freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Seed

$Seed

27/04/2022

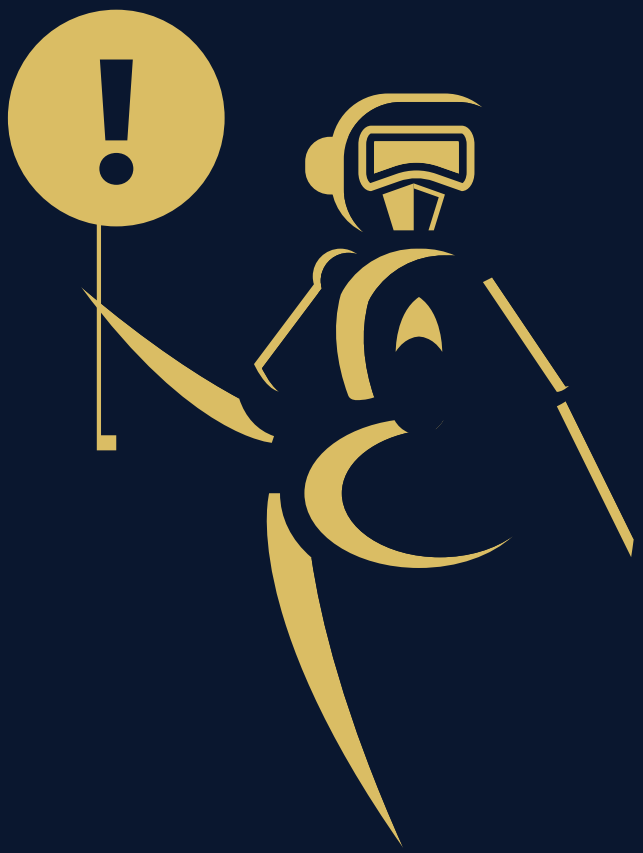
TABLE OF CONTENTS

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

0x6d69c23A98963B1EA83d620e018ca9e62A60E809

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 27/04/2022

AUDIT OVERVIEW

97

Security Score

96

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't exclude address from transactions

Contract owner can change rebase settings

```
function setAutoInfo(bool autoRebase, bool autoAddLiquidity, bool autoSwapBack, bool autoSwapUSDT)
external onlyOwner {
    if (autoRebase) {
        _lastRebasedTime = block.number;
    }
    _autoRebase = autoRebase;
    _autoAddLiquidity = autoAddLiquidity;
    _autoSwapBack = autoSwapBack;
    _autoSwapUSDT = autoSwapUSDT;
}
```

Contract owner can change autoLiquidityReceiver, treasuryReceiver, safuuInsuranceFundReceiver, firePit and pairContract addresses

```
function setFeeReceivers(
        address _autoLiquidityReceiver,
        address _treasuryReceiver,
        address _safuuInsuranceFundReceiver,
        address _firePit
    ) external onlyOwner {
        autoLiquidityReceiver = _autoLiquidityReceiver;
        treasuryReceiver = _treasuryReceiver;
        safuuInsuranceFundReceiver = _safuuInsuranceFundReceiver;
        firePit = _firePit;
}

function setLP(address _address) external onlyOwner {
        pairContract = IPancakeSwapPair(_address);
        dividendTracker.excludeFromDividends(_address);
}
```

Contract owner can exclude wallet from dividends

```
function excludeFromDividends(address account) external onlyOwner {
        require(!excludedFromDividends[account]);
        excludedFromDividends[account] = true;

        _setBalance(account, 0);
        tokenHoldersMap.remove(account);

        emit ExcludeFromDividends(account);
}
```

Contract owner can exclude/include wallet from tax

```
function setWhitelist(address _addr, bool flag) external onlyOwner {
        _isFeeExempt[_addr] = flag;
}
```

Contract owner can change fees up to 100%

```
function setFee(
        uint256 _liquidityFee,
        uint256 _treasuryFee,
        uint256 _safuuInsuranceFundFee,
        uint256 _usdtFee,
        uint256 _firePitFee
    ) external onlyOwner {
        liquidityFee = _liquidityFee;
        treasuryFee = _treasuryFee;
        safuuInsuranceFundFee = _safuuInsuranceFundFee;
        usdtFee = _usdtFee;
        firePitFee = _firePitFee;
        totalFee = liquidityFee.add(treasuryFee).add(safuuInsuranceFundFee).add(usdtFee).add(firePitFee);
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public onlyOwner {
            emit OwnershipRenounced(_owner);
            _owner = address(0);
}
```

Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public onlyOwner {
        _transferOwnership(newOwner);
}

function _transferOwnership(address newOwner) internal {
        require(newOwner != address(0));
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
}
```

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:	15%
Sell fees:	15%
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

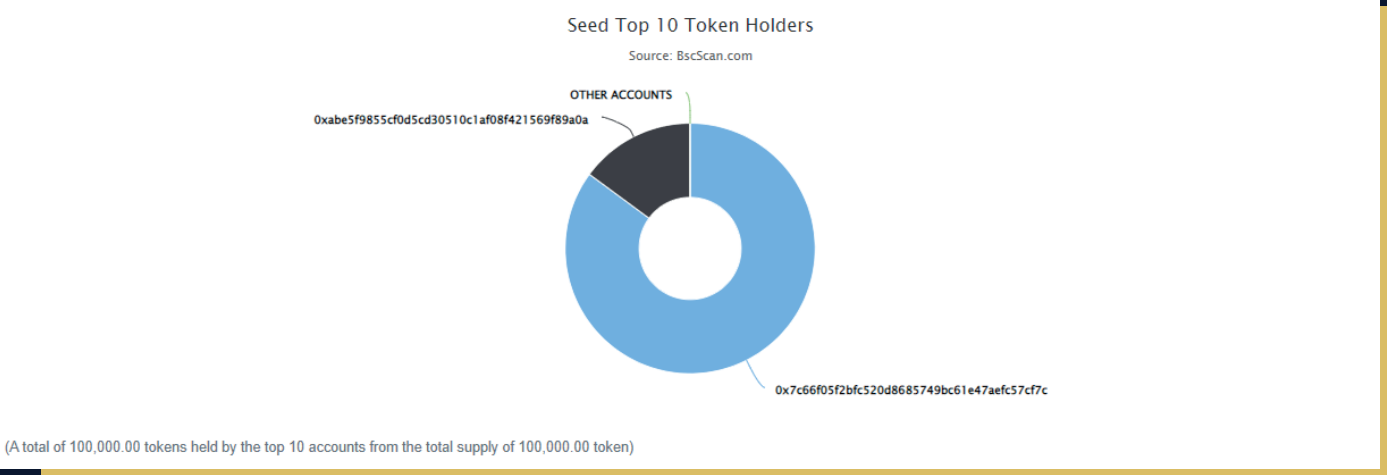
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

SEED TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

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

Token Total Supply: 100,000.00 Token | Total Token Holders: 2



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

Rank	Address	Quantity (Token)	Percentage
1	0x7c66f05f2bfc520d8685749bc61e47aefc57cf7c	85,100	85.1000%
2	0xabe5f9855cf0d5cd30510c1af08f421569f89a0a	14,900	14.9000%

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.