freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

BAKENEKO

$BAKENEKO

24/12/2024

TOKEN OVERVIEW

Fees

- Buy fees: 1%
- Sell fees: 1%

Fees privileges

- Can change buy fees up to 10%, sell fees up to 10% and transfer fees up to 10%

Ownership

- Owned

Minting

- No mint function

Max Tx Amount / Max Wallet Amount

- Can’t change max tx amount and / or max wallet amount

Blacklist

- Blacklist function not detected

Other privileges

- Can exclude/include wallet from fees

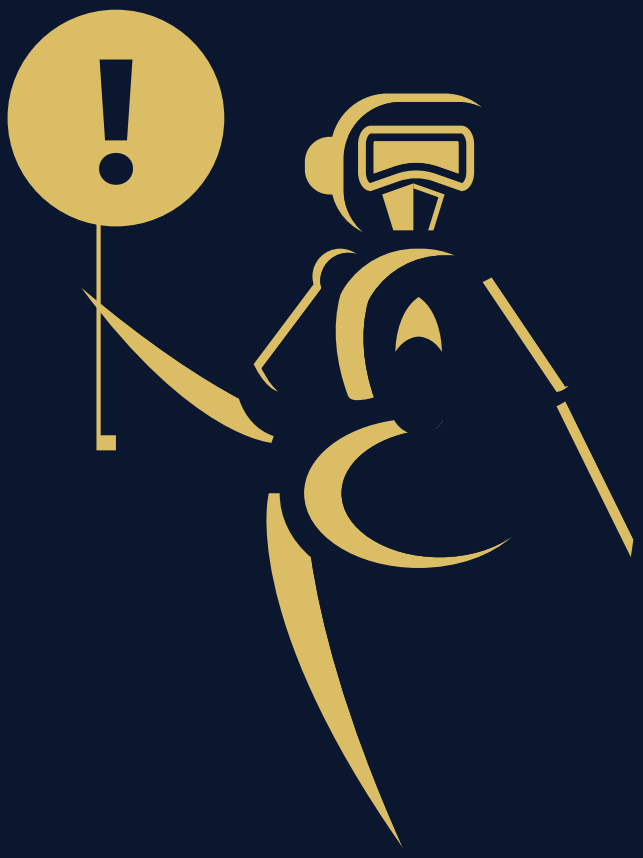
TABLE OF CONTENTS

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

0x0abeaf56546a563824c96bb93d1c46f1d4cd11eb

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 24/12/2024

WEBSITE DIAGNOSTIC

https://www.bakenekobsc.com/

0-49 50-89 90-100

97	96	95	92	NA
Performance	Accessibility	Best Practices	SEO	Progressive Web App

Socials

X (Twitter)

https://x.com/bakenekobsc

Telegram

https://t.me/bakenekobsc

AUDIT OVERVIEW

94

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

92

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	Low
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 setExcludeFromFees(
        address account,
        bool excluded
    ) external onlyOwner {
        require(
            _isExcludedFromFees[account] != excluded,
            "Account is already the value of 'excluded'"
        );
        _isExcludedFromFees[account] = excluded;
        emit UpdateExcludeFromFees(account, excluded);
    }
```

- Contract owner can change marketingWallet address

Current value:

marketingWallet: 0x7875824C8418B8e1f16CBBeE86A9fb81Bf42D6a5

```
function setMarketingWallet(address _marketingWallet) external onlyOwner {
        require(
            _marketingWallet != marketingWallet,
            "Marketing wallet is already that address"
        );
        require(
            _marketingWallet != address(0),
            "Marketing wallet cannot be the zero address"
        );
        require(
            !isContract(_marketingWallet),
            "Marketing wallet cannot be a contract"
        );
        marketingWallet = _marketingWallet;
        emit UpdateMarketingWallet(_marketingWallet);
    }
```

- Contract owner can set buy fees up to 10%, sell fees up to 10% and transfer fees up to 10%

```
function updateBuyFees(
        uint256 _liquidityFee,
        uint256 _marketingTax
    ) external onlyOwner {
        require(
            _liquidityFee + _marketingTax <= 1000,
            "Total fees cannot be more than 10%"
        );

        liquidityFeeBuy = _liquidityFee;
        marketingTaxBuy = _marketingTax;

        emit UpdateBuyFees(_liquidityFee, _marketingTax);
}

function updateSellFees(
        uint256 _liquidityFee,
        uint256 _marketingTax

    ) external onlyOwner {
        require(
            _liquidityFee + _marketingTax <= 1000,
            "Total fees cannot be more than 10%"
        );

        liquidityFeeSell = _liquidityFee;
        marketingTaxSell = _marketingTax;

        emit UpdateSellFees(_liquidityFee, _marketingTax);
}

function updateTransferFees(
        uint256 _liquidityFee,
        uint256 _marketingTax
    ) external onlyOwner {
        require(
            _liquidityFee + _marketingTax <= 1000,
            "Total fees cannot be more than 10%"
        );

        liquidityFeeTransfer = _liquidityFee;
        marketingTaxTransfer = _marketingTax;

        emit UpdateTransferFees(_liquidityFee, _marketingTax);
}
```

- Contract owner can renounce ownership

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

Contract owner can change swap settings

```
function setSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(
      swapTokensAtAmount != amount,
      "SwapTokensAtAmount already on that amount"
    );
    require(amount >= 1, "Amount must be equal or greater than 1 Wei");

    swapTokensAtAmount = amount;

    emit UpdateSwapTokensAtAmount(amount);
}

function toggleSwapBack(bool status) external onlyOwner {
    require(isSwapBackEnabled != status, "SwapBack already on status");

    isSwapBackEnabled = status;
    emit UpdateSwapBackStatus(status);
}
```

Contract owner has ability to retrieve any token held by the contract

Native tokens excluded

```
function claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "Owner cannot claim native tokens");

    if (token == address(0x0)) {
      payable(msg.sender).transfer(address(this).balance);
      return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.safeTransfer(msg.sender, balance);
}
```

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);
}
```

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:	1%
Sell fees:	1%
Max TX:	N/A
Max Sell:	N/A

Honeypot Risk

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

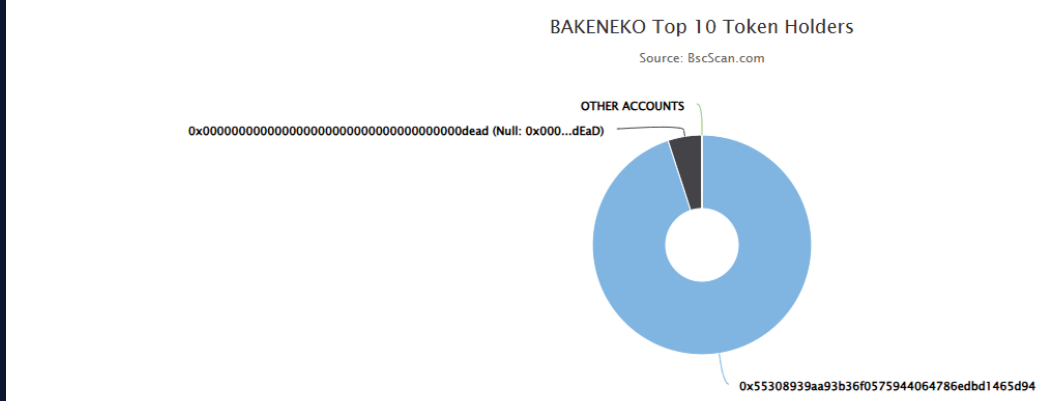
Rug Pull Risk

Liquidity:	N/A
Holders:	95% unlocked tokens

BAKENEKO TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (690,000,000,000.00 Tokens) of BAKENEKO

Token Total Supply: 690,000,000,000.00 Token | Total Token Holders: 2



(A total of 690,000,000,000.00 tokens held by the top 10 accounts from the total supply of 690,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x55308939...bd1465d94	655,500,000,000	95.0000%
2	Null: 0x000...dEaD	34,500,000,000	5.0000%

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.