freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Colonizer

$COL

20/08/2023

TOKEN OVERVIEW

Fees

- Buy fees: 0%
- Sell fees: 0%

Fees privileges

- Can change fees up to 20%

Ownership

- Owned

Minting

- No mint function

Max Tx Amount / Max Wallet Amount

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

Blacklist

- Blacklist function not detected

Other privileges

- Can exclude / include from fees

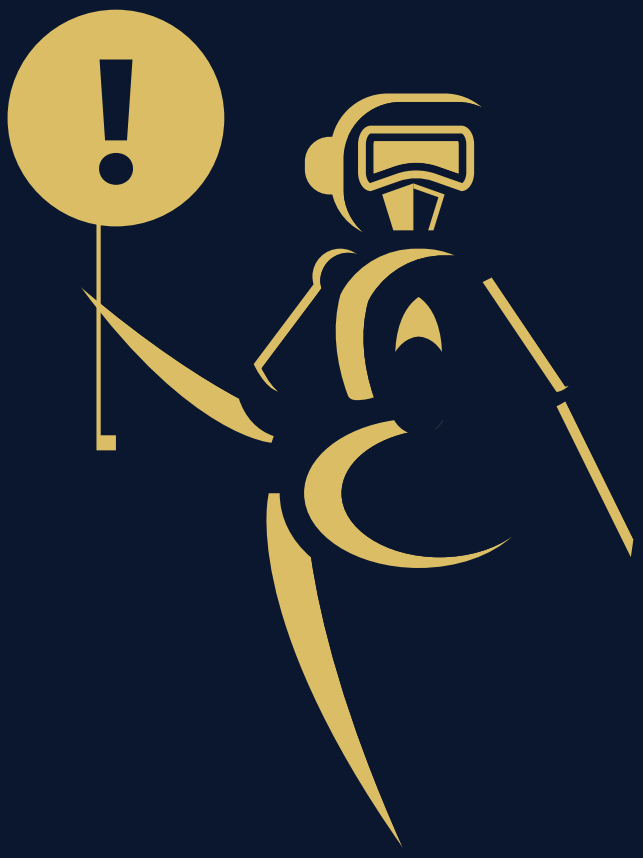
TABLE OF CONTENTS

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

0x0B0f3A9A9571d8F6EbB480c6EA3a5CF585d9FdB1

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 20/08/2023

WEBSITE DIAGNOSTIC

https://colonizer.pro/

0-49 50-89 90-100

Performance Accessibility Best Practices SEO Progressive Web App

Socials

Twitter

https://twitter.com/colonizer_pro

Telegram

https://t.me/colonizer_pro

AUDIT OVERVIEW

95

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

91

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 an address from transactions
- Contract owner can exclude/include wallet(s) from tax

```
function excludeMultipleAccountsFromFees(
      address[] calldata accounts,
      bool excluded
  ) external onlyOwner {
      for (uint256 i = 0; i < accounts.length; i++) {
          _isExcludedFromFee[accounts[i]] = excluded;
      }

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

- Contract owner can change swap settings

```
function toggleSwap(bool enable) external onlyOwner {
      swapEnabled = enable;

      emit ChangedSwapEnable(enable);
}
```

- Contract owner can change treasuryAddress address

Current value:

treasuryAddress: 0x93cA3072DD0B4E5e55221198Fe9Be90976a99285

```
function setTreasuryAddress(address payable account) external onlyOwner {
      require(account != address(0x0), "treasury address cannot be zero");

      treasuryAddress = account;
      _isExcludedFromFee[account] = true;

      emit UpdatedTreasuryWallet(account);
}
```

Contract owner can change fees up to 20%

```
function setFee(
        uint16 reflectionTax_,
        uint16 treasuryTax_
    ) public onlyOwner {
        validateFees(reflectionTax_, treasuryTax_);

        reflectionTax = reflectionTax_;
        treasuryTax = treasuryTax_;

        emit ChangedFees(reflectionTax_, treasuryTax_);
}

function validateFees(
        uint16 reflectionTax_,
        uint16 treasuryTax_
    ) internal pure {
        require(
            reflectionTax_ + treasuryTax_ <= 20,
            "Fees cannot be greater than 20%"
        );
}
```

Contract owner has ability to retrieve token held by the contract

```
function withdrawETH() external onlyOwner {
        treasuryAddress.transfer(address(this).balance);
}

function withdrawTokens(
        address token,
        address to,
        uint amount
    ) external onlyOwner {
        require(IERC20(token).transfer(to, amount), "transfer rejected");

        emit WithdrawedTokens(token, to, amount);
}
```

Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public virtual onlyOwner {
        require(
            newOwner != address(0x0),
            "call the renounceOwnership for zero address"
        );

        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    emit OwnershipTransferred(_owner, address(0));
    _owner = 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:	0%
Sell fees:	0%
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

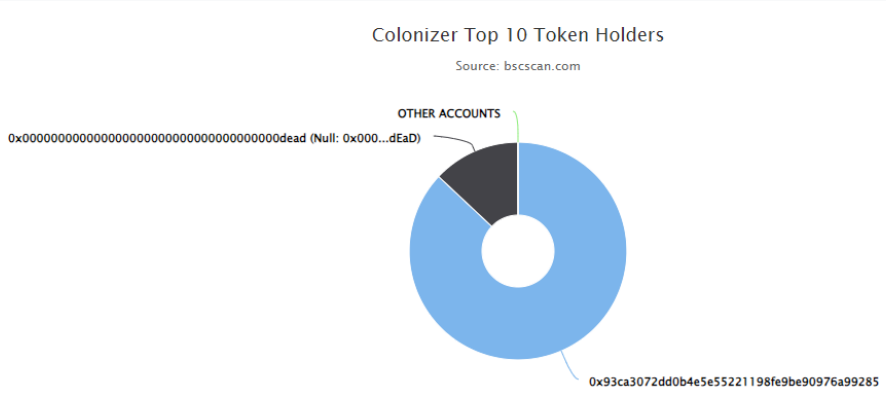
Rug Pull Risk

Liquidity:	N/A
Holders:	87% unlocked tokens

COL TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

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

Token Total Supply: 10,000,000,000.00 Token | Total Token Holders: 2



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

Rank	Address	Quantity (Token)	Percentage
1	0x93cA30...76a99285	8,700,000,000	87.0000%
2	Null: 0x000...dEaD	1,300,000,000	13.0000%

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.