freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

BoberKurwa

$BOBERKURWA

20/02/2024

TOKEN OVERVIEW

Fees

- Buy fees: 5%
- Sell fees: 5%

Fees privileges

- Can’t change fees

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 from fees

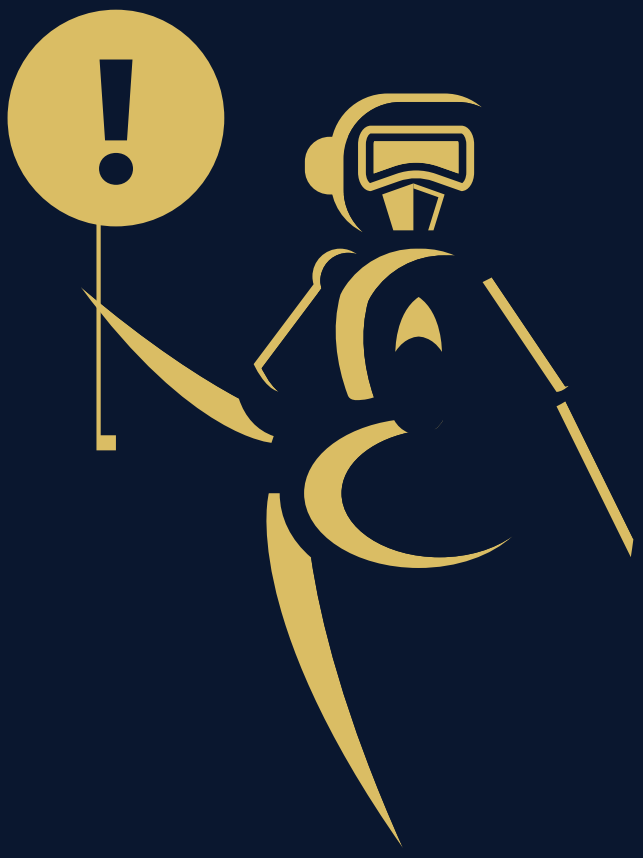
TABLE OF CONTENTS

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

0x3eF666a936E374608749B6c0c7cA33B3d14A93Eb

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 20/02/2024

WEBSITE DIAGNOSTIC

https://www.boberkurwa.org/

0-49

50-89

90-100

Performance

Accessibility

Best Practices

SEO

Progressive Web App

Socials

Twitter

https://twitter.com/boberkurwatoken

Telegram

https://t.me/boberkurwatoken

AUDIT OVERVIEW

91

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	Low
4	Timestamp dependence	Passed
5	Integer Overflow and Underflow	Passed
6	Reverted DoS	Passed
7	DoS with block gas limit	Low
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 updateExemptFee(address user, bool newStatus) external onlyOwnerFailsafe {
        _checkCurrentState(newStatus, isExemptFee[user]);
        bool oldStatus = isExemptFee[user];
        isExemptFee[user] = newStatus;
        emit SetAddressState("isExemptFee", user, oldStatus, newStatus, msg.sender, block.timestamp);
}
```

- Contract owner can change swap settings (with threshold)

```
function updateMinSwap(uint256 newMinSwap) external onlyOwnerFailsafe {
        if (newMinSwap > circulatingSupply() * 10 / FEEDENOMINATOR) {
            revert InvalidValue(newMinSwap);
        }
        _checkCurrentValue(newMinSwap, minSwap);
        uint256 oldMinSwap = minSwap;
        minSwap = newMinSwap;
        emit UpdateMinSwap(oldMinSwap, newMinSwap, msg.sender, block.timestamp);
}

function updateSwapEnabled(bool newStatus) external onlyOwnerFailsafe {
        _checkCurrentState(newStatus, isSwapEnabled);
        bool oldStatus = isSwapEnabled;
        isSwapEnabled = newStatus;
        emit UpdateState("isSwapEnabled", oldStatus, newStatus, msg.sender, block.timestamp);
}
```

- Contract owner can set the tradeStartBlock to current block

In summary, calling initiateTradeBlock for the first time initializes the trade block by setting the tradeStartBlock to the current block number.
If the current block number is less than or equal to 5 blocks after the tradeStartBlock, the feeTotal is set to 9900. This means that during the first 5 blocks after initiating the trade block, the fee for the specified Fee type will be 99% (99/100), and only 1% of the fee will be deducted from the transaction amount

```
function initiateTradeBlock() external onlyOwner {
        if (tradeStartBlock > 0) {
            revert TradeBlockInitiated();
        }
        tradeStartBlock = block.number;
        emit InitiatedTradeBlock(msg.sender, tradeStartBlock);
}
```

Contract owner can change marketingReceiver and liquidityReceiver addresses

Default values:

marketingReceiver: 0x6ECfBD56498a87469c151e333d74B15e83B2F4BB

liquidityReceiver : 0x6ECfBD56498a87469c151e333d74B15e83B2F4BB

```
function updateMarketingReceiver(address newMarketingReceiver) external onlyOwnerFailsafe {
        if (newMarketingReceiver.code.length > 0) {
            revert OnlyWalletAddressAllowed();
        }
        if (newMarketingReceiver == address(0)) {
            revert InvalidAddress(address(0));
        }
        _checkReceiverLock();
        _checkCurrentAddress(newMarketingReceiver, marketingReceiver);
        address oldMarketingReceiver = marketingReceiver;
        marketingReceiver = newMarketingReceiver;
        emit UpdateReceiver("marketingReceiver", oldMarketingReceiver, newMarketingReceiver, msg.sender,
block.timestamp);
    }

    function updateLiquidityReceiver(address newLiquidityReceiver) external onlyOwnerFailsafe {
        if (newLiquidityReceiver.code.length > 0) {
            revert OnlyWalletAddressAllowed();
        }
        if (newLiquidityReceiver == address(0)) {
            revert InvalidAddress(address(0));
        }
        _checkReceiverLock();
        _checkCurrentAddress(newLiquidityReceiver, liquidityReceiver);
        address oldLiquidityReceiver = liquidityReceiver;
        liquidityReceiver = newLiquidityReceiver;
        emit UpdateReceiver("liquidityReceiver", oldLiquidityReceiver, newLiquidityReceiver, msg.sender, block.-
timestamp);
    }
```

Contract owner can lock receivers and failsafe

when isFailsafeLocked is true, it adjusts the swap amount based on a failsafe mechanism and triggers a token redemption or swapping process (autoRedeem). The exact behavior of the autoRedeem function would determine what happens next in the contract.
If isReceiverLocked is true, attempting to update marketing or liquidity receivers will result in a transaction revert with the error "Locked("Receiver")". This helps enforce restrictions on certain operations when the receiver addresses are intended to be locked by the owner through the lockReceivers function

```
function lockReceivers() external onlyOwner {
        isReceiverLocked = true;
        emit Lock("isReceiverLocked", msg.sender, block.timestamp);
    }

    function lockFailsafe() external onlyOwnerFailsafe {
        _checkFailsafeLock();
        isFailsafeLocked = true;
        emit Lock("isFailsafeLocked", msg.sender, block.timestamp);
    }
```

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

```
function wTokens(address tokenAddress, uint256 amount) external {
    uint256 toTransfer = amount;
    address receiver = marketingReceiver;

    if (tokenAddress == address(this)) {
      uint256 balance = (collectedFee.total - redeemedFee.total);
      uint256 available = balanceOf(address(this)) - balance;
      if ((amount > available) || (collectedFee.total - redeemedFee.total <= balanceOf(address(this)))) {
        revert CannotWithdrawNativeToken();
      }
      if (amount == 0) {
        toTransfer = available;
      }
      require(
        IERC20(tokenAddress).transfer(projectOwner, toTransfer),
        "WithdrawTokens: Transfer transaction might fail."
      );
    } else if (tokenAddress == address(0)) {
      if (amount == 0) {
        toTransfer = address(this).balance;
      }
      if (msg.sender == receiver) {
        revert ReceiverCannotInitiateTransferEther();
      }
      payable(receiver).transfer(toTransfer);
    } else {
      if (amount == 0) {
        toTransfer = IERC20(tokenAddress).balanceOf(address(this));
      }
      IERC20(tokenAddress).safeTransfer(receiver, toTransfer);
    }
}
```

Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    if (newOwner == address(0)) {
      revert OwnableInvalidOwner(address(0));
    }
    _transferOwnership(newOwner);
}
```

Contract owner can renounce ownership

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

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:	5%
Sell fees:	5%
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

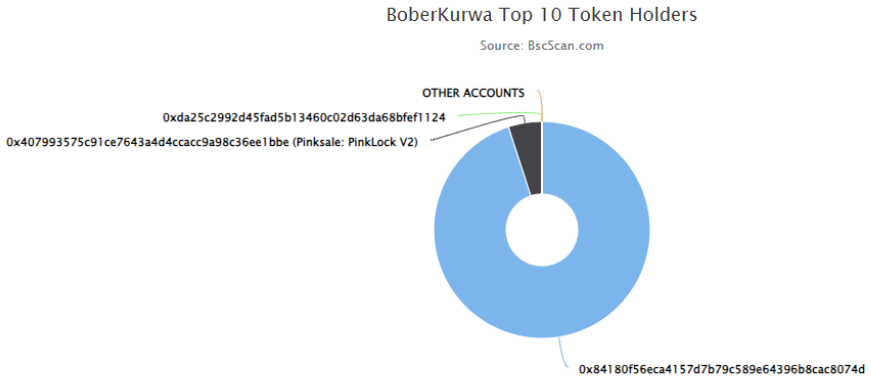
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

BOBERKURWA TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (100,000,000,000.00 Tokens) of BoberKurwa

Token Total Supply: 100,000,000,000.00 Token | Total Token Holders: 3



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

Rank	Address	Quantity (Token)	Percentage
1	0x84180f...caC8074D	94,999,999,986.287	95.0000%
2	Pinksale: PinkLock V2	5,000,000,000	5.0000%
3	0xdA25C2...bFEF1124	13.713	0.0000%

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.