freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

ScoobyZilla

$SBZ

08/01/2023

TOKEN OVERVIEW

Fees

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

Fees privileges

- Can change fees up to 40%

Ownership

- Owned

Minting

- No mint function detected

Max Tx Amount / Max Wallet Amount

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

Blacklist

- No blacklist function detected

Other privileges

- Can exclude / include from fees

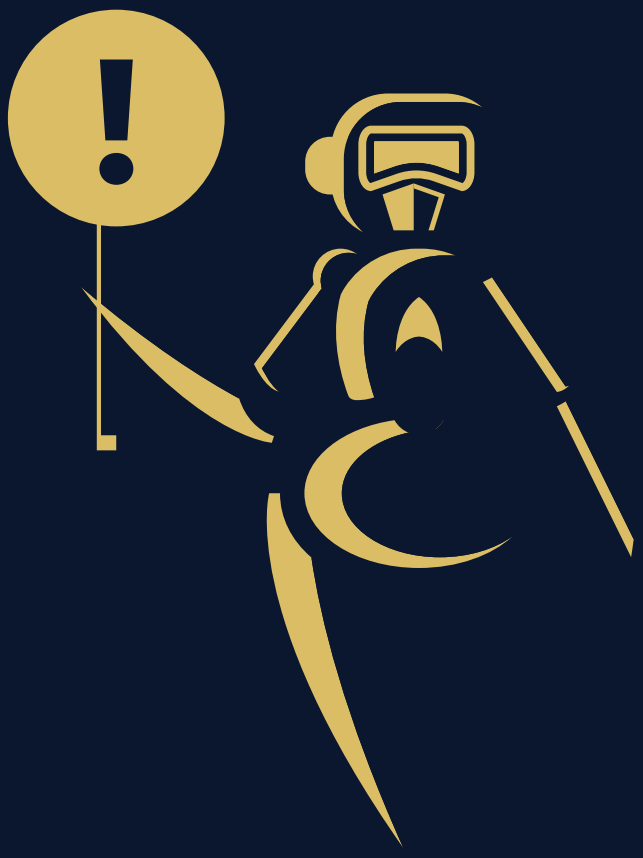
TABLE OF CONTENTS

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

0x7dA81f2f2e713EC32c9621Cb666F937C7b4B929f

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 08/01/2023

AUDIT OVERVIEW

87

Security Score

89

Static Scan
Automatic scanning for common vulnerabilities

84

ERC Scan
Automatic checks for ERC's conformance

0 High

5 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 change tx limitations and wallet limitations

```
function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner {
        require(maxTxPercent > 1, "very low maxTxPercent");
        _maxTxAmount = _tTotal.mul(maxTxPercent).div(10**2);
}

function setMaxWalletPercent(uint256 maxWallPercent) external onlyOwner {
        require(maxWallPercent > 1, "very low maxWallPercent");
        _maxWalletSize = _tTotal.mul(maxWallPercent).div(10**2);
}
```

- Contract owner can change fees up to 40%

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner {
        require(taxFee < 10, "very high taxFee");
        _taxFee = taxFee;
}

function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner {
        require(liquidityFee < 10, "very high liquidityFee");
        _liquidityFee = liquidityFee;
}

function setBurnFeePercent(uint256 burnFee) external onlyOwner {
        require(burnFee < 10, "very high burnFee");
        _burnFee = burnFee;
}
```

- Contract owner can exclude/include wallet 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(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Uniswap router.');
    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 withdraw stuck tokens from smart contract

```
function removeStuckToken(address _address) external onlyOwner {
    require(
        _address != address(this),
        "Can't withdraw tokens destined for liquidity"
    );
    require(
        IERC20(_address).balanceOf(address(this)) > 0,
        "Can't withdraw 0"
    );
    IERC20(_address).transfer(
        owner(),
        IERC20(_address).balanceOf(address(this))
    );
}

function withdrawStuckBNB() external onlyOwner {
    require(address(this).balance > 0, "Can't withdraw negative or zero");
    payable(owner()).transfer(address(this).balance);
}
```

Contract owner can change swap settings

```
function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}
```

Contract owner can exclude/include wallet from tx limitations

```
function excludeInMaxTxAmount(address account) public onlyOwner {
    _isExcludedFromMaxTxAmount[account] = true;
}

function includeInMaxTxAmount(address account) public onlyOwner {
    _isExcludedFromMaxTxAmount[account] = false;
}
```

Contract owner can exclude/include wallet from wallet limitations

```
function excludeInWalletSize(address account) public onlyOwner {
    _isExcludedFromWalletSize[account] = true;
}

function includeInWalletSize(address account) public onlyOwner {
    _isExcludedFromWalletSize[account] = false;
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    _transferOwnership(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");
    _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 issues during the first review.

TOKEN DETAILS

Details

Buy fees:	5%
Sell fees:	5%
Max TX:	30,000,000,000
Max Sell:	N/A

Honeypot Risk

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

Others

Liquidity:	N/A
Holders:	Clean

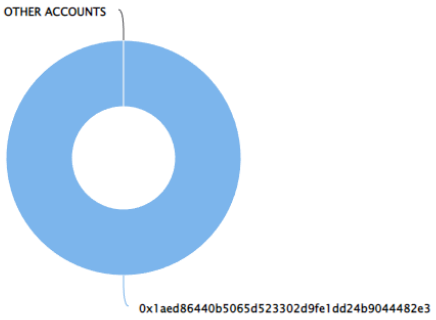
SBZ TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

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

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

ScoobyZilla Top 10 Token Holders

Source: BscScan.com



(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	0x1aed86440b5065d523302d9fe1dd24b9044482e3	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.