freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

EGG

$EGG

17/05/2022

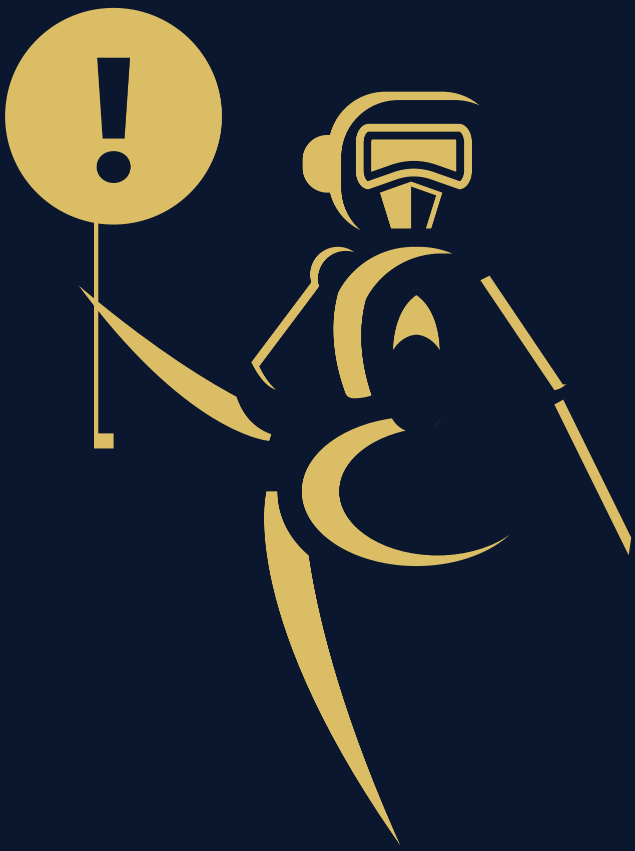
TABLE OF CONTENTS

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

0xc3b2Ae4337254a10afaba93Ff191fcA749cdA1bB

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 17/05/2022

AUDIT OVERVIEW

95

Security Score

94

Static Scan
Automatic scanning for common vulnerabilities

97

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 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) external onlyOwner {
        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 change swap settings

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

Contract owner can change buy fees up to 20%

```
function UpdateBuyTaxFeePercentage(
      uint256 taxFee,
      uint256 liquidityFee,
      uint256 MarketingFee,
      uint256 DevelopmentFee
   ) external onlyOwner {
      _taxFee = taxFee;
      _previousTaxFee = _taxFee;
      _liquidityFee = liquidityFee;
      _previousLiquidityFee = _liquidityFee;
      _MarketingFee = MarketingFee;
      _previousMarketingFee = _MarketingFee;
      _DevelopmentFee = DevelopmentFee;
      _previousDevelopmentFee = _DevelopmentFee;

      _buyTaxFee = _taxFee;
      _buyLiquidityFee = _liquidityFee;
      _buyMarketingFee = _MarketingFee;
      _buyDevelopmentFee = DevelopmentFee;

      require(
         taxFee.add(liquidityFee).add(MarketingFee).add(
            DevelopmentFee
         ) <= 20,
         "You can't set more than 20%"
      );
      emit BuyTaxFeeUpdated(
         taxFee.add(liquidityFee).add(MarketingFee).add(
            DevelopmentFee
         )
      );
}
```

Contract owner can change sell fees up to 20%

```
function UpdateSellTaxFeePercentage(
      uint256 taxFee,
      uint256 liquidityFee,
      uint256 MarketingFee,
      uint256 DevelopmentFee
   ) external onlyOwner {
      _sellTaxFee = taxFee;
      _sellLiquidityFee = liquidityFee;
      _sellMarketingFee = MarketingFee;
      _sellDevelopmentFee = DevelopmentFee;
      require(
         taxFee.add(liquidityFee).add(MarketingFee).add(
            DevelopmentFee
         ) <= 20,
         "You can't set more than 20%"
      );
      emit SellTaxFeeUpdated(
         taxFee.add(liquidityFee).add(MarketingFee).add(
            DevelopmentFee
         )
      );
}
```

Contract owner can set max tx percentage (with threshold)

```
function UpdateMaxTxPercentage(uint256 maxTxPercentage) external onlyOwner {
    require(
      maxTxPercentage >= 1,
      "Percentage should be greater or equal to 1%"
    );
    _maxTxAmount = _tTotal.mul(maxTxPercentage).div(10**2);
    emit MaxTxAmountUpdated(_maxTxAmount);
}
```

Contract owner can set wallet limitation (with threshold)

```
function UpdateMaxWalletHoldingPercentage(uint256 maxWalletPercentage)
    external
    onlyOwner
  {
    require(
      maxWalletPercentage >= 1,
      "Percentage should be greater or equal to 1%"
    );
    _maxWalletHoldingLimit = _tTotal.mul(maxWalletPercentage).div(10**2);
```

Contract owner can exclude/include address from wallet limitations

```
function setExcludedFromWalletHoldingLimitStatus(address account, bool status) external onlyOwner(){
    _isExcludedFromWalletHoldingLimit[account] = status;
}
```

Contract owner can change MarketingWallet and DevelopmentWallet addresses

Current values:

MarketingWallet : 0xc9ae92454622f1f800c9896a4df4c3d8b9cc5709

DevelopmentWallet : 0x777a90525a500cded3f87cf7fec390a5df35d22c

```
function UpdateWallets(
    address payable newMarkeitngWallet,
    address newDevelopmentWallet
  ) external onlyOwner {
    require(
      newMarkeitngWallet != address(0) &&
        newDevelopmentWallet != address(0),
      "You can't set Zero Address"
    );
    MarketingWallet = newMarkeitngWallet;
    DevelopmentWallet = newDevelopmentWallet;
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    emit OwnershipTransferred(_owner, address(0));
    _owner = 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"
    );
    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:	6%
Sell fees:	7%
Max TX:	10,000,000,000,000
Max Sell:	N/A

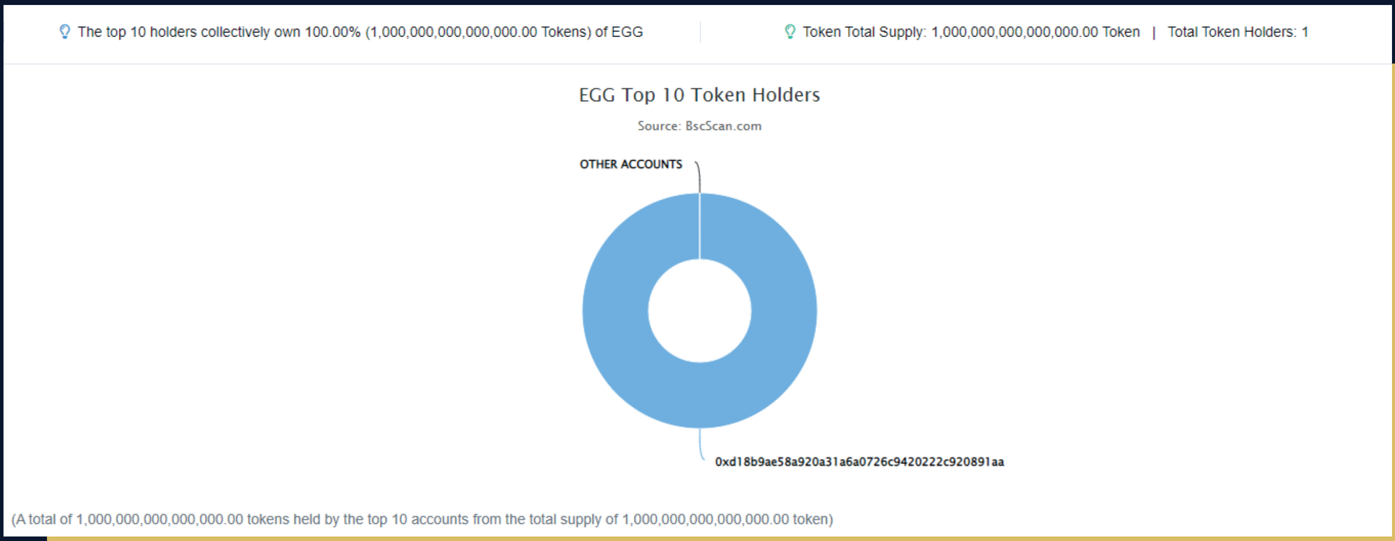
Honeypot Risk

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

EGG TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

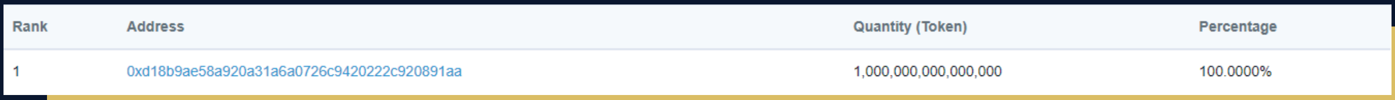

The top 10 holders collectively own 100.00% (1,000,000,000,000,000.00 Tokens) of EGG

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

EGG Top 10 Token Holders

Source: BscScan.com

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



Rank	Address	Quantity (Token)	Percentage
1	0xd18b9ae58a920a31a6a0726c9420222c920891aa	1,000,000,000,000,000	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.