6a36b9ea8b738badf6b9af950d51d7b913672aa8	45,062,195,837,188.255162168142478566	4.5062%
3	Pinksale: PinkLock V2	20,000,000,000,000	2.0000%
4	0x1a6353648985b95a8a4c79a92f4416c89c4ec2c5	19,994,048,374,160.278843259453249353	1.9994%
5	0x884292eb5db6cfb1e583c36a9b4c896f2c3f563e	19,970,128,956,084.582782401128143908	1.9970%
6	0xfd15737de105f020a9a85270eddc8b6e056e55e1	19,894,875,160,677.700872440309227984	1.9895%
7	0x13bf874480ed59bba022d4947c8ad9a15f1c7f13	19,838,521,056,337.827277127506756195	1.9839%
8	0x013af0a7acccca1e1d7ff2a1f8ba70567f6342b6	19,837,999,999,999.99942113071808381	1.9838%
9	0xfc71af726a9b0cdcda68f472f023c0ee4e28b245	19,825,645,516,927.198079511866555185	1.9826%
10	0xf41c6164b658ebed3696f49974ee204bb3022ba4	19,225,640,562,720.078299711731346239	1.9226%

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.