freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

NPC Coin

$NPC

13/08/2022

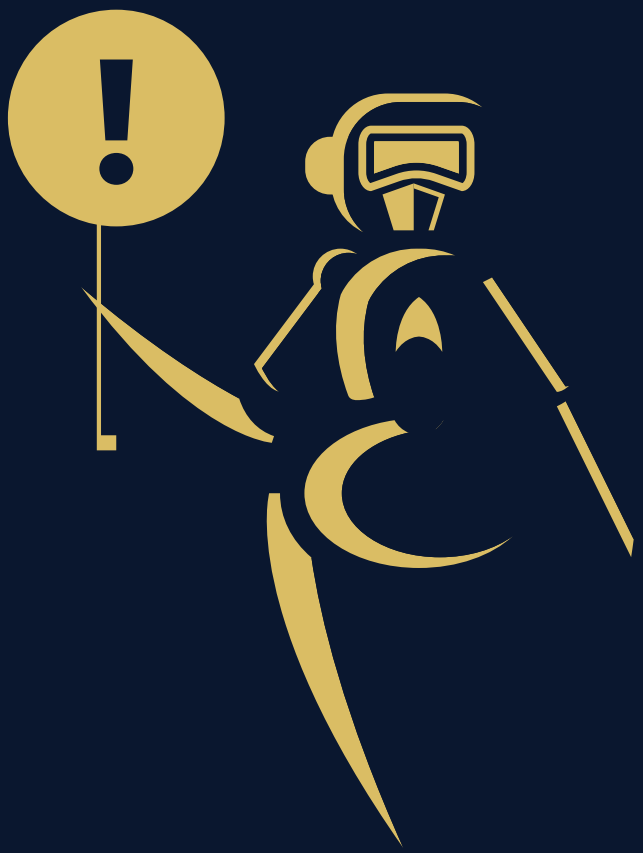
TABLE OF CONTENTS

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

0x86D6faDB5F7A0c069fB50F998e120835CaE15A54

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 13/08/2022

AUDIT OVERVIEW

92

Security Score

90

Static Scan
Automatic scanning for common vulnerabilities

96

ERC Scan
Automatic checks for ERC's conformance

0 High

2 Medium

2 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(s) from tax

```
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 change fees up to 10%

```
function setFeePercent(uint256 taxFee, uint256 liquidityFee, uint256 marketingFee, uint256 burnFee) external onlyOwner() {
        require(taxFee.add(liquidityFee).add(marketingFee).add(burnFee) <= 10, "tax too high");
        _taxFee = taxFee;
        _liquidityFee = liquidityFee;
        _marketingFee = marketingFee;
        _burnFee = burnFee;
    }
```

Contract owner can change max tx amount limitations

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {
        _maxTxAmount = maxTxAmount;
        require(_maxTxAmount > totalSupply().div(400), "value too low");
    }
```

Contract owner can change marketingWallet address

Current value:

marketingWallet : 0x543dfde9c6a7b914c0381602dde4e493d019cc02

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

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 HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	7%
Sell fees:	7%
Max TX:	1,000,000,000
Max Sell:	N/A

Honeypot Risk

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

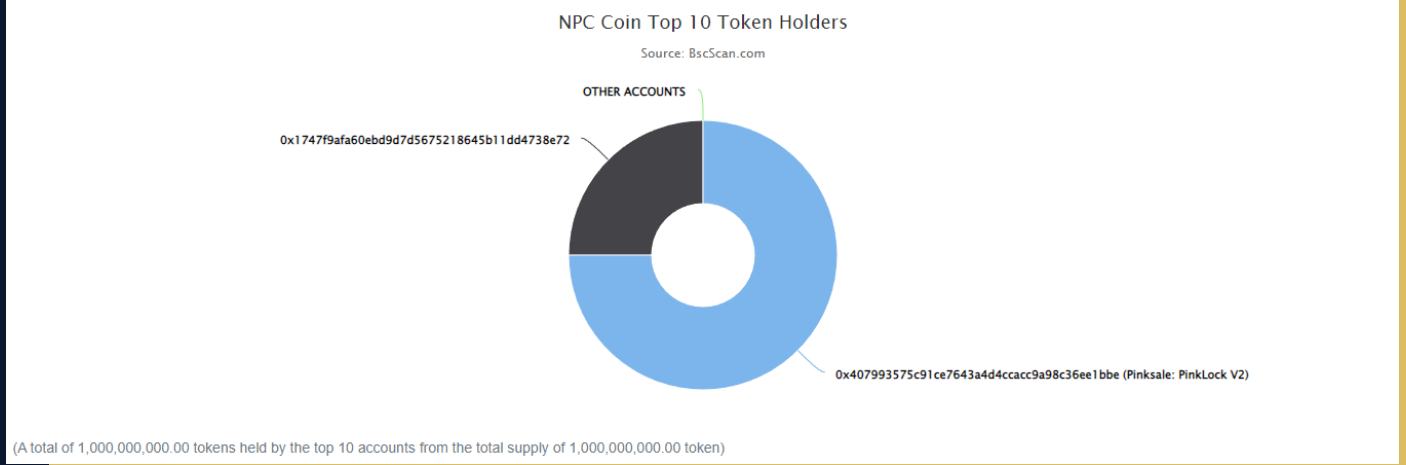
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

NPC COIN TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (1,000,000,000.00 Tokens) of NPC Coin

Token Total Supply: 1,000,000,000.00 Token | Total Token Holders: 2



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

Rank	Address	Quantity (Token)	Percentage
1	Pinksale: PinkLock V2	750,250,000	75.0250%
2	0x1747f9afa60ebd9d7d5675218645b11dd4738e72	249,750,000	24.9750%

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.