freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

ZombieBUSD

$ZB

23/01/2022

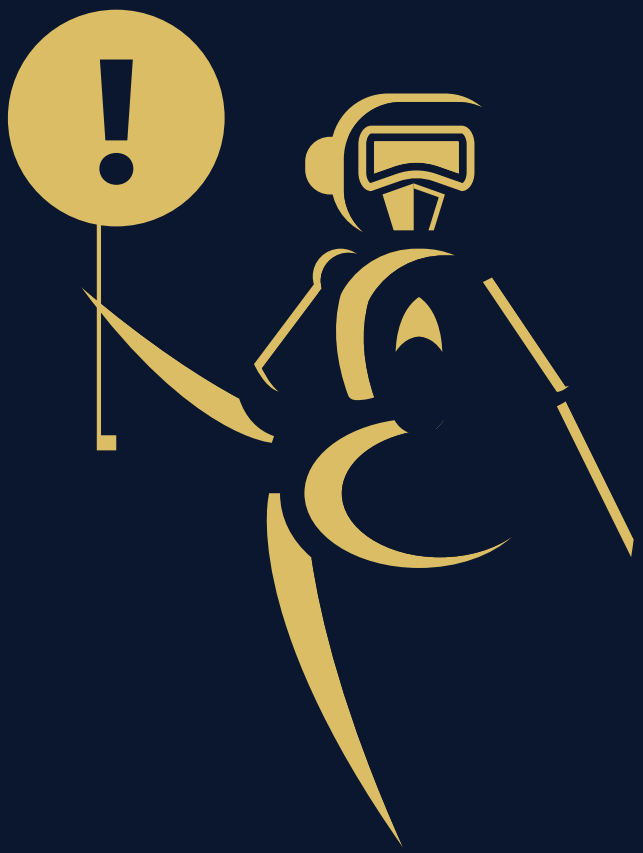
TABLE OF CONTENTS

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

0x730f0d14B19b0D83885d67bba4599cbe24678371

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 23/01/2022

WEBSITE DIAGNOSTIC

https://zombiebusd.info/

0-49 50-89 90-100

87	92	100	92	NA
Performance	Accessibility	Best Practices	SEO	Progressive Web App

Metrics

First Contentful Paint
2.1 s

Time to interactive
3.4 s

Speed Index
2.8 s

Total Blocking Time
280 ms

Large Contentful Paint
6.9 s

Cumulative Layout Shift
0

WEBSITE IMPROVEMENTS

Properly size images

Reduce unused JavaScript

Reduce unused CSS

Image elements do not have explicit width and height

Avoid enormous network payloads Total size was 3,716 KiB

Background and foreground colors do not have a sufficient contrast ratio

Heading elements are not in a sequentially-descending order

AUDIT OVERVIEW

98

Security Score

98

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(s) from fees

```
function excludeFromFees(address account, bool excluded) public onlyOwner {
        require(
            _isExcludedFromFees[account] != excluded,
            "BABYTOKEN: Account is already the value of 'excluded'"
        );
        _isExcludedFromFees[account] = excluded;

        emit ExcludeFromFees(account, excluded);
    }
    .
    .
    .
    function excludeMultipleAccountsFromFees(
        address[] calldata accounts,
        bool excluded
    ) public onlyOwner {
        for (uint256 i = 0; i < accounts.length; i++) {
            _isExcludedFromFees[accounts[i]] = excluded;
        }

        emit ExcludeMultipleAccountsFromFees(accounts, excluded);
    }
```

Contract owner can change max tx amount

```
function setSwapTokensAtAmount(uint256 amount) external onlyOwner {
        swapTokensAtAmount = amount;
    }
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
        _setOwner(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");
        _setOwner(newOwner);
    }
```

Contract owner can change the fees

```
function setTokenRewardsFee(uint256 value) external onlyOwner {
        tokenRewardsFee = value;
        totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee);
        require(totalFees <= 25, "Total fee is over 25%");
}

function setLiquiditFee(uint256 value) external onlyOwner {
        liquidityFee = value;
        totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee);
        require(totalFees <= 25, "Total fee is over 25%");
}

function setMarketingFee(uint256 value) external onlyOwner {
        marketingFee = value;
        totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee);
        require(totalFees <= 25, "Total fee is over 25%");
}
```

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 issue during the first review.

TOKEN DETAILS

Details

Buy fees:	14%
Sell fees:	14%
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

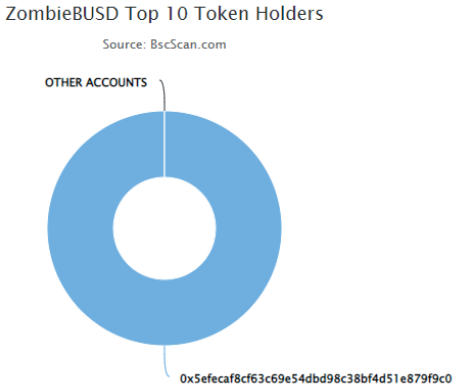
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

ZOMBIEBUSD TOKEN DISTRIBUTION & TOP 10 TOKEN HOLDERS

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

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



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

Rank	Address	Quantity (Token)	Percentage
1	0x5efecaf8cf63c69e54dbd98c38bf4d51e879f9c0	1,000,000,000,000	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.