freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Candylad Coin

$Candylad Coin

09/04/2022

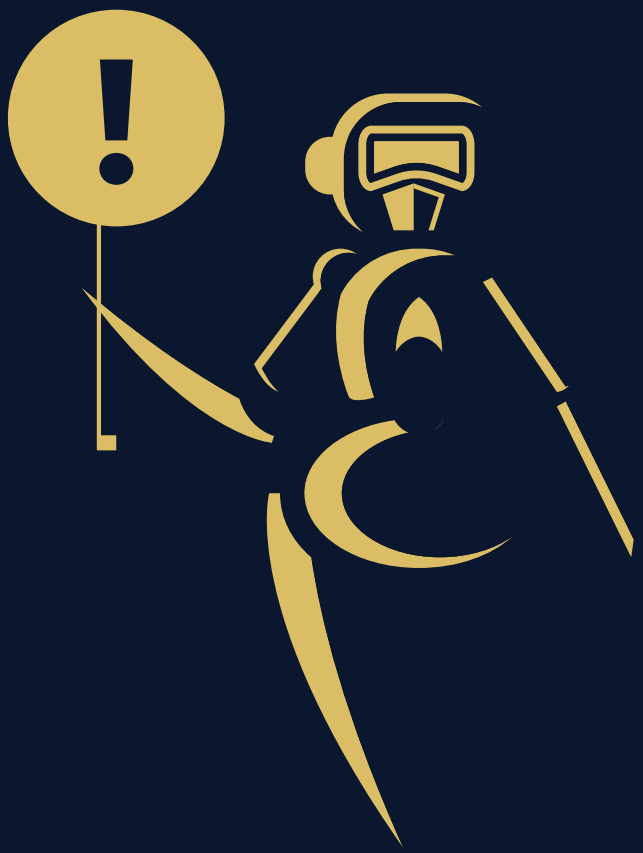
TABLE OF CONTENTS

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

0x31Fe40a8b96Fb152d35243C3B81120102B5Fa2B1

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 09/04/2022

AUDIT OVERVIEW

96

Security Score

95

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 an address from transactions

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 exclude/include wallet from tax

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

Contract owner can change fees up to 25%

```
function _setFees(
        uint256 _liquidityFee,
        uint256 _buybackFee,
        uint256 _reflectionFee,
        uint256 _marketingFee,
        uint256 _feeDenominator
    ) internal {
        liquidityFee = _liquidityFee;
        buybackFee = _buybackFee;
        reflectionFee = _reflectionFee;
        marketingFee = _marketingFee;
        totalFee = _liquidityFee.add(_buybackFee).add(_reflectionFee).add(
            _marketingFee
        );
        feeDenominator = _feeDenominator;
        require(
            totalFee < feeDenominator / 4,
            "Total fee should not be greater than 1/4 of fee denominator"
        );
}
```

Contract owner can change autoLiquidityReceiver and marketingFeeReceiver addresses

Current values:

autoLiquidityReceiver : 0x306f71e6cbcecff9614c6a5d65178a3aba0277fd

marketingFeeReceiver : 0x306f71e6cbcecff9614c6a5d65178a3aba0277fd

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

Contract owner can change buyback settings

```
function setAutoBuybackSettings(
        bool _enabled,
        uint256 _cap,
        uint256 _amount,
        uint256 _period
    ) external authorized {
        autoBuybackEnabled = _enabled;
        autoBuybackCap = _cap;
        autoBuybackAccumulator = 0;
        autoBuybackAmount = _amount;
        autoBuybackBlockPeriod = _period;
        autoBuybackBlockLast = block.number;
}

function setBuybackMultiplierSettings(
        uint256 numerator,
        uint256 denominator,
        uint256 length
    ) external authorized {
        require(numerator / denominator <= 2 && numerator > denominator);
        buybackMultiplierNumerator = numerator;
        buybackMultiplierDenominator = denominator;
        buybackMultiplierLength = length;
}

function setBuyBacker(address acc, bool add) external authorized {
        buyBacker[acc] = add;
}
```

Contract owner can transfer ownership

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

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:	12%
Sell fees:	12%
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

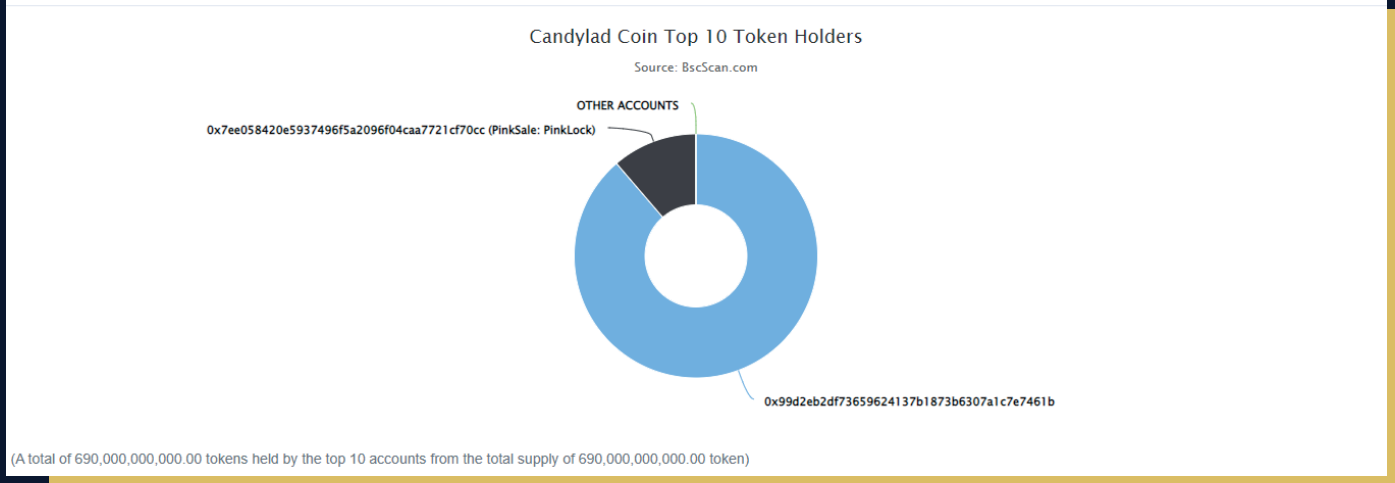
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

CANDYLAD TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (690,000,000,000.00 Tokens) of Candylad Coin

Token Total Supply: 690,000,000,000.00 Token | Total Token Holders: 2



(A total of 690,000,000,000.00 tokens held by the top 10 accounts from the total supply of 690,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x99d2eb2df73659624137b1873b6307a1c7e7461b	612,000,000,000	88.6957%
2	PinkSale: PinkLock	78,000,000,000	11.3043%

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.