auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Xmar Token

DATED : 30 November 23'

AUDIT SUMMARY

Project name – Xmar Token

Date: 24 November 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	0	0	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/0xdf5e6911f7d7a4429f52410c8218a46d3cdc39fe#code

Token Information

Token Address: -
0x2f785f38Ae0AB755570777b6A2c3395E7916f0B0

Name: XmarToken

Symbol: XT

Decimals: 18

Network: Binance smart chain

Token Type: BEP-20

Owner: - 0x7d42391EF358707B3781eca47dce8448D6a295Ac

Deployer: -
0x7d42391EF358707B3781eca47dce8448D6a295Ac

Token Supply: 100000000000000000000000000

Checksum: 511e96358f29bfd04d2679343a6d247a

Testnet version:
The tests were performed using the contract deployed on the Binance smart chain Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0xdf5e6911f7d7a4429f52410c8218a46d3cdc39fe#code

TOKEN OVERVIEW

Max tax

Buy : 6%

Sell : 10%

Transfer : 0%

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

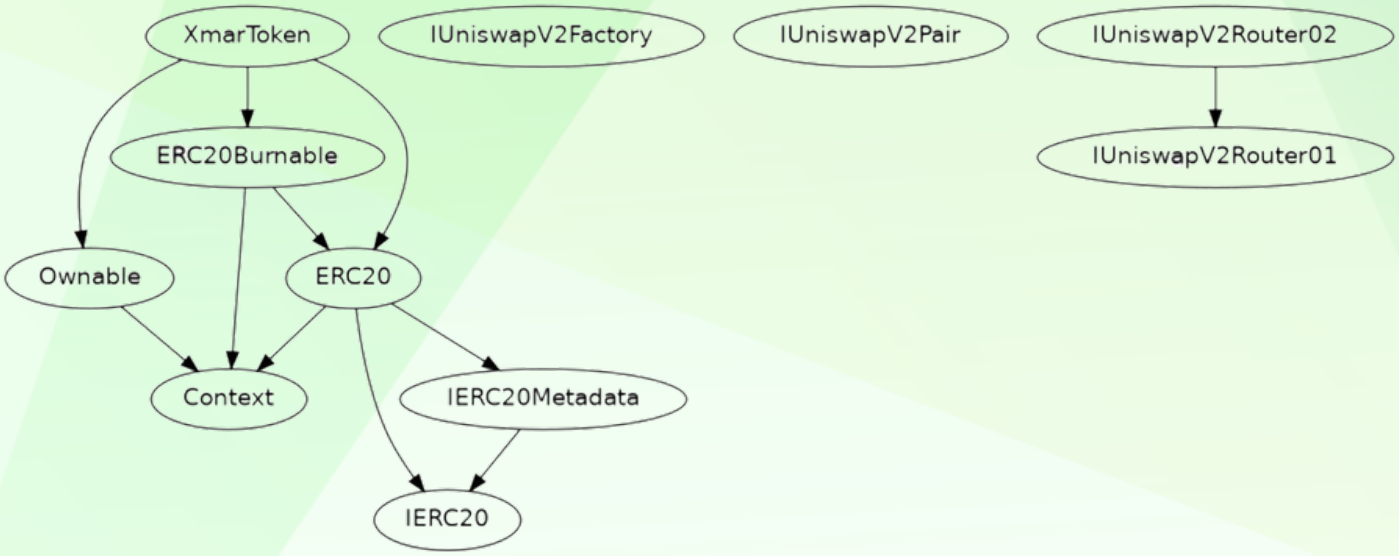
CLASSIFICATION OF RISK

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	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

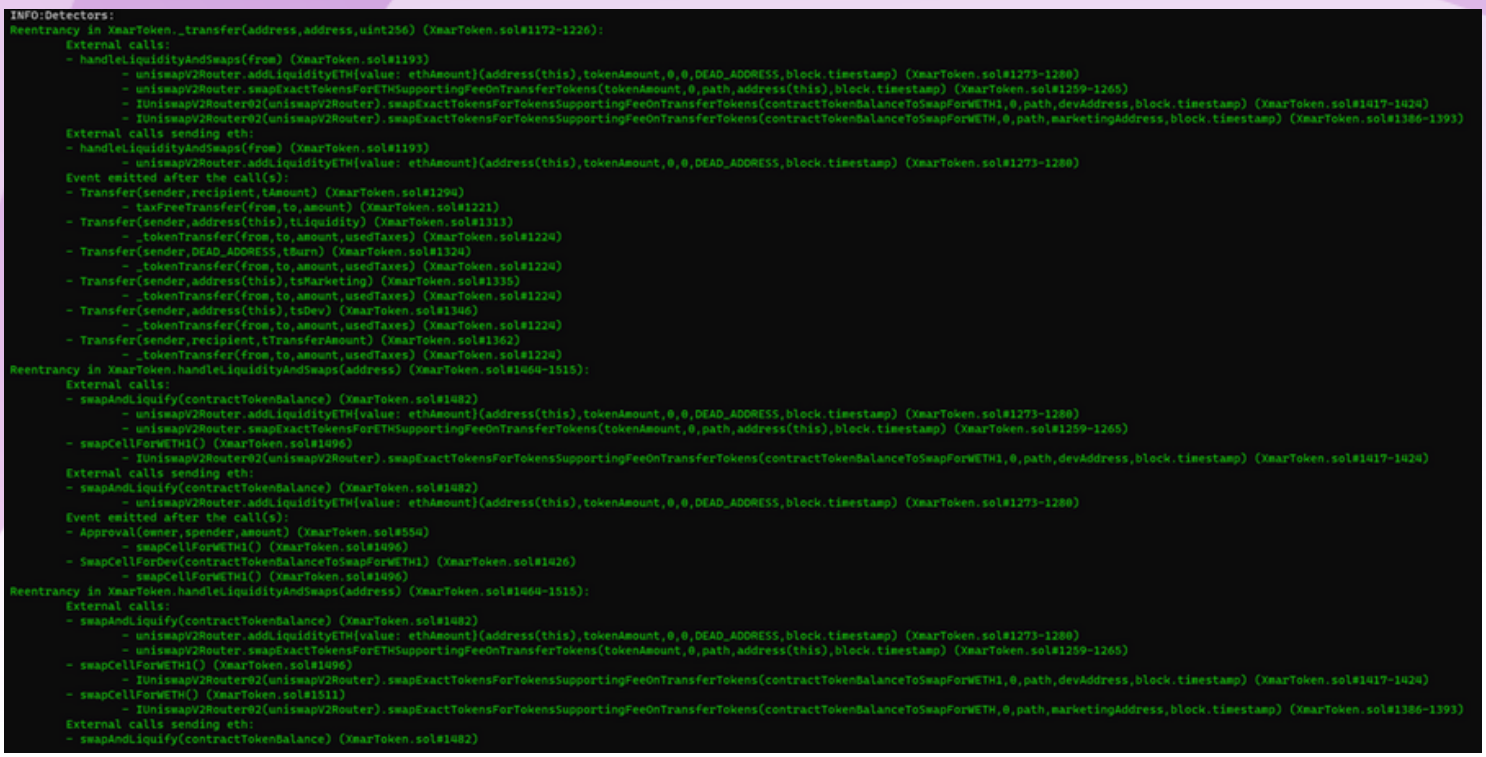
- Owner can set marketing wallet address,
- Owner can set Dev wallet address
- Owner can set any arbitary value in NumTokensSellToAddToLiquidity, NumTokensSellMarketing and NumTokensSellDev
- owner can exclude/include accounts from rewards
- owner can enable/disable swapping

STATIC ANALYSIS

```
INFO:Detectors:
XmarToken._rTotal (XmarToken.sol#942) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state
INFO:Detectors:
Pragma version0.8.19 (XmarToken.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (XmarToken.sol#717) is not in mixedCase
Function IUniswapV2Router01.WETH() (XmarToken.sol#757) is not in mixedCase
Parameter XmarToken.setMarketingAddress(address)._marketingAddress (XmarToken.sol#1075) is not in mixedCase
Parameter XmarToken.setDevAddress(address)._devAddress (XmarToken.sol#1083) is not in mixedCase
Parameter XmarToken.updateNumTokensSellToAddToLiquidity(uint256)._numTokensSellToAddToLiquidity (XmarToken.sol#1091) is not in mixedCase
Parameter XmarToken.updateNumTokensSellMarketing(uint256)._numTokensSellMarketing (XmarToken.sol#1097) is not in mixedCase
Parameter XmarToken.updateNumTokensSellDev(uint256)._numTokensSellDev (XmarToken.sol#1103) is not in mixedCase
Parameter XmarToken.setSwapAndLiquifyEnabled(bool)._enabled (XmarToken.sol#1139) is not in mixedCase
Variable XmarToken.WETH (XmarToken.sol#972) is not in mixedCase
Variable XmarToken._maxTxAmount (XmarToken.sol#987) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (XmarToken.sol#762) is too similar to IUniswapV2Router01.addLiquidity(address,address,ui
nt256,uint256,uint256,uint256,address,uint256).amountBDesired (XmarToken.sol#763)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
XmarToken.slitherConstructorVariables() (XmarToken.sol#927-1519) uses literals with too many digits:
        - _maxTxAmount = 10000000000 * 10 ** 18 (XmarToken.sol#987)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
XmarToken._allowances (XmarToken.sol#930) is never used in XmarToken (XmarToken.sol#927-1519)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Loop condition i < _excluded.length (XmarToken.sol#1156) should use cached array length instead of referencing `length` member of the storage array.
 Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cache-array-length
INFO:Detectors:
XmarToken._maxTxAmount (XmarToken.sol#987) should be constant
XmarToken._tTotal (XmarToken.sol#941) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
XmarToken.WETH (XmarToken.sol#972) should be immutable
XmarToken.uniswapV2Pair (XmarToken.sol#980) should be immutable
XmarToken.uniswapV2Router (XmarToken.sol#979) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:XmarToken.sol analyzed (11 contracts with 93 detectors), 47 result(s) found
```

```
INFO:Detectors:
XmarToken._tokenTransfer(address,address,uint256,XmarToken.Taxes) (XmarToken.sol#1298-1364) performs a multiplication on the result of a division:
        - tRfi = (tAmount * usedTaxes.cellRewards) / 100 (XmarToken.sol#1351)
        - _rTotal -= tRfi * _getRate() (XmarToken.sol#1354)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
XmarToken.addLiquidity(uint256,uint256) (XmarToken.sol#1268-1281) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToke
n.sol#1273-1280)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
XmarToken.updateNumTokensSellToAddToLiquidity(uint256) (XmarToken.sol#1090-1094) should emit an event for:
        - numTokensSellToAddToLiquidity = _numTokensSellToAddToLiquidity (XmarToken.sol#1093)
XmarToken.updateNumTokensSellMarketing(uint256) (XmarToken.sol#1096-1100) should emit an event for:
        - numTokensSellMarketing = _numTokensSellMarketing (XmarToken.sol#1099)
XmarToken.updateNumTokensSellDev(uint256) (XmarToken.sol#1102-1106) should emit an event for:
        - numTokensSellDev = _numTokensSellDev (XmarToken.sol#1105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Reentrancy in XmarToken._transfer(address,address,uint256) (XmarToken.sol#1172-1226):
        External calls:
        - handleLiquidityAndSwaps(from) (XmarToken.sol#1193)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToken.sol#1273-1280)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XmarToken.sol#1259-1265)
                - IUniswapV2Router02(uniswapV2Router).swapExactTokensForTokensSupportingFeeOnTransferTokens(contractTokenBalanceToSwapForWETH1,0,path,devAddress,block.timestamp) (XmarToken.sol#1417-1424)
                - IUniswapV2Router02(uniswapV2Router).swapExactTokensForTokensSupportingFeeOnTransferTokens(contractTokenBalanceToSwapForWETH,0,path,marketingAddress,block.timestamp) (XmarToken.sol#1386-1393)
        External calls sending eth:
        - handleLiquidityAndSwaps(from) (XmarToken.sol#1193)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToken.sol#1273-1280)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,usedTaxes) (XmarToken.sol#1224)
                - totFeesPaid.liquidity += tLiquidity (XmarToken.sol#1311)
                - totFeesPaid.cellBurn += tBurn (XmarToken.sol#1322)
                - totFeesPaid.marketing += tsMarketing (XmarToken.sol#1332)
                - totFeesPaid.dev += tsDev (XmarToken.sol#1343)
                - totFeesPaid.cellRewards += tRfi (XmarToken.sol#1355)
Reentrancy in XmarToken.handleLiquidityAndSwaps(address) (XmarToken.sol#1464-1515):
        External calls:
        - swapAndLiquify(contractTokenBalance) (XmarToken.sol#1482)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToken.sol#1273-1280)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XmarToken.sol#1259-1265)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (XmarToken.sol#1482)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToken.sol#1273-1280)
        State variables written after the call(s):
        - tradeTaxEnabled = true (XmarToken.sol#1483)
        - tradeTaxEnabled = false (XmarToken.sol#1495)
Reentrancy in XmarToken.handleLiquidityAndSwaps(address) (XmarToken.sol#1464-1515):
        External calls:
        - swapAndLiquify(contractTokenBalance) (XmarToken.sol#1482)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToken.sol#1273-1280)
```

STATIC ANALYSIS



```
INFO:Detectors:
Reentrancy in XmarToken._transfer(address,address,uint256) (XmarToken.sol#1172-1226):
        External calls:
        - handleLiquidityAndSwaps(from) (XmarToken.sol#1193)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToken.sol#1273-1280)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XmarToken.sol#1259-1265)
                - IUniswapV2Router02(uniswapV2Router).swapExactTokensForTokensSupportingFeeOnTransferTokens(contractTokenBalanceToSwapForWETH1,0,path,devAddress,block.timestamp) (XmarToken.sol#1417-1424)
                - IUniswapV2Router02(uniswapV2Router).swapExactTokensForTokensSupportingFeeOnTransferTokens(contractTokenBalanceToSwapForWETH,0,path,marketingAddress,block.timestamp) (XmarToken.sol#1386-1393)
        External calls sending eth:
        - handleLiquidityAndSwaps(from) (XmarToken.sol#1193)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToken.sol#1273-1280)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tAmount) (XmarToken.sol#1294)
                - taxFreeTransfer(from,to,amount) (XmarToken.sol#1221)
        - Transfer(sender,address(this),tLiquidity) (XmarToken.sol#1313)
                - _tokenTransfer(from,to,amount,usedTaxes) (XmarToken.sol#1224)
        - Transfer(sender,DEAD_ADDRESS,tBurn) (XmarToken.sol#1324)
                - _tokenTransfer(from,to,amount,usedTaxes) (XmarToken.sol#1224)
        - Transfer(sender,address(this),tsMarketing) (XmarToken.sol#1335)
                - _tokenTransfer(from,to,amount,usedTaxes) (XmarToken.sol#1224)
        - Transfer(sender,address(this),tsDev) (XmarToken.sol#1346)
                - _tokenTransfer(from,to,amount,usedTaxes) (XmarToken.sol#1224)
        - Transfer(sender,recipient,tTransferAmount) (XmarToken.sol#1362)
                - _tokenTransfer(from,to,amount,usedTaxes) (XmarToken.sol#1224)
Reentrancy in XmarToken.handleLiquidityAndSwaps(address) (XmarToken.sol#1464-1515):
        External calls:
        - swapAndLiquify(contractTokenBalance) (XmarToken.sol#1482)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToken.sol#1273-1280)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XmarToken.sol#1259-1265)
        - swapCellForWETH1() (XmarToken.sol#1496)
                - IUniswapV2Router02(uniswapV2Router).swapExactTokensForTokensSupportingFeeOnTransferTokens(contractTokenBalanceToSwapForWETH1,0,path,devAddress,block.timestamp) (XmarToken.sol#1417-1424)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (XmarToken.sol#1482)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToken.sol#1273-1280)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (XmarToken.sol#554)
                - swapCellForWETH1() (XmarToken.sol#1496)
        - SwapCellForDev(contractTokenBalanceToSwapForWETH1) (XmarToken.sol#1426)
                - swapCellForWETH1() (XmarToken.sol#1496)
Reentrancy in XmarToken.handleLiquidityAndSwaps(address) (XmarToken.sol#1464-1515):
        External calls:
        - swapAndLiquify(contractTokenBalance) (XmarToken.sol#1482)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,DEAD_ADDRESS,block.timestamp) (XmarToken.sol#1273-1280)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XmarToken.sol#1259-1265)
        - swapCellForWETH1() (XmarToken.sol#1496)
                - IUniswapV2Router02(uniswapV2Router).swapExactTokensForTokensSupportingFeeOnTransferTokens(contractTokenBalanceToSwapForWETH1,0,path,devAddress,block.timestamp) (XmarToken.sol#1417-1424)
        - swapCellForWETH() (XmarToken.sol#1511)
                - IUniswapV2Router02(uniswapV2Router).swapExactTokensForTokensSupportingFeeOnTransferTokens(contractTokenBalanceToSwapForWETH,0,path,marketingAddress,block.timestamp) (XmarToken.sol#1386-1393)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (XmarToken.sol#1482)
```

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/0x292061948d8f10e4464ab84e18834841e02062225e5b11d7eee3d9eceb70e522

2- Increase Allowance (passed):
https://testnet.bscscan.com/tx/0x2f6bb01f53f330eb5c1c1e5549f2a4e9bc91a7e8f7edcd3be366062b43e5c0c8

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0x4b9407bfe997c98a21914efb12b50ece7f2b95acdd8184bef51f8ce3af0bd2ed

4- Set Dev Address (passed):
https://testnet.bscscan.com/tx/0x495bda067ed751b5ff666afec6fb43579a3e69deaa32c1c3386a3ddf62c9e6db

5- Set Marketing Address (passed):
https://testnet.bscscan.com/tx/0x07f2f557d89fb2b33ed49bdf2bdf7d7ed92e8d718ca6b1ecb42ce8ed2529701a

6- Set Swap and Liquify Enabled (passed):
https://testnet.bscscan.com/tx/0x74b2a7a0b4af84f98481b3cdaa1e2bebd46fc1a2877d9f8a2712eb9f963b77fa

7- Set Trade Tax Status (passed):
https://testnet.bscscan.com/tx/0xc3f7431b40bca3e4d18b91ab6d7649ccff216243d21b353756d29de1cea99e67

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