AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

MyCarrot

DATED : 04 Dec 23'

AUDIT SUMMARY

Project name – MyCarrot

Date: 04 Dec, 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: Passed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	1	1	0
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0x1e49862acd46c38d376d9da02974b4183e4ccfdc#code

Token Information

Token Address: --

Symbol: --

Decimals: 18

Network: --

Token Type: --

Owner: --

Deployer: --

Token Supply: 100000000000000000000000000000

Checksum: 9bff351e1897814d20411ecffceb9024

Testnet:
https://testnet.bscscan.com/address/0x1e49862acd46c38d376d9da02974b4183e4ccfdc#code

TOKEN OVERVIEW

Buy Fee: 0-20%

Sell Fee: 0-20%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: Yes

Blacklist: No

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

- [x] Return values of low-level calls
- [x] Private modifier
- [x] Multiple Sends
- [x] Using Suicide
- [x] Gas Limitand Loops
- [x] Address hardcoded
- [x] Exception Disorder
- [x] Using inline assembly
- [x] Divide before multiply
- [x] Missing Zero Address Validation
- [x] Compiler version not fixed
- [x] **Gasless Send**
- [x] Using block.timestamp
- [x] Re-entrancy
- [x] Tautology or contradiction
- [x] Timestamp Dependence
- [x] Revert/require functions
- [x] Use of tx.origin
- [x] Integer overflow/underflow
- [x] Dangerous strict equalities
- [x] Using SHA3
- [x] Using throw

CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	0
Medium-Risk	1
Low-Risk	1
Gas Optimization / Suggestions	0

POINTS TO NOTE

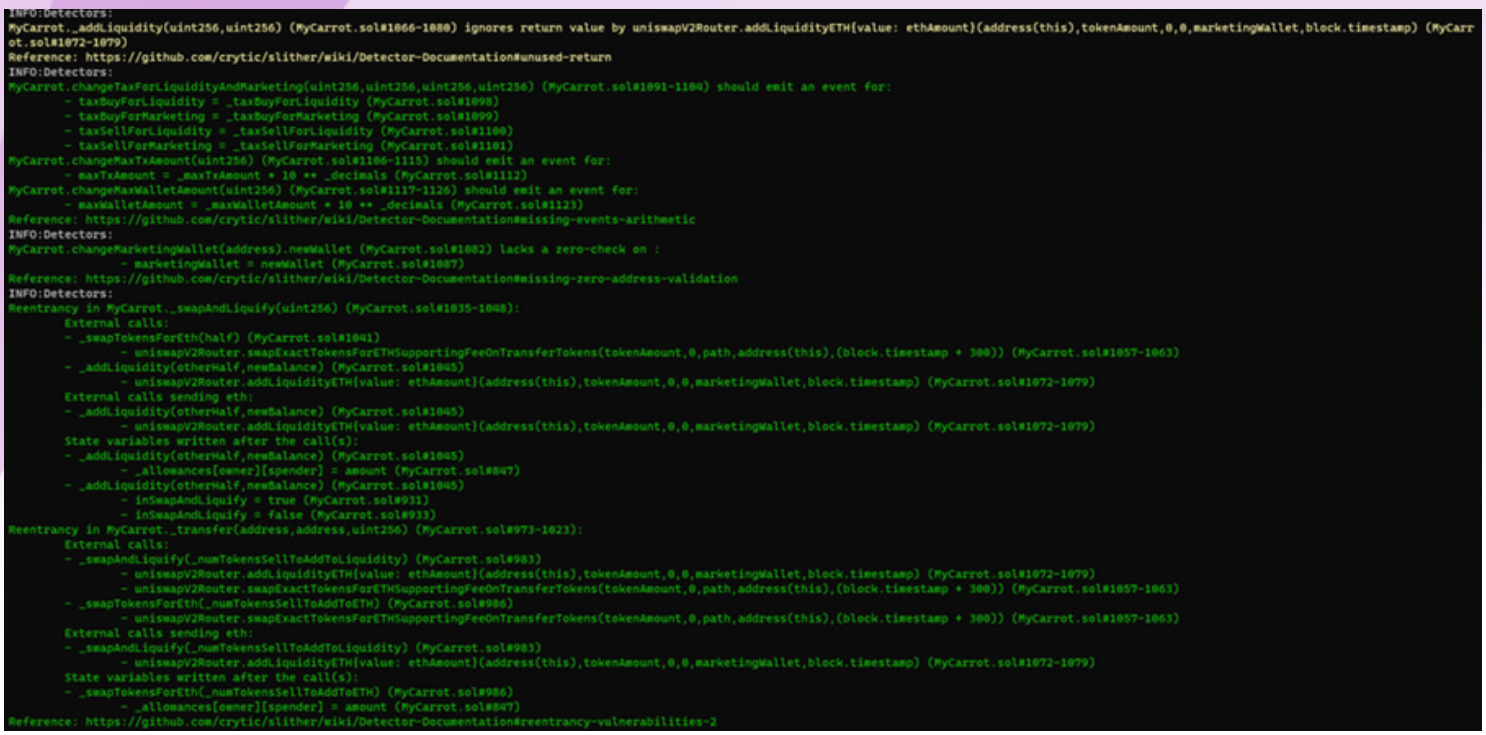
- The owner can renounce the ownership.
- The owner can transfer ownership.
- The owner can change the marketing wallet address.
- The owner can set buy and sell fees of not more than 20%.
- The owner can Include/exclude the address from fees.

STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in MyCarrot._swapAndLiquify(uint256) (MyCarrot.sol#1035-1048):
        External calls:
        - _swapTokensForEth(half) (MyCarrot.sol#1041)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),(block.timestamp + 300)) (MyCarrot.sol#1057-1063)
        - _addLiquidity(otherHalf,newBalance) (MyCarrot.sol#1045)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
        External calls sending eth:
        - _addLiquidity(otherHalf,newBalance) (MyCarrot.sol#1045)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (MyCarrot.sol#848)
                - _addLiquidity(otherHalf,newBalance) (MyCarrot.sol#1045)
        - SwapAndLiquify(half,newBalance,otherHalf) (MyCarrot.sol#1047)
Reentrancy in MyCarrot._transfer(address,address,uint256) (MyCarrot.sol#973-1023):
        External calls:
        - _swapAndLiquify(_numTokensSellToAddToLiquidity) (MyCarrot.sol#983)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),(block.timestamp + 300)) (MyCarrot.sol#1057-1063)
        - _swapTokensForEth(_numTokensSellToAddToETH) (MyCarrot.sol#986)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),(block.timestamp + 300)) (MyCarrot.sol#1057-1063)
        External calls sending eth:
        - _swapAndLiquify(_numTokensSellToAddToLiquidity) (MyCarrot.sol#983)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (MyCarrot.sol#848)
                - _swapTokensForEth(_numTokensSellToAddToETH) (MyCarrot.sol#986)
Reentrancy in MyCarrot._transfer(address,address,uint256) (MyCarrot.sol#973-1023):
        External calls:
        - _swapAndLiquify(_numTokensSellToAddToLiquidity) (MyCarrot.sol#983)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),(block.timestamp + 300)) (MyCarrot.sol#1057-1063)
        - _swapTokensForEth(_numTokensSellToAddToETH) (MyCarrot.sol#986)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),(block.timestamp + 300)) (MyCarrot.sol#1057-1063)
        External calls sending eth:
        - _swapAndLiquify(_numTokensSellToAddToLiquidity) (MyCarrot.sol#983)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
        - sent = address(marketingWallet).send(address(this).balance) (MyCarrot.sol#988)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (MyCarrot.sol#896)
                - super._transfer(from,to,transferAmount) (MyCarrot.sol#1018)
        - Transfer(from,to,amount) (MyCarrot.sol#896)
                - super._transfer(from,address(this),(marketingShare + liquidityShare)) (MyCarrot.sol#1015)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ERC20._burn(address,uint256) (MyCarrot.sol#812-824) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (MyCarrot.sol#68) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (MyCarrot.sol#70) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (MyCarrot.sol#101) is not in mixedCase
Function IUniswapV2Router01.WETH() (MyCarrot.sol#147) is not in mixedCase
Parameter MyCarrot.changeTaxForLiquidityAndMarketing(uint256,uint256,uint256,uint256)._taxBuyForLiquidity (MyCarrot.sol#1091) is not in mixedCase
Parameter MyCarrot.changeTaxForLiquidityAndMarketing(uint256,uint256,uint256,uint256)._taxBuyForMarketing (MyCarrot.sol#1091) is not in mixedCase
Parameter MyCarrot.changeTaxForLiquidityAndMarketing(uint256,uint256,uint256,uint256)._taxSellForLiquidity (MyCarrot.sol#1091) is not in mixedCase
Parameter MyCarrot.changeTaxForLiquidityAndMarketing(uint256,uint256,uint256,uint256)._taxSellForMarketing (MyCarrot.sol#1091) is not in mixedCase
Parameter MyCarrot.changeMaxTxAmount(uint256)._maxTxAmount (MyCarrot.sol#1106) is not in mixedCase
Parameter MyCarrot.changeMaxWalletAmount(uint256)._maxWalletAmount (MyCarrot.sol#1117) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in MyCarrot._transfer(address,address,uint256) (MyCarrot.sol#973-1023):
        External calls:
        - sent = address(marketingWallet).send(address(this).balance) (MyCarrot.sol#988)
        External calls sending eth:
        - _swapAndLiquify(_numTokensSellToAddToLiquidity) (MyCarrot.sol#983)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
        - sent = address(marketingWallet).send(address(this).balance) (MyCarrot.sol#988)
        State variables written after the call(s):
        - super._transfer(from,address(this),(marketingShare + liquidityShare)) (MyCarrot.sol#1015)
                - _balances[from] = fromBalance - amount (MyCarrot.sol#890)
                - _balances[to] += amount (MyCarrot.sol#893)
        - super._transfer(from,to,transferAmount) (MyCarrot.sol#1018)
                - _balances[from] = fromBalance - amount (MyCarrot.sol#890)
                - _balances[to] += amount (MyCarrot.sol#893)
        - _marketingReserves += marketingShare (MyCarrot.sol#1013)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (MyCarrot.sol#896)
                - super._transfer(from,to,transferAmount) (MyCarrot.sol#1018)
        - Transfer(from,to,amount) (MyCarrot.sol#896)
                - super._transfer(from,address(this),(marketingShare + liquidityShare)) (MyCarrot.sol#1015)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (MyCarrot.sol#152) is too similar to IUniswapV2Router01.addLiquidity(address,address,uin
t256,uint256,uint256,uint256,address,uint256).amountBDesired (MyCarrot.sol#153)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
MyCarrot._decimals (MyCarrot.sol#911) should be constant
MyCarrot._name (MyCarrot.sol#909) should be constant
MyCarrot._supply (MyCarrot.sol#912) should be constant
MyCarrot._symbol (MyCarrot.sol#910) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
MyCarrot._numTokensSellToAddToETH (MyCarrot.sol#920) should be immutable
MyCarrot._numTokensSellToAddToLiquidity (MyCarrot.sol#919) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:MyCarrot.sol analyzed (10 contracts with 93 detectors), 40 result(s) found
```

STATIC ANALYSIS



```
INFO:Detectors:
MyCarrot._addLiquidity(uint256,uint256) (MyCarrot.sol#1066-1080) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
MyCarrot.changeTaxForLiquidityAndMarketing(uint256,uint256,uint256,uint256) (MyCarrot.sol#1091-1104) should emit an event for:
        - taxBuyForLiquidity = _taxBuyForLiquidity (MyCarrot.sol#1098)
        - taxBuyForMarketing = _taxBuyForMarketing (MyCarrot.sol#1099)
        - taxSellForLiquidity = _taxSellForLiquidity (MyCarrot.sol#1100)
        - taxSellForMarketing = _taxSellForMarketing (MyCarrot.sol#1101)
MyCarrot.changeMaxTxAmount(uint256) (MyCarrot.sol#1106-1115) should emit an event for:
        - maxTxAmount = _maxTxAmount * 10 ** _decimals (MyCarrot.sol#1112)
MyCarrot.changeMaxWalletAmount(uint256) (MyCarrot.sol#1117-1126) should emit an event for:
        - maxWalletAmount = _maxWalletAmount * 10 ** _decimals (MyCarrot.sol#1123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
MyCarrot.changeMarketingWallet(address).newWallet (MyCarrot.sol#1082) lacks a zero-check on :
        - marketingWallet = newWallet (MyCarrot.sol#1087)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in MyCarrot._swapAndLiquify(uint256) (MyCarrot.sol#1035-1048):
        External calls:
        - _swapTokensForEth(half) (MyCarrot.sol#1041)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),(block.timestamp + 300)) (MyCarrot.sol#1057-1063)
        - _addLiquidity(otherHalf,newBalance) (MyCarrot.sol#1045)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
        External calls sending eth:
        - _addLiquidity(otherHalf,newBalance) (MyCarrot.sol#1045)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
        State variables written after the call(s):
        - _addLiquidity(otherHalf,newBalance) (MyCarrot.sol#1045)
                - _allowances[owner][spender] = amount (MyCarrot.sol#847)
        - _addLiquidity(otherHalf,newBalance) (MyCarrot.sol#1045)
                - inSwapAndLiquify = true (MyCarrot.sol#931)
                - inSwapAndLiquify = false (MyCarrot.sol#933)
Reentrancy in MyCarrot._transfer(address,address,uint256) (MyCarrot.sol#973-1023):
        External calls:
        - _swapAndLiquify(_numTokensSellToAddToLiquidity) (MyCarrot.sol#983)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),(block.timestamp + 300)) (MyCarrot.sol#1057-1063)
        - _swapTokensForEth(_numTokensSellToAddToETH) (MyCarrot.sol#986)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),(block.timestamp + 300)) (MyCarrot.sol#1057-1063)
        External calls sending eth:
        - _swapAndLiquify(_numTokensSellToAddToLiquidity) (MyCarrot.sol#983)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,marketingWallet,block.timestamp) (MyCarrot.sol#1072-1079)
        State variables written after the call(s):
        - _swapTokensForEth(_numTokensSellToAddToETH) (MyCarrot.sol#986)
                - _allowances[owner][spender] = amount (MyCarrot.sol#847)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0x04aff4f19acf80771024348febe887c5c5f54708be7bf9ea170add1a72a01658

2- Increase Allowance (passed):

https://testnet.bscscan.com/tx/0xf6d74fac95b75a03ffb7ab1a7595b8353bc82524cc694f22e8f7519b992929dc

3- Decrease Allowance (passed):

https://testnet.bscscan.com/tx/0xb7f13bee916abf79d16eef0649fb46a4a2915b1c0b4cf66ff1ff9ea7091b9435

4- Change Tax for Liquidity And Marketing (passed):

https://testnet.bscscan.com/tx/0xc3110b52f8d0911334f2d8ec74cd53e4a62e1686ea908812f1ade0f2faf50a8c

5- Exclude From Fee (passed):

https://testnet.bscscan.com/tx/0xe53807007981 0ef1983d49f24c5f262950d0666c3bb76fc83a6d6b505e88b266

6- Include In Fee(passed):

https://testnet.bscscan.com/tx/0x26c313b4895b0c83587172e1560127e4138f554281e1a8db7e5df1732929425f

MANUAL TESTING

Centralization – Liquidity is added to EOA.

Severity: Medium

function: _addLiquidity

Status: Open

Overview:
Liquidity is added to EOA. It may be drained by the addLiquidityETH.

```
function _addLiquidity(uint256 tokenAmount, uint256 ethAmount)
  private
  lockTheSwap
{
  _approve(address(this), address(uniswapV2Router),
tokenAmount);

  uniswapV2Router.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0,
    0,
    marketingWallet,
    block.timestamp
  );
}
```

Suggestion:
It is suggested that the address should be a contract address or a dead address.

MANUAL TESTING

Centralization – Missing Events

Severity: Low

subject: Missing Events

Status: Open

Overview:
They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function changeMarketingWallet(address newWallet)
    public
    onlyOwner
    returns (bool)
{
    marketingWallet = newWallet;
    return true;
function changeTaxForLiquidityAndMarketing(uint256
_taxBuyForLiquidity, uint256 _taxBuyForMarketing, uint256
_taxSellForLiquidity, uint256 _taxSellForMarketing)
    public
    onlyOwner
    returns (bool)
{
    require((_taxBuyForLiquidity+_taxBuyForMarketing) <= 20,
"ERC20: total tax must not be greater than 20");
    require((_taxSellForLiquidity+_taxSellForMarketing) <= 20,
"ERC20: total tax must not be greater than 20");
    taxBuyForLiquidity = _taxBuyForLiquidity;
    taxBuyForMarketing = _taxBuyForMarketing;
```

MANUAL TESTING

```
taxSellForLiquidity = _taxSellForLiquidity;
taxSellForMarketing = _taxSellForMarketing;

return true;
}

function changeMaxTxAmount(uint256 _maxTxAmount)
public
onlyOwner
returns (bool)
{
require(_maxTxAmount >= 100_000_000, "ERC20: maxTxAmount
must not be less than 0.1% total supply");
maxTxAmount = _maxTxAmount * 10**_decimals;

return true;
}
function changeMaxWalletAmount(uint256 _maxWalletAmount)
public
onlyOwner
returns (bool)
{
require(_maxWalletAmount >= 100_000_000, "ERC20:
maxWalletAmount must not be less than 0.1% total supply");
maxWalletAmount = _maxWalletAmount * 10**_decimals;

return true;
}
```

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace