freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

MERGEX ETH

$MERGEX

03/06/2023

TOKEN OVERVIEW

Fees

- Buy fees: 10%
- Sell fees: 10%

Fees privileges

- Can change fees up to 25%

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

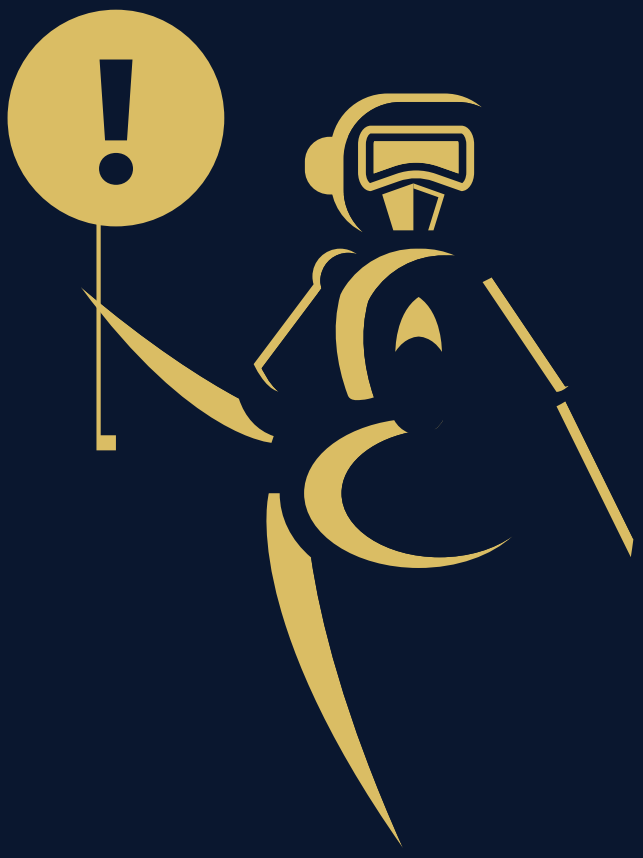
TABLE OF CONTENTS

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

0x450EbE72ba34651b771F36C3C439FE2f4B534650

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 03/06/2023

WEBSITE DIAGNOSTIC

https://mergeeth.io/


0-49


50-89


90-100

Performance

Accessibility

Best Practices

SEO

Progressive Web App

Socials


Twitter

https://twitter.com/mergex_official/


Telegram

https://t.me/mergexofficial

AUDIT OVERVIEW

91

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

88

ERC Scan
Automatic checks for ERC's conformance

0 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 setIsFeeExempt(address holder) external authorized {
        isFeeExempt[holder] = true;
}
```

- 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 marketingFeeReceiver address

Current value:

marketingFeeReceiver : 0x90f6d62730556d3524c397adb81e779911f9d58b

```
function setFeeReceivers(address _marketingFeeReceiver)
        external
        authorized
    {
        require(
            _marketingFeeReceiver != marketingFeeReceiver,
            "Marketing wallet is already that address"
        );
        require(
            !_marketingFeeReceiver.isContract(),
            "Marketing wallet cannot be a contract"
        );
        marketingFeeReceiver = _marketingFeeReceiver;
}
```

Contract owner can change fees up to 25%

```
function setFees(
        uint256 _liquidityFee,
        uint256 _buybackFee,
        uint256 _reflectionFee,
        uint256 _marketingFee,
        uint256 _feeDenominator
    ) public authorized {
        _setFees(
            _liquidityFee,
            _buybackFee,
            _reflectionFee,
            _marketingFee,
            _feeDenominator
        );
}

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 buy back settings

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

function setAutoBuybackSettings( bool _enabled, uint256 _cap, uint256 _amount, uint256 _period) external
authorized {
        require(_period > 0, "Period must be greater than 0");
        autoBuybackEnabled = _enabled;
        autoBuybackCap = _cap;
        autoBuybackAccumulator = 0;
        autoBuybackAmount = _amount;
        autoBuybackBlockPeriod = _period;
        autoBuybackBlockLast = block.number;
}
```

```
function setBuybackMultiplierSettings(
        uint256 numerator,
        uint256 denominator,
        uint256 length
    ) external authorized {
        require(length <= 2 hours, "Length must be less than 2 hours");
        require(numerator / denominator <= 2 && numerator > denominator);
        buybackMultiplierNumerator = numerator;
        buybackMultiplierDenominator = denominator;
        buybackMultiplierLength = length;
}
```

Contract owner can transfer ownership

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

Contract owner can change swap settings

```
function setSwapBackSettings(bool _enabled, uint256 _amount)
        external
        authorized
    {
        require(
            _enabled && _amount >= _totalSupply / 100_000,
            "Swapback amount should be at least 0.001% of total supply"
        );
        swapEnabled = _enabled;
        swapThreshold = _amount;
}
```

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:	10%
Sell fees:	10%
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:	100% unlocked tokens

MERGEX TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

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

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	0x90f6d62730556d3524c397adb81e779911f9d58b	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.