auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

IASOLUTIONS

DATED : 11 November 23'

AUDIT SUMMARY

Project name – IASOLUTIONS

Date: 11 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	1	1
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/0x05736c781fe8129a05f6dac4f9ef47a8027bfe20

Token Information

Token Address:
0x2fB2BBd6b34cD338EF597Ea5C17324a27b7f55Be

Name: IASOLUTIONS

Symbol: IAS

Decimals: 18

Network: Binance smart chain

Token Type: BEP20

Owner: 0x568F64dA3e6e19D0e4C06e74156dFFC7AaD35858

Deployer:
0x568F64dA3e6e19D0e4C06e74156dFFC7AaD35858

Token Supply: 100000000000000000000000000

Checksum:
6f144f5f2f4594df790fef2f00fafc7a

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/0x05736c781fe8129a05f6dac4f9ef47a8027bfe20

TOKEN OVERVIEW

Buy/sell fees 10%

Transfer Fee 0%

Marketing fee and reflection fee 5%

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

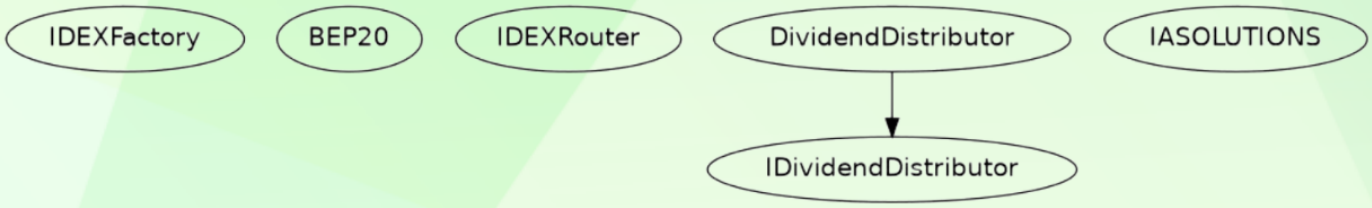
CLASSIFICATION OF RISK

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	1
Gas Optimization / Suggestions	1

INHERITANCE TREE



POINTS TO NOTE

- Owner can set pair.
- Owner can exclude wallet from fees.
-Owner can include wallets in fee.
-Owner can set dividend exempt.
-Owner can set marketing fee receiver.
-Owner can stake pool receiver.
-Owner can set buy token receiver.
-Owner can set swap back settings.
-Owner can set fees.
-Owner can do manual send.
-Owner can set distribution criteria.
-Owner can set distributor settings.
-Owner can set dividend token.
-Owner can transfer ownership.
-Owner can renounce ownership.

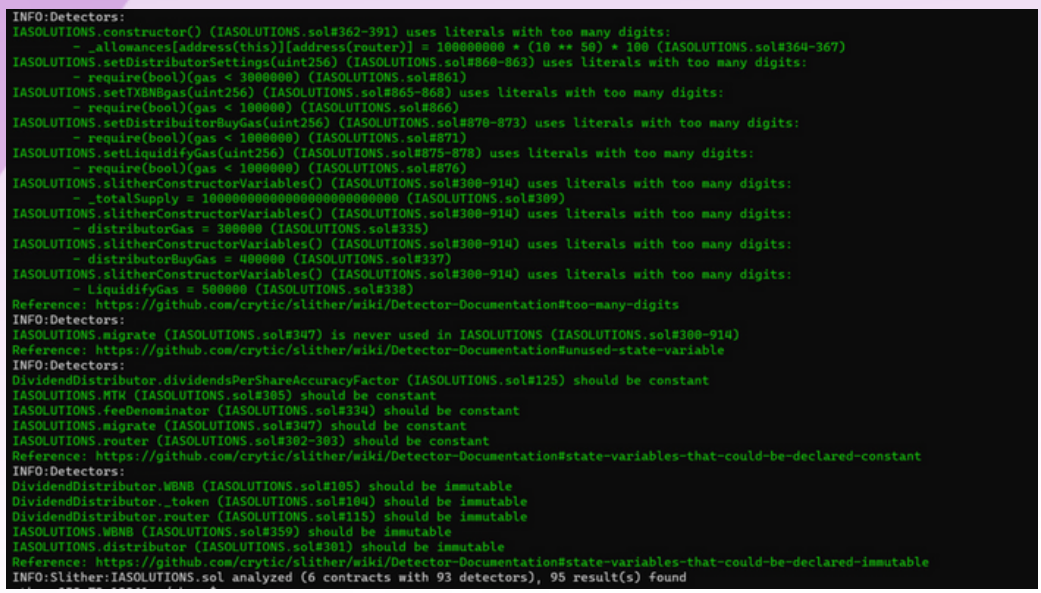
STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in DividendDistributor.distributeDividend(address) (IASOLUTIONS.sol#229-246):
        External calls:
        - REWARD.transfer(shareholder,amount) (IASOLUTIONS.sol#237)
        State variables written after the call(s):
        - shares[shareholder].totalRealised = shares[shareholder].totalRealised + amount (IASOLUTIONS.sol#239-241)
        DividendDistributor.shares (IASOLUTIONS.sol#119) can be used in cross function reentrancies:
        - DividendDistributor.distributeDividend(address) (IASOLUTIONS.sol#229-246)
        - DividendDistributor.getUnpaidEarnings(address) (IASOLUTIONS.sol#252-270)
        - DividendDistributor.setShare(address,uint256) (IASOLUTIONS.sol#162-180)
        - DividendDistributor.shares (IASOLUTIONS.sol#119)
        - shares[shareholder].totalExcluded = getCumulativeDividends(shares[shareholder].amount) (IASOLUTIONS.sol#242-244)
        DividendDistributor.shares (IASOLUTIONS.sol#119) can be used in cross function reentrancies:
        - DividendDistributor.distributeDividend(address) (IASOLUTIONS.sol#229-246)
        - DividendDistributor.getUnpaidEarnings(address) (IASOLUTIONS.sol#252-270)
        - DividendDistributor.setShare(address,uint256) (IASOLUTIONS.sol#162-180)
        - DividendDistributor.shares (IASOLUTIONS.sol#119)
Reentrancy in DividendDistributor.process(uint256) (IASOLUTIONS.sol#197-217):
        External calls:
        - distributeDividend(shareholders[currentIndex]) (IASOLUTIONS.sol#210)
                - REWARD.transfer(shareholder,amount) (IASOLUTIONS.sol#237)
        State variables written after the call(s):
        - currentIndex = 0 (IASOLUTIONS.sol#207)
        DividendDistributor.currentIndex (IASOLUTIONS.sol#129) can be used in cross function reentrancies:
        - DividendDistributor.currentIndex (IASOLUTIONS.sol#129)
        - DividendDistributor.process(uint256) (IASOLUTIONS.sol#197-217)
        - currentIndex ++ (IASOLUTIONS.sol#214)
        DividendDistributor.currentIndex (IASOLUTIONS.sol#129) can be used in cross function reentrancies:
        - DividendDistributor.currentIndex (IASOLUTIONS.sol#129)
        - DividendDistributor.process(uint256) (IASOLUTIONS.sol#197-217)
Reentrancy in DividendDistributor.setShare(address,uint256) (IASOLUTIONS.sol#162-180):
        External calls:
        - distributeDividend(shareholder) (IASOLUTIONS.sol#168)
                - REWARD.transfer(shareholder,amount) (IASOLUTIONS.sol#237)
        State variables written after the call(s):
        - shares[shareholder].amount = amount (IASOLUTIONS.sol#176)
        DividendDistributor.shares (IASOLUTIONS.sol#119) can be used in cross function reentrancies:
        - DividendDistributor.distributeDividend(address) (IASOLUTIONS.sol#229-246)
```

```
INFO:Detectors:
IASOLUTIONS.swapBack(uint256,uint256,uint256,uint256) (IASOLUTIONS.sol#678-760) ignores return value by router.addLiquidityETH{gas: LiquidifyGas,value: amou
ntBNBLiquidity}(address(this),amountToLiquidify,0,0,address(this),block.timestamp) (IASOLUTIONS.sol#746-758)
IASOLUTIONS.swapBack(uint256,uint256,uint256,uint256) (IASOLUTIONS.sol#678-760) ignores return value by router.addLiquidityETH{gas: LiquidifyGas,value: addr
ess(this).balance}(address(this),amountToLiquidify,0,0,address(this),block.timestamp) (IASOLUTIONS.sol#746-758)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
DividendDistributor.setDistributionCriteria(uint256,uint256) (IASOLUTIONS.sol#154-160) should emit an event for:
        - minPeriod = _minPeriod (IASOLUTIONS.sol#158)
        - minDistribution = _minDistribution (IASOLUTIONS.sol#159)
IASOLUTIONS.setFees(uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256,uint256) (IASOLUTIONS.sol#762-801) should emit an event for:
        - liquidityFee = _liquidityFee (IASOLUTIONS.sol#774)
        - marketingFee = _marketingFee (IASOLUTIONS.sol#775)
        - reflectionFee = _reflectionFee (IASOLUTIONS.sol#776)
        - PoolFee = _stakePoolFee (IASOLUTIONS.sol#777)
        - burnFee = _burnFee (IASOLUTIONS.sol#778)
        - buyTax = _liquidityFee + _marketingFee + _stakePoolFee + _reflectionFee + _burnFee (IASOLUTIONS.sol#780-785)
        - sellLiquidityFee = _sellLiquidityFee (IASOLUTIONS.sol#787)
        - sellReflectionFee = _sellReflectionFee (IASOLUTIONS.sol#788)
        - sellPoolFee = _sellStakePoolFee (IASOLUTIONS.sol#789)
        - sellBurnFee = _sellBurnFee (IASOLUTIONS.sol#790)
        - sellMarketingFee = _sellMarketingFee (IASOLUTIONS.sol#791)
        - sellTax = _sellLiquidityFee + _sellReflectionFee + _sellStakePoolFee + _sellBurnFee + _sellMarketingFee (IASOLUTIONS.sol#793-798)
IASOLUTIONS.setDistributorSettings(uint256) (IASOLUTIONS.sol#860-863) should emit an event for:
        - distributorGas = gas (IASOLUTIONS.sol#862)
IASOLUTIONS.setTXBNBgas(uint256) (IASOLUTIONS.sol#865-868) should emit an event for:
        - txbnbGas = gas (IASOLUTIONS.sol#867)
IASOLUTIONS.setDistributorBuyGas(uint256) (IASOLUTIONS.sol#870-873) should emit an event for:
        - distributorBuyGas = gas (IASOLUTIONS.sol#872)
IASOLUTIONS.setLiquidifyGas(uint256) (IASOLUTIONS.sol#875-878) should emit an event for:
        - LiquidifyGas = gas (IASOLUTIONS.sol#877)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
DividendDistributor.constructor(address,BEP20,address).token (IASOLUTIONS.sol#146) lacks a zero-check on :
                - _token = token (IASOLUTIONS.sol#150)
IASOLUTIONS.setmarketingFeeReceivers(address)._marketingFeeReceiver (IASOLUTIONS.sol#504) lacks a zero-check on :
                - marketingFeeReceiver = _marketingFeeReceiver (IASOLUTIONS.sol#508)
IASOLUTIONS.setStakePoolReceiver(address)._autoStakePoolReceiver (IASOLUTIONS.sol#511) lacks a zero-check on :
```

```
INFO:Detectors:
DividendDistributor.constructor(address,BEP20,address).token (IASOLUTIONS.sol#146) lacks a zero-check on :
                - _token = token (IASOLUTIONS.sol#150)
IASOLUTIONS.setmarketingFeeReceivers(address)._marketingFeeReceiver (IASOLUTIONS.sol#504) lacks a zero-check on :
                - marketingFeeReceiver = _marketingFeeReceiver (IASOLUTIONS.sol#508)
IASOLUTIONS.setStakePoolReceiver(address)._autoStakePoolReceiver (IASOLUTIONS.sol#511) lacks a zero-check on :
                - StakePoolReceiver = _autoStakePoolReceiver (IASOLUTIONS.sol#515)
IASOLUTIONS.setbuytokensReceiver(address)._buytokensReceiver (IASOLUTIONS.sol#518) lacks a zero-check on :
                - buytokensReceiver = _buytokensReceiver (IASOLUTIONS.sol#522)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
DividendDistributor.distributeDividend(address) (IASOLUTIONS.sol#229-246) has external calls inside a loop: REWARD.transfer(shareholder,amount) (IASOLUTIONS
.sol#237)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in IASOLUTIONS._transferFrom(address,address,uint256) (IASOLUTIONS.sol#585-647):
        External calls:
        - swapBack(marketingFeeAmount,liquidifyFeeAmount,stkpoolFeeAmount,refFeeAmount) (IASOLUTIONS.sol#615-620)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(amountToSwap,0,path,address(this),block.timestamp) (IASOLUTIONS.sol#713-719)
                - (success) = address(marketingFeeReceiver).call{gas: txbnbGas,value: amountBNBMarketing}() (IASOLUTIONS.sol#722-725)
                - (success_scope_0) = address(StakePoolReceiver).call{gas: txbnbGas,value: amountBNBStakePool}() (IASOLUTIONS.sol#729-732)
                - distributor.deposit{gas: distributorBuyGas,value: amountBNBReflection}() (IASOLUTIONS.sol#737-742)
                - router.addLiquidityETH{gas: LiquidifyGas,value: amountBNBLiquidity}(address(this),amountToLiquidify,0,0,address(this),block.timestamp) (IA
SOLUTIONS.sol#746-758)
                - router.addLiquidityETH{gas: LiquidifyGas,value: address(this).balance}(address(this),amountToLiquidify,0,0,address(this),block.timestamp)
(IASOLUTIONS.sol#746-758)
        External calls sending eth:
        - swapBack(marketingFeeAmount,liquidifyFeeAmount,stkpoolFeeAmount,refFeeAmount) (IASOLUTIONS.sol#615-620)
                - (success) = address(marketingFeeReceiver).call{gas: txbnbGas,value: amountBNBMarketing}() (IASOLUTIONS.sol#722-725)
                - (success_scope_0) = address(StakePoolReceiver).call{gas: txbnbGas,value: amountBNBStakePool}() (IASOLUTIONS.sol#729-732)
                - distributor.deposit{gas: distributorBuyGas,value: amountBNBReflection}() (IASOLUTIONS.sol#737-742)
                - router.addLiquidityETH{gas: LiquidifyGas,value: amountBNBLiquidity}(address(this),amountToLiquidify,0,0,address(this),block.timestamp) (IA
SOLUTIONS.sol#746-758)
                - router.addLiquidityETH{gas: LiquidifyGas,value: address(this).balance}(address(this),amountToLiquidify,0,0,address(this),block.timestamp)
(IASOLUTIONS.sol#746-758)
        State variables written after the call(s):
        - _burnIN(sender,burnFeeAmount) (IASOLUTIONS.sol#627)
                - _totalSupply = _totalSupply - amount (IASOLUTIONS.sol#492)
Reentrancy in DividendDistributor.deposit() (IASOLUTIONS.sol#182-195):
```

STATIC ANALYSIS



```
INFO:Detectors:
IASOLUTIONS.constructor() (IASOLUTIONS.sol#362-391) uses literals with too many digits:
        - _allowances[address(this)][address(router)] = 100000000 * (10 ** 50) * 100 (IASOLUTIONS.sol#364-367)
IASOLUTIONS.setDistributorSettings(uint256) (IASOLUTIONS.sol#860-863) uses literals with too many digits:
        - require(bool)(gas < 3000000) (IASOLUTIONS.sol#861)
IASOLUTIONS.setTXBNBgas(uint256) (IASOLUTIONS.sol#865-868) uses literals with too many digits:
        - require(bool)(gas < 100000) (IASOLUTIONS.sol#866)
IASOLUTIONS.setDistributorBuyGas(uint256) (IASOLUTIONS.sol#870-873) uses literals with too many digits:
        - require(bool)(gas < 1000000) (IASOLUTIONS.sol#871)
IASOLUTIONS.setLiquidifyGas(uint256) (IASOLUTIONS.sol#875-878) uses literals with too many digits:
        - require(bool)(gas < 1000000) (IASOLUTIONS.sol#876)
IASOLUTIONS.slitherConstructorVariables() (IASOLUTIONS.sol#300-914) uses literals with too many digits:
        - _totalSupply = 100000000000000000000000000 (IASOLUTIONS.sol#309)
IASOLUTIONS.slitherConstructorVariables() (IASOLUTIONS.sol#300-914) uses literals with too many digits:
        - distributorGas = 300000 (IASOLUTIONS.sol#335)
IASOLUTIONS.slitherConstructorVariables() (IASOLUTIONS.sol#300-914) uses literals with too many digits:
        - distributorBuyGas = 400000 (IASOLUTIONS.sol#337)
IASOLUTIONS.slitherConstructorVariables() (IASOLUTIONS.sol#300-914) uses literals with too many digits:
        - LiquidifyGas = 500000 (IASOLUTIONS.sol#338)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
IASOLUTIONS.migrate (IASOLUTIONS.sol#347) is never used in IASOLUTIONS (IASOLUTIONS.sol#300-914)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
DividendDistributor.dividendsPerShareAccuracyFactor (IASOLUTIONS.sol#125) should be constant
IASOLUTIONS.MTK (IASOLUTIONS.sol#305) should be constant
IASOLUTIONS.feeDenominator (IASOLUTIONS.sol#334) should be constant
IASOLUTIONS.migrate (IASOLUTIONS.sol#347) should be constant
IASOLUTIONS.router (IASOLUTIONS.sol#302-303) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
DividendDistributor.WBNB (IASOLUTIONS.sol#105) should be immutable
DividendDistributor._token (IASOLUTIONS.sol#104) should be immutable
DividendDistributor.router (IASOLUTIONS.sol#115) should be immutable
IASOLUTIONS.WBNB (IASOLUTIONS.sol#359) should be immutable
IASOLUTIONS.distributor (IASOLUTIONS.sol#301) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:IASOLUTIONS.sol analyzed (6 contracts with 93 detectors), 95 result(s) found
```

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/0x66eefb38b6613a75a7e6d0ec728abdb939a0dcb46c0fe6cef63fba27d70819f1

2- Exclude From Fee (passed):
https://testnet.bscscan.com/tx/0x342e48180d100673e1408c9e0dc67dc6df3fa07fc285421a45dad7fde706f61d

3- Include In Fee (passed):
https://testnet.bscscan.com/tx/0xac9c9913c2edd9a772d29dba692becce81eefd6ddef9e30b69d153cf145f75b5

4- Manual Send (passed):
https://testnet.bscscan.com/tx/0x53112997818e089da3ab1a52c7f3a756be938cfb16e05b663d56a1ff0a52ade2

5- Transfer passed):
https://testnet.bscscan.com/tx/0x9c98ae729b2611c88c07c083af47217e476dc150bbdcebfeb54702110b89c1de

6- Set Dividend Exempt (passed):
https://testnet.bscscan.com/tx/0x1b7ed09fcc2ec0faebb7209d9d347416351c086805a996499cfa879b7ad8535f

7- Set Fees (passed):
https://testnet.bscscan.com/tx/0x1a57ca6f2b5e666facb6b0073679e95404f8131d56d5dba65fbb4af68b57a515

8- Set Pair (passed):
https://testnet.bscscan.com/tx/0x4addc30922955d7f7a1b79945f96bf0a6ca0e77398ac78e2697c52be3c686a06

9- Set Swap Back Settings (passed):
https://testnet.bscscan.com/tx/0x7ed187ba592b7590d57ba4df8ca443dd70069befe359490e98f7671284f00c4f

MANUAL TESTING

Severity: Low
subject: floating Pragma Solidity version
Status: Open

Overview:
It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

pragma solidity ^0.8.7;

Suggestion
Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

MANUAL TESTING

Severity: Suggestion/Informational
subject: Else block
Status: Open

Overview:
The else loop does not do anything useful. Need to delete.

```
internal swapping nonReentrant(){
  uint256 a = marketing↑ + liquidity↑ + stakePool↑ + reflection↑;
  if (a <= swapThreshold) {} else {
      a = swapThreshold;
  }
```

Suggestion:
It is recommended not to use unusual block to save gas fees.

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