freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Cyberquad

$Cyberquad

02/08/2023

TOKEN OVERVIEW

Fees

- Buy fees: 1%
- Sell fees: 3%

Fees privileges

- Can change fees up to 100%

Ownership

- Owned

Minting

- No mint function

Max Tx Amount / Max Wallet Amount

- Can change max tx amount (without threshold)

Blacklist

- Blacklist function not detected

Other privileges

- Can exclude / include from fees
- Contract owner has to call confirmLpFilled() function to enable trade

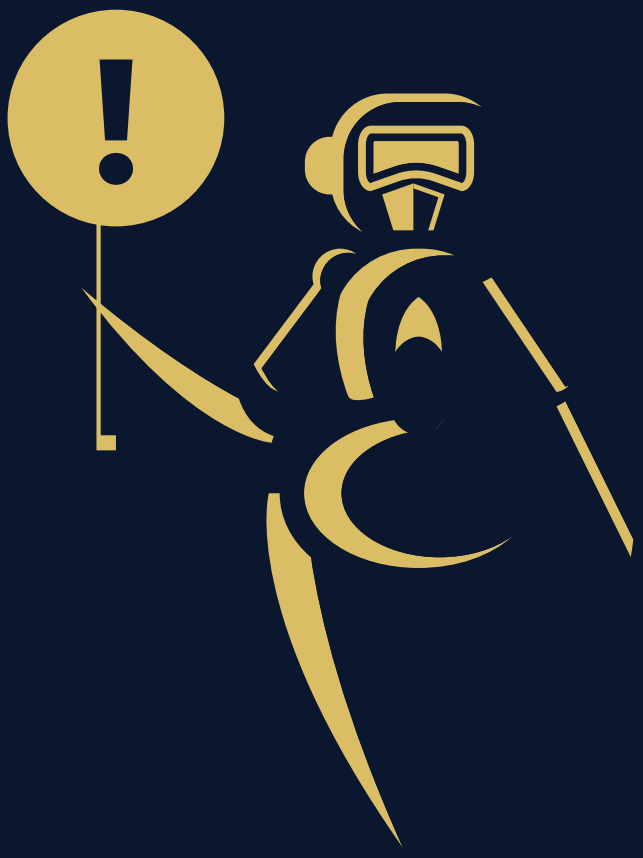
TABLE OF CONTENTS

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

0x0f16b22deCbCee96147aB5E67B9DFac8a4941ae2

Network: Ethereum (ETH)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 02/08/2023

WEBSITE DIAGNOSTIC

https://www.cyberquad.charity/

 0-49

 50-89

 90-100

Performance

Accessibility

Best Practices

SEO

Progressive Web App

Socials

Twitter

https://twitter.com/CyberquadC66512

Telegram

https://t.me/CyberquadEN

AUDIT OVERVIEW

71

Security Score
HIGH RISK
Audit FAIL

93

Static Scan
Automatic scanning for common vulnerabilities

86

ERC Scan
Automatic checks for ERC's conformance

3	High
2	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 wallet from tax

```
function updateExcludedFromFees(address _address, bool state) external onlyOwner {
    excludedFromFees[_address] = state;
}
```

- Contract owner can change swap settings (without threshold)

```
function setSwapEnabled(bool state) external onlyOwner {
    swapEnabled = state;
}

function setSwapThreshold(uint256 new_amount) external onlyOwner {
    swapThreshold = new_amount * 10**decimals();
}
```

- Contract owner has to call confirmLpFilled function to enable trade

Note that any wallet excluded from fees can trade even if trading is disabled

```
function confirmLpFilled() external onlyOwner{
    lpFilled = true;
    swapEnabled = true;
}
.
.
.
require(lpFilled || excludedFromFees[sender] || excludedFromFees[recipient], "Trading disabled");
.
.
.
```

- Contract owner can change max tx amount (without threshold)

Note that setting the value too low may prevent users from making purchase transactions

```
function setMaxBuy(uint256 amount) external onlyOwner{
    maxBuy = amount * 10**decimals();
}
```

Contract owner can change fees up to 100%

```
function setBuyTaxes(uint256 _marketing, uint256 _dev, uint256 _liquidity) external onlyOwner{
        buyTaxes = Taxes(_marketing, _dev, _liquidity);
        totalBuyTax = _marketing + _dev + _liquidity;
}

function setSellTaxes(uint256 _marketing, uint256 _dev, uint256 _liquidity) external onlyOwner{
        sellTaxes = Taxes(_marketing, _dev, _liquidity);
        totalSellTax = _marketing + _dev + _liquidity;
}
```

Contract owner can change marketingWallet and charityFunds addresses

Current values:

marketingWallet : 0x228Fe119Adf2d141D2020C44eB66F2641E7833a5

charityFunds : 0x818A00Aa2D2F008e3730e70B27BB672bA172a789

```
function updateCharityFunds(address newWallet) external onlyOwner{
        charityFunds = newWallet;
}

function updateMarketingWallet(address newWallet) external onlyOwner{
        marketingWallet = newWallet;
}
```

Contract owner has ability to retrieve any token held by the contract

Native tokens excluded

```
function rescueBEP20(address tokenAddress, uint256 amount) external onlyOwner{
        require(tokenAddress != address(this), "Can't take self token");
        IERC20(tokenAddress).transfer(owner(), amount);
}

function rescueBNB(uint256 weiAmount) external onlyOwner{
        payable(owner()).sendValue(weiAmount);
}
```

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);
}
```

- **The liquidity of the contract automatically gets credited into the owner's wallet whenever the 'addLiquidity' function is called inside the contract.**
  Note that it cannot be called manually but it will be done automatically every time the swap and liquify function is called. Moreover, even after the renouncement of the ownership, this liquidity will still be credited to the owner's wallet.

```
function addLiquidity(uint256 tokenAmount, uint256 bnbAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(router), tokenAmount);

    // add the liquidity
    router.addLiquidityETH{value: bnbAmount}(
      address(this),
      tokenAmount,
      0, // slippage is unavoidable
      0, // slippage is unavoidable
      owner(),
      block.timestamp
    );
}
```

- **Contract owner can renounce ownership**

```
function renounceOwnership() public virtual onlyOwner {
    _transferOwnership(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 3 HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	1%
Sell fees:	3%
Max TX:	0
Max Sell:	N/A

Honeypot Risk

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

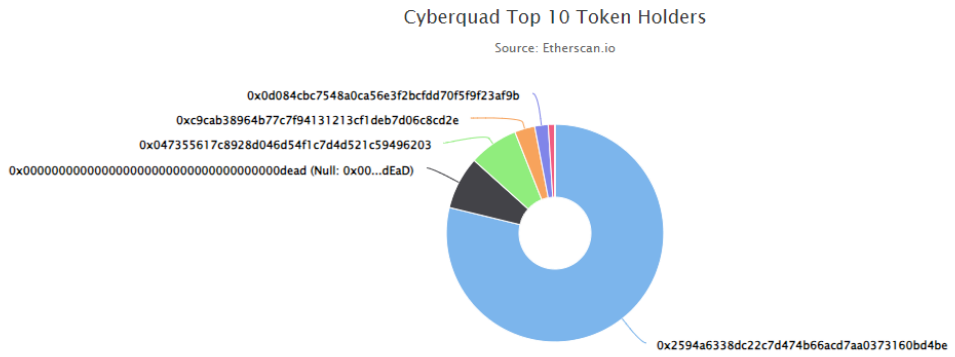
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

CYBERQUAD TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (6,100,000,000,000.00 Tokens) of Cyberquad

Token Total Supply: 6,100,000,000,000.00 Token | Total Token Holders: 6



(A total of 6,100,000,000,000.00 tokens held by the top 10 accounts from the total supply of 6,100,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x2594A6...160bD4BE	4,806,312,000,000	78.7920%
2	Null: 0x00...dEaD	479,338,000,000	7.8580%
3	0x047355...59496203	448,350,000,000	7.3500%
4	0xc9cAb3...06C8CD2e	183,000,000,000	3.0000%
5	0x0D084C...9f23aF9b	122,000,000,000	2.0000%
6	0x161823...ed64Eae0	61,000,000,000	1.0000%

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.