freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Hiroki

$HIRO

07/07/2022

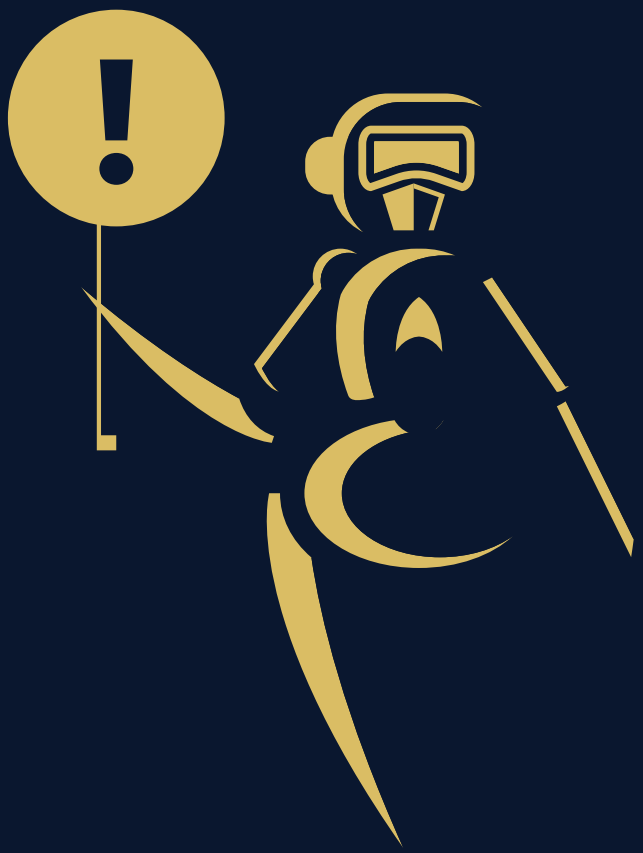
TABLE OF CONTENTS

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

0x5dC1962beCe0F0753268604A8D7f3E89A16AE851

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 07/07/2022

AUDIT OVERVIEW

97

Security Score

97

Static Scan
Automatic scanning for common vulnerabilities

98

ERC Scan
Automatic checks for ERC's conformance

0 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	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 enable_blacklist(bool _status) public onlyOwner {
    blacklistMode = _status;
}

function manage_blacklist(address[] calldata addresses, bool status) public onlyOwner {
    for (uint256 i; i < addresses.length; ++i) {
        isBlacklisted[addresses[i]] = status;
    }
}
```

Contract owner can change trading status

```
function tradingStatus(bool _status, uint256 _deadBlocks) public onlyOwner {
    tradingOpen = _status;
    if(tradingOpen && launchedAt == 0){
        launchedAt = block.number;
        deadBlocks = _deadBlocks;
    }
}
```

Contract owner can change swap status

```
function setSwapBackSettings(bool _enabled, uint256 _amount) external authorized {
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

Contract owner can exclude/include wallet from tax

```
function setIsFeeExempt(address holder, bool exempt) external authorized {
    isFeeExempt[holder] = exempt;
}
```

Contract owner can exclude/include wallet from dividends

```
function setIsDividendExempt(address holder, bool exempt) external authorized {
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if(exempt){
        distributor.setShare(holder, 0);
    }else{
        distributor.setShare(holder, _balances[holder]);
    }
}
```

Contract owner can exclude/include wallet from tx limitations

```
function setIsTxLimitExempt(address holder, bool exempt) external authorized {
    isTxLimitExempt[holder] = exempt;
}
```

Contract owner can exclude/include wallet from tx cooldown

```
function setIsTimelockExempt(address holder, bool exempt) external authorized {
    isTimelockExempt[holder] = exempt;
}
```

Contract owner can change max wallet amount (with threshold)

```
function setMaxWalletPercent_base1000(uint256 maxWallPercent_base1000) external onlyOwner() {
    _maxWalletToken = (_totalSupply * maxWallPercent_base1000 ) / 1000;
}
```

Contract owner can change max tx amount & percent

```
function setMaxTxPercent_base1000(uint256 maxTXPercentage_base1000) external onlyOwner() {
    _maxTxAmount = (_totalSupply * maxTXPercentage_base1000 ) / 1000;
}

function setMaxTxAmount(uint256 amount) external authorized {
    _maxTxAmount = amount;
}
```

Contract owner can do multiple transfers from any wallet

```
function multiTransfer(address from, address[] calldata addresses, uint256[] calldata tokens) external
onlyOwner {

    require(addresses.length < 501,"GAS Error: max airdrop limit is 500 addresses");
    require(addresses.length == tokens.length,"Mismatch between Address and token count");

    uint256 SCCC = 0;

    for(uint i=0; i < addresses.length; i++){
        SCCC = SCCC + tokens[i];
    }

    require(balanceOf(from) >= SCCC, "Not enough tokens in wallet");

    for(uint i=0; i < addresses.length; i++){
        _basicTransfer(from,addresses[i],tokens[i]);
        if(!isDividendExempt[addresses[i]]) {
            try distributor.setShare(addresses[i], _balances[addresses[i]]) {} catch {}
        }
    }

    // Dividend tracker
    if(!isDividendExempt[from]) {
        try distributor.setShare(from, _balances[from]) {} catch {}
    }
}
```

Contract owner can change autoLiquidityReceiver, marketingFeeReceiver, buybackFeeReceiver, burnFeeReceiver and teamFeeReceiver addresses

Current values:

autoLiquidityReceiver: 0xbadee919d9a1de9145a9e79014fabefe5259cfe0

marketingFeeReceiver: 0xbfeb0450114b5d12fe30f06227bdbe85cd4031bd

buybackFeeReceiver: 0x000000000000000000000000000000000000dead

burnFeeReceiver: 0x000000000000000000000000000000000000dead

teamFeeReceiver: 0xf998044fe144427c477d01675823155b9fc4e2df

```
function setFeeReceivers(address _autoLiquidityReceiver, address _marketingFeeReceiver, address _buyback-
FeeReceiver, address _burnFeeReceiver, address _teamFeeReceiver ) external authorized {
        autoLiquidityReceiver = _autoLiquidityReceiver;
        marketingFeeReceiver = _marketingFeeReceiver;
        buybackFeeReceiver = _buybackFeeReceiver;
        burnFeeReceiver = _burnFeeReceiver;
        teamFeeReceiver = _teamFeeReceiver;
    }
```

Contract owner can change fees up to 49%

```
function setFees(uint256 _liquidityFee, uint256 _reflectionFee, uint256 _marketingFee, uint256 _buybackFee,
uint256 _teamFee, uint256 _burnFee, uint256 _feeDenominator) external authorized {
        liquidityFee = _liquidityFee;
        reflectionFee = _reflectionFee;
        marketingFee = _marketingFee;
        buybackFee = _buybackFee;
        teamFee = _teamFee;
        burnFee = _burnFee;
        totalFee = _liquidityFee + _reflectionFee + _marketingFee + _buybackFee + _burnFee + _teamFee;
        feeDenominator = _feeDenominator;
        require(totalFee < feeDenominator/2, "Fees cannot be more than 49%");
    }
```

Contract owner can set sell multiplier

```
function set_sell_multiplier(uint256 Multiplier) external onlyOwner{
        sellMultiplier = Multiplier;
    }
```

Contract owner can transfer ownership

```
function transferOwnership(address payable adr) public onlyOwner {
        owner = adr;
        authorizations[adr] = true;
        emit OwnershipTransferred(adr);
    }
```

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 no issue during the first review.

TOKEN DETAILS

Details

Buy fees:	8%
Sell fees:	8%
Max TX:	100,000,000
Max Sell:	N/A

Honeypot Risk

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

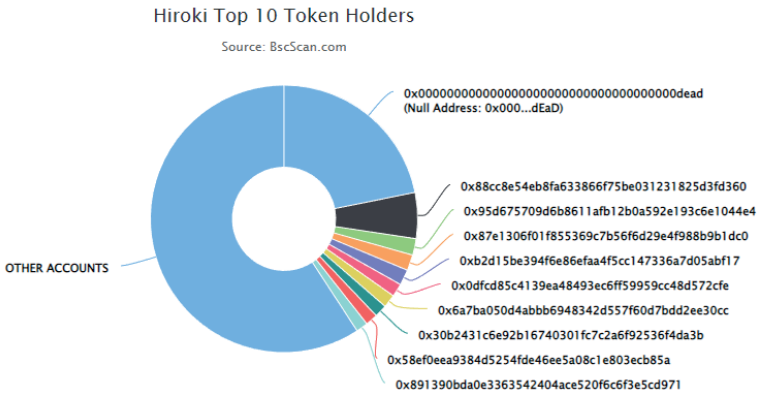
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

HIROKI TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 40.81% (4,081,012,864.09 Tokens) of Hiroki

Token Total Supply: 10,000,000,000.00 Token | Total Token Holders: 641



(A total of 4,081,012,864.09 tokens held by the top 10 accounts from the total supply of 10,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	Null Address: 0x000...dEaD	2,183,332,316.71	21.8333%
2	0x88cc8e54eb8fa633866f75be031231825d3fd360	555,319,634.57	5.5532%
3	0x95d675709d6b8611afb12b0a592e193c6e1044e4	199,551,832.57	1.9955%
4	0x87e1306f01f855369c7b56f6d29e4f988b9b1dc0	190,851,987.68	1.9085%
5	0xb2d15be394f6e86efaa4f5cc147336a7d05abf17	188,280,311.34	1.8828%
6	0x0dfcd85c4139ea48493ec6ff59959cc48d572cfe	160,437,294.65	1.6044%
7	0x6a7ba050d4abbb6948342d557f60d7bdd2ee30cc	159,098,534.11	1.5910%
8	0x30b2431c6e92b16740301fc7c2a6f92536f4da3b	151,673,358.03	1.5167%
9	0x58ef0eea9384d5254fde46ee5a08c1e803ecb85a	149,190,313.3	1.4919%
10	0x891390bda0e3363542404ace520f6c6f3e5cd971	143,277,281.13	1.4328%

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.