freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

NGA TIGER

$NGA

26/03/2023

TOKEN OVERVIEW

Fees

• Buy fees:	8%
• Sell fees:	8%

Fees privileges

• Can change buy fees up to 10% and sell fees up to 10%

Ownership

• Ownership Renounced

Minting

• No mint function

Max Tx Amount / Max Wallet Amount

• Can change max tx amount and max wallet amount (with threshold)

Blacklist

• Blacklist function not detected

Other privileges

• Can exclude / include from fees

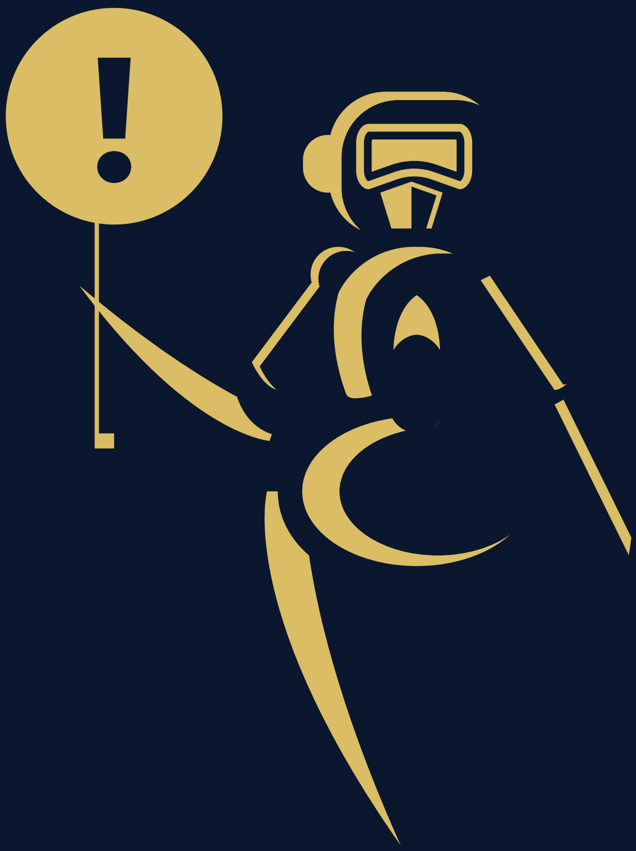
TABLE OF CONTENTS

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

0xAa3ED6E6Ea3Ed78D4d57E373aABD6f54DF5bb508

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 26/03/2023

WEBSITE DIAGNOSTIC

https://www.ngatiger.io/

🟥	🟨	🟩
0-49	50-89	90-100

91	93	92	90	NA
Performance	Accessibility	Best Practices	SEO	Progressive Web App

Socials

Twitter

https://twitter.com/NGATIGER_BSC

Telegram

https://t.me/NGATIGEROFFICIAL

AUDIT OVERVIEW

90

Security Score

98 **Static Scan**
Automatic scanning for common vulnerabilities

84 **ERC Scan**
Automatic checks for ERC's conformance

0	High
1	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(s) from tax (ownership renounced)

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
      _isExcludedFromFees[account] = excluded;
      emit ExcludeFromFees(account, excluded);
}

function batchAddComm(address[] calldata addresses) external onlyOwner {
      for (uint i = 0; i < addresses.length; i++) {
         excludeFromFees(addresses[i], true);
      }
}

function removeBatch(address[] calldata addresses) external onlyOwner {
      for (uint i = 0; i < addresses.length; i++) {
         if(addresses[i] != owner()) {
            excludeFromFees(addresses[i], false);
         }
      }
}
```

- Contract owner can exclude/include wallet from tx limitations (ownership renounced)

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

- Contract owner can withdraw tokens from smart contract (NGA excluded) (ownership renounced)

```
function withdraw() external onlyOwner {
      uint256 balance = IERC20(address(this)).balanceOf(address(this));
      IERC20(address(this)).transfer(msg.sender, balance);
      payable(msg.sender).transfer(address(this).balance);
   }

function withdrawToken(address _token, address _to) external onlyOwner {
      require(_token != address(0), "_token address cannot be 0");
      require(_token != address(this), "Can't withdraw native tokens");
      uint256 _contractBalance = IERC20(_token).balanceOf(address(this));
      IERC20(_token).transfer(_to, _contractBalance);
}
```

● The liquidity of the contract automatically gets credited into the owner's wallet whenever the 'addLiquidity' function is called inside the contract.

Note that it cannot be called manually but it will be done automatically every time the swap and liquify function is called. Moreover, even after the renouncement of the ownership, this liquidity will still be credited to the owner's wallet.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(uniswapV2Router), tokenAmount);

        // add the liquidity
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            owner(),
            block.timestamp
        );
}
```

● Contract owner can remove all limits (tx limitations, wallet limitations, etc)

(ownership renounced)

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

● Contract owner can change buy fees up to 10% and sell fees up to 10%

(ownership renounced)

```
function updateBuyFees(
        uint256 _marketingFee,
        uint256 _liquidityFee,
        uint256 _devFee
    ) external onlyOwner {
        buyMarketingFee = _marketingFee;
        buyLiquidityFee = _liquidityFee;
        buyDevFee = _devFee;
        buyTotalFees = buyMarketingFee + buyLiquidityFee + buyDevFee;
        require(buyTotalFees <= 10, "Must keep fees at 10% or less");
}

function updateSellFees(
        uint256 _marketingFee,
        uint256 _liquidityFee,
        uint256 _devFee
    ) external onlyOwner {
        sellMarketingFee = _marketingFee;
        sellLiquidityFee = _liquidityFee;
        sellDevFee = _devFee;
        sellTotalFees = sellMarketingFee + sellLiquidityFee + sellDevFee;
        require(sellTotalFees <= 10, "Must keep fees at 10% or less");
}
```

● Contract owner can change tx limitations and wallet limitations

(ownership renounced)

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

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 marketingWallet and devWallet addresses

(ownership renounced)

marketingWallet : 0x15c3eb64e7333f36382a99d15c5853b77b73fa99

devWallet : 0x258987d14821e85dd62b9b0bb65a527275122753

```
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

(ownership renounced)

```
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

(ownership renounced)

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

Contract owner can change swap settings

(ownership renounced)

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

function updateSwapTokensAtAmount(uint256 newAmount) external onlyOwner returns (bool) {
        require(newAmount >= (totalSupply() * 1) / 10000,
           "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;
}
```

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 no HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	8%
Sell fees:	8%
Max TX:	5,000,000,000,000,000,000,000,000,000
Max Sell:	N/A

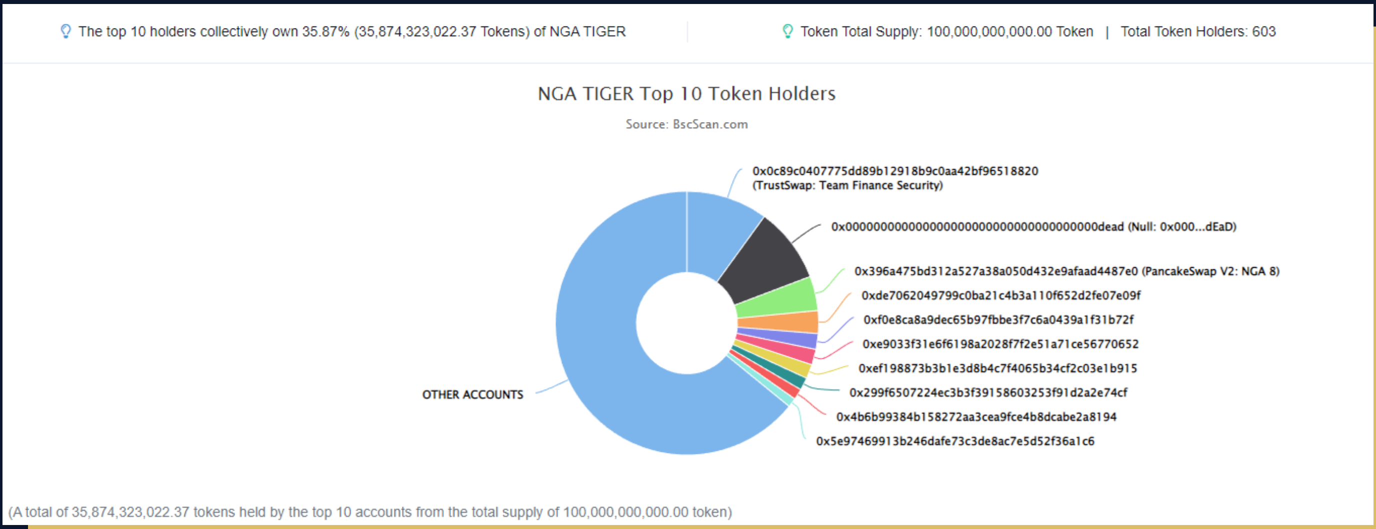
Honeypot Risk

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

Rug Pull Risk

Liquidity:	N/A
Holders:	Clear

NGA TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

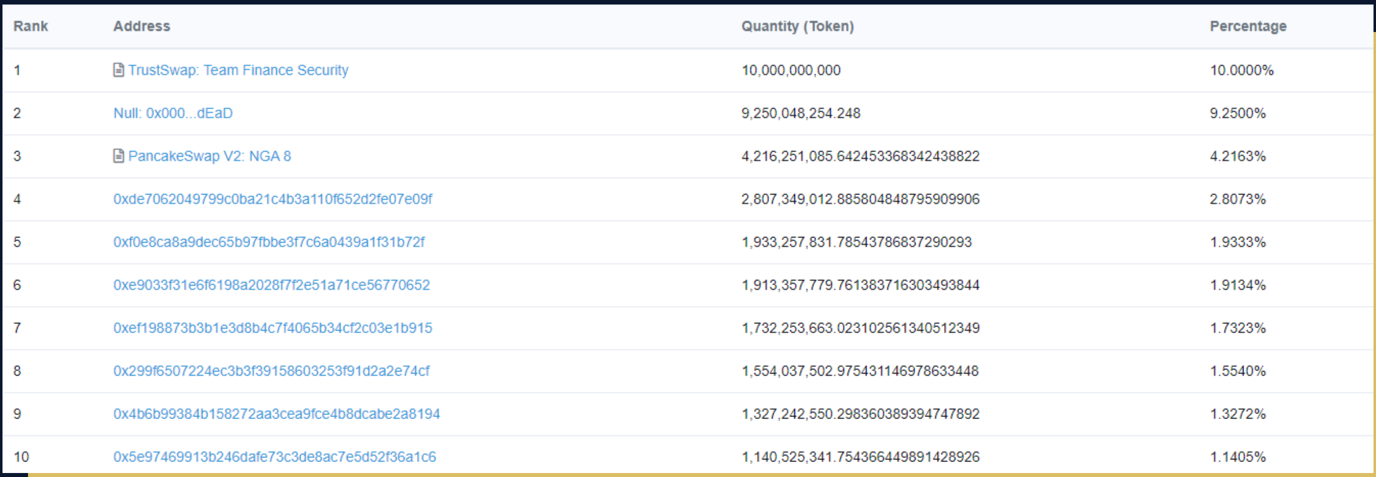


The top 10 holders collectively own 35.87% (35,874,323,022.37 Tokens) of NGA TIGER | Token Total Supply: 100,000,000,000.00 Token | Total Token Holders: 603

NGA TIGER Top 10 Token Holders

Source: BscScan.com

(A total of 35,874,323,022.37 tokens held by the top 10 accounts from the total supply of 100,000,000,000.00 token)



Rank	Address	Quantity (Token)	Percentage
1	TrustSwap: Team Finance Security	10,000,000,000	10.0000%
2	Null: 0x000...dEaD	9,250,048,254.248	9.2500%
3	PancakeSwap V2: NGA 8	4,216,251,085.642453368342438822	4.2163%
4	0xde7062049799c0ba21c4b3a110f652d2fe07e09f	2,807,349,012.885804848795909906	2.8073%
5	0xf0e8ca8a9dec65b97fbbe3f7c6a0439a1f31b72f	1,933,257,831.78543786837290293	1.9333%
6	0xe9033f31e6f6198a2028f7f2e51a71ce56770652	1,913,357,779.761383716303493844	1.9134%
7	0xef198873b3b1e3d8b4c7f4065b34cf2c03e1b915	1,732,253,663.023102561340512349	1.7323%
8	0x299f6507224ec3b3f39158603253f91d2a2e74cf	1,554,037,502.975431146978633448	1.5540%
9	0x4b6b99384b158272aa3cea9fce4b8dcabe2a8194	1,327,242,550.298360389394747892	1.3272%
10	0x5e97469913b246dafe73c3de8ac7e5d52f36a1c6	1,140,525,341.754366449891428926	1.1405%

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.