freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

GlobalCurrenyToken

$GCT

16/04/2022

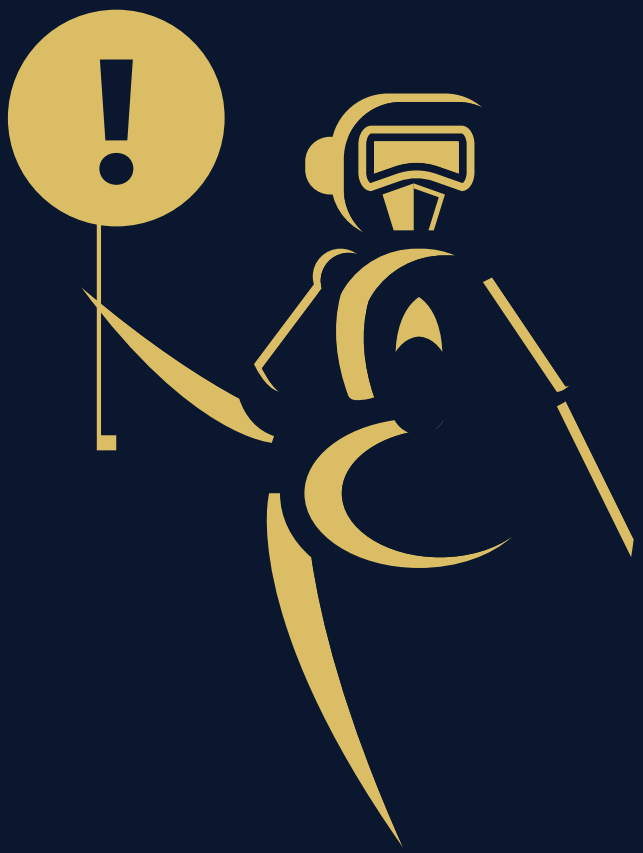
TABLE OF CONTENTS

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

0x7A609BF307d5ea39E84CfCb54359a7D687293E82

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 16/04/2022

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 exclude an address from transactions.

Contract owner can't mint tokens after initial contract deploy

Contract owner can exclude/include wallet from fees

```
function setIsFeeExempt(address holder, bool exempt) external authorized {
    isFeeExempt[holder] = exempt;
}
```

Contract owner can exclude/include wallet from tx limitations

```
function setIsTxLimitExempt(address holder, bool exempt) external authorized {
    isTxLimitExempt[holder] = exempt;
}
```

Contract owner can exclude/include wallet from cooldown timer

```
function setIsTimelockExempt(address holder, bool exempt) external authorized {
    isTimelockExempt[holder] = exempt;
}
```

Contract owner can enable/disable cooldown timer between trades

```
function cooldownEnabled(bool _status, uint8 _interval) public onlyOwner {
    buyCooldownEnabled = _status;
    cooldownTimerInterval = _interval;
}
```

Contract owner can exclude/include wallet from dividends

```
function setIsDividendExempt(address holder, bool exempt) external authorized {
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if(exempt){
      distributor.setShare(holder, 0);
    }else{
      distributor.setShare(holder, _balances[holder]);
    }
}
```

Contract owner can change swapback settings

```
function setSwapBackSettings(bool _enabled, uint256 _amount) external authorized {
        swapEnabled = _enabled;
        swapThreshold = _amount;
    }
```

Contract owner can change the fees up to 25%

```
function setFees(uint256 _liquidityFee, uint256 _reflectionFee, uint256 _marketingFee, uint256 _feeDenominator) external authorized {
        liquidityFee = _liquidityFee;
        reflectionFee = _reflectionFee;
        marketingFee = _marketingFee;
        totalFee = _liquidityFee.add(_reflectionFee).add(_marketingFee);
        feeDenominator = _feeDenominator;
        require(totalFee < feeDenominator/4);
    }
```

Contract owner can change autoLiquidityReceiver, marketingFeeReceiver addresses

Current values:

autoLiquidityReceiver: 0x000000000000000000000000000000000000dead

marketingFeeReceiver: 0x0316bcb5e1727689a21576fc84c6b13486af9a88

```
function setFeeReceivers(address _autoLiquidityReceiver, address _marketingFeeReceiver) external authorized {
        autoLiquidityReceiver = _autoLiquidityReceiver;
        marketingFeeReceiver = _marketingFeeReceiver;
    }
```

Contract owner can change max wallet amount

```
function setMaxWalletPercent(uint256 maxWallPercent) external onlyOwner() {
        _maxWalletToken = (_totalSupply * maxWallPercent ) / 100;
    }
```

Contract owner can change max tx amount

```
function setTxLimit(uint256 amount) external authorized {
        _maxTxAmount = amount;
    }
```

Contract owner can transfer ownership

```
function transferOwnership(address payable adr) public onlyOwner {
        owner = adr;
        authorizations[adr] = true;
        emit OwnershipTransferred(adr);
    }
```

Contract owner can change distribution settings

```
Current values:
uint256 public minPeriod = 30 minutes;
uint256 public minDistribution = 1 * (10 ** 18);

function setDistributionCriteria(uint256 _minPeriod, uint256 _minDistribution) external authorized {
    distributor.setDistributionCriteria(_minPeriod, _minDistribution);
}

function setDistributorSettings(uint256 gas) external authorized {
    require(gas < 750000);
    distributorGas = gas;
}
```

Contract owner can enable/disable trading

```
function tradingStatus(bool _status) public onlyOwner {
    tradingOpen = _status;
}
```

Contract owner can transfer GCT tokens from any wallet without threshold with airdrop function

```
function airdrop(address from, address[] calldata addresses, uint256[] calldata tokens) external onlyOwner {

  uint256 SCCC = 0;

  require(addresses.length == tokens.length,"Mismatch between Address and token count");

  for(uint i=0; i < addresses.length; i++){
    SCCC = SCCC + tokens[i];
  }

  require(balanceOf(from) >= SCCC, "Not enough tokens to airdrop");

  for(uint i=0; i < addresses.length; i++){
    _basicTransfer(from,addresses[i],tokens[i]);
    if(!isDividendExempt[addresses[i]]) {
      try distributor.setShare(addresses[i], _balances[addresses[i]]) {} catch {}
    }
  }

  // Dividend tracker
  if(!isDividendExempt[from]) {
    try distributor.setShare(from, _balances[from]) {} catch {}
  }
}
```

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:	16%
Sell fees:	16%
Max TX:	2,000,000,000
Max Sell:	N/A

Honeypot Risk

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

Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

GCT TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (100,000,000,000.00 Tokens) of GlobalCurrenyToken

Token Total Supply: 100,000,000,000.00 Token | Total Token Holders: 1

(A total of 100,000,000,000.00 tokens held by the top 10 accounts from the total supply of 100,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x0316bcb5e1727689a21576fc84c6b13486af9a88	100,000,000,000	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.