freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

NFTshit

$SHIT

06/02/2022

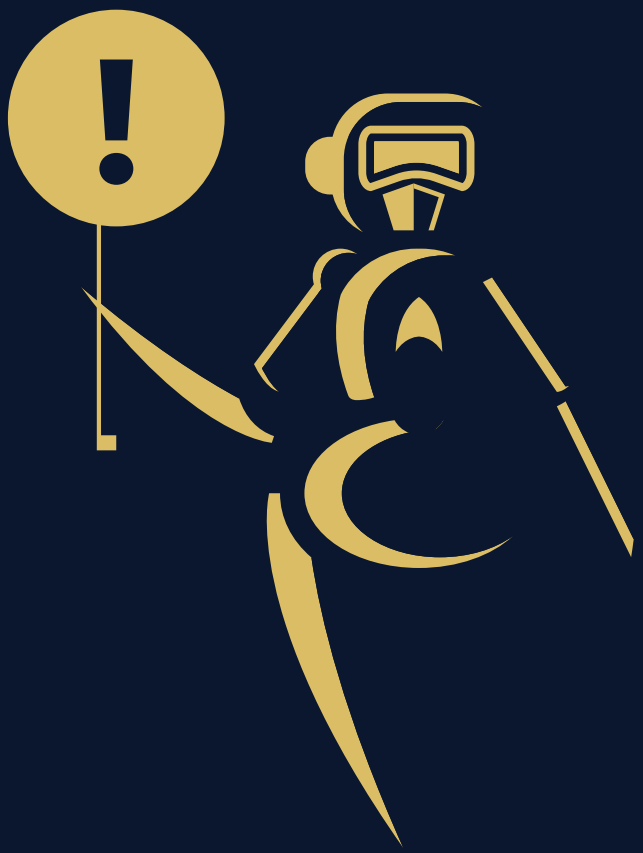
TABLE OF CONTENTS

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

0x8195a9233FA8Feff2492520EbA93Aa30dE42a70C

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 06/02/2022

AUDIT OVERVIEW

96

Security Score

94

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/include wallet from fees

```
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 already 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 can change swap settings

```
function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}
```

Contract owner can set max tx amount

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {
    require(maxTxAmount > 0, "Cannot set transaction amount as zero");
    _maxTxAmount = maxTxAmount * 10**_decimals;
}
```

Contract owner can change the fees up to 100%

```
function setBurnFeePercent(uint256 burnFee) external onlyOwner() {
        _burnFee = burnFee;
}

function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
        _taxFee = taxFee;
}

function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
        _liquidityFee = liquidityFee;
}

function setMarketingFeePercent(uint256 MarketingFee) external onlyOwner() {
        _marketingFee = MarketingFee;
}
```

Contract owner can change marketingWallet address

Current address:

marketingWallet: 0x8dda616329be9a77c6281d6d164d99edf0a6e6ba

```
function setMarketingWallet(address payable newWallet) external onlyOwner() {
        marketingWallet = newWallet;
}
```

Contract owner can change numTokensSellToAddToLiquidity

```
function setNumTokensSellToAddToLiquidity(uint256 newAmt) external onlyOwner() {
        numTokensSellToAddToLiquidity = newAmt*10**_decimals;
}
```

Contract owner can renounce ownership

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

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:	0%
Sell fees:	10%
Max TX:	N/A
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

NFTSHIT TOKEN DISTRIBUTION & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (50,000,000.00 Tokens) of NFTshit

Token Total Supply: 50,000,000.00 Token | Total Token Holders: 1

(A total of 50,000,000.00 tokens held by the top 10 accounts from the total supply of 50,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0xb6e4f7b0d64381a52532ddd5e2f45bca40bcaf30	50,000,000	100.0000%

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.