auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Pepe Farm

DATED : 11 May 23'

AUDIT SUMMARY

Project name – Pepe Farm

Date: 11 May, 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	1	1	1	13
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/token/0xdd7d824551cea5a11a79be16df8b9f93ed0bd242

Token Information

Token Name : Pepe Farm

Token Symbol: PEPEF

Decimals: 18

Token Supply: 420,000,000,000,000

Token Address:
0xC3665064663d9fCc535BF121aAb4eF5062004D14

Checksum:
994e9c6befef5f38a67c80ed7bfa1d87ab1975e1

Owner:
0x7030aa2347d5dA811a34AD8B1c6D2C03a05b7Db3
(at time of writing the audit)

Deployer:
0x7030aa2347d5dA811a34AD8B1c6D2C03a05b7Db3

TOKEN OVERVIEW

Fees:

Buy Fees: 6%

Sell Fees: 6%

Transfer Fees: 0%

Fees Privilege: None

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: changing internal swap settings

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

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

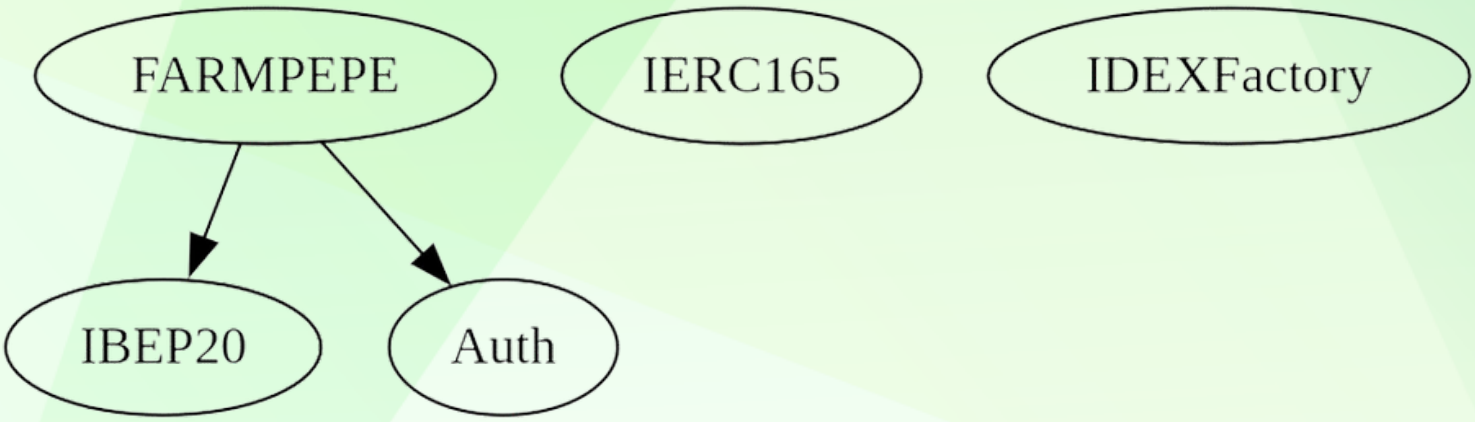
CLASSIFICATION OF RISK

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	1
Medium-Risk	1
Low-Risk	1
Gas Optimization / Suggestions	13

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to change buy/sell fees at current version of the contract (6%)
- Owner is not able to set transfer fee (0%)
- Owner is not able to set max buy/sell/transfer/hold amount
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able able to limit buys/transfers/sells by a max amount as limit
- Owner is not able to mint new tokens
- Trades are already enabled

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
DividendDistributor	Implementation	IDividendDistributor		
└	<Constructor>	Public ❗	🛑	NO ❗
└	setDistributionCriteria	External ❗	🛑	onlyToken
└	setShare	External ❗	🛑	onlyToken
└	deposit	External ❗	💵	onlyToken
└	process	External ❗	🛑	onlyToken
└	shouldDistribute	Internal 🔒		
└	distributeDividend	Internal 🔒	🛑	
└	claimDividend	External ❗	🛑	onlyToken
└	getUnpaidEarnings	Public ❗	NO ❗	
└	getCumulativeDividends	Internal 🔒		
└	addShareholder	Internal 🔒	🛑	
└	removeShareholder	Internal 🔒	🛑	
└	setDividendTokenAddress	External ❗	🛑	onlyToken
FARMPEPE	Implementation	IBEP20, Auth		
└	<Constructor>	Public ❗	🛑	Auth
└	<Receive Ether>	External ❗	💵	NO ❗
└	totalSupply	External ❗	NO ❗	
└	decimals	External ❗	NO ❗	
└	symbol	External ❗	NO ❗	
└	name	External ❗	NO ❗	
└	getOwner	External ❗	NO ❗	
└	balanceOf	Public ❗	NO ❗	
└	allowance	External ❗	NO ❗	
└	approve	Public ❗	🛑	NO ❗
└	approveMax	External ❗	🛑	NO ❗
└	savetokens	External ❗	🛑	devwall
└	burning	External ❗	🛑	NO ❗
└	transfer	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
└	_transferFrom	Internal 🔒	🛑	
└	_basicTransfer	Internal 🔒	🛑	
└	shouldTakeFee	Internal 🔒		
└	shouldTakeFeer	Internal 🔒		
└	getTotalFee	Public ❗	NO ❗	
└	getMultipliedFee	Public ❗	NO ❗	
└	takeFee	Internal 🔒	🛑	

CONTRACT ASSESMENT

└	shouldSwapBack	Internal 🔒		
└	swapBack	Internal 🔒	🛑	swapping
└	buyTokens	Internal 🔒	🛑	swapping
└	launched	Internal 🔒		
└	setFeeReceivers	External ❗	🛑	devwall
└	setSwapBackSettings	External ❗	🛑	devwall
└	setTargetLiquidity	External ❗	🛑	onlyOwner
└	manualSend	External ❗	🛑	NO ❗
└	setDistributionCriteria	External ❗	🛑	onlyOwner
└	claimDividend	External ❗	🛑	NO ❗
└	getUnpaidEarnings	Public ❗		NO ❗
└	setDistributorSettings	External ❗	🛑	onlyOwner
└	getCirculatingSupply	Public ❗		NO ❗
└	getLiquidityBacking	Public ❗		NO ❗
└	isOverLiquified	Public ❗		NO ❗
Auth	Implementation		
└	<Constructor>	Public ❗	🛑	NO ❗
└	isOwner	Public ❗		NO ❗
└	isdevwallet	Public ❗		NO ❗
└	transferOwnership	Public ❗	🛑	onlyOwner
└	renounceOwnership	Public ❗	🛑	onlyOwner
IERC165	Interface		
└	supportsInterface	External ❗		NO ❗
IBEP20	Interface		
└	totalSupply	External ❗		NO ❗
└	decimals	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	name	External ❗		NO ❗
└	getOwner	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	burning	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IDEXFactory	Interface		
└	createPair	External ❗	🛑	NO ❗
IDEXRouter	Interface			

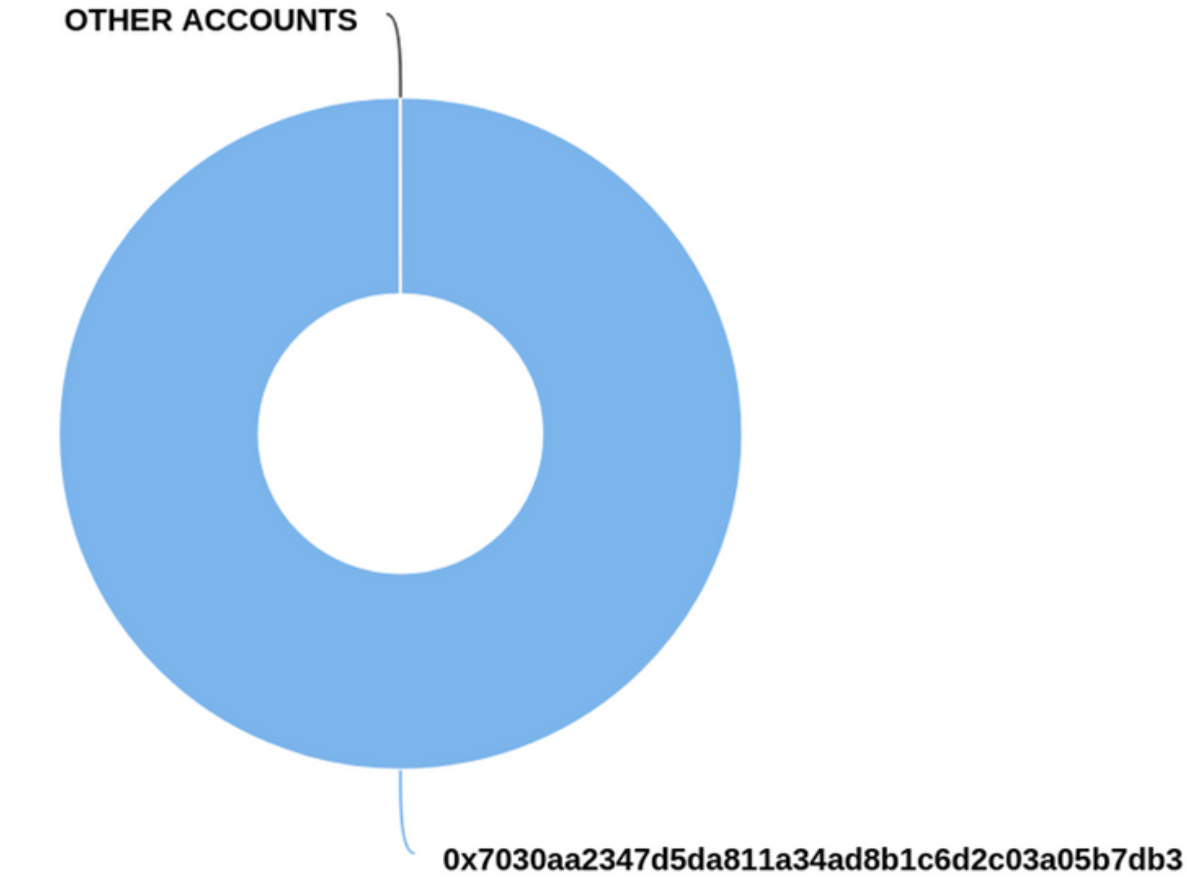
CONTRACT ASSESMENT

└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidity	External ❗	🛑	NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
IDividendDistributor	Interface			
└	setDistributionCriteria	External ❗	🛑	NO ❗
└	setShare	External ❗	🛑	NO ❗
└	deposit	External ❗	💵	NO ❗
└	process	External ❗	🛑	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

TOKENOMICS AT TIME OF AUDIT



STATIC ANALYSIS

```
FARMPEPE.maxfee (contracts/FarmPepe/Token.sol#246) is never used in FARMPEPE (contracts/FarmPepe/Token.sol#222-753)
FARMPEPE.liquidityTransferFee (contracts/FarmPepe/Token.sol#262) is never used in FARMPEPE (contracts/FarmPepe/Token.sol#222-753)
FARMPEPE.marketingTransferFee (contracts/FarmPepe/Token.sol#263) is never used in FARMPEPE (contracts/FarmPepe/Token.sol#222-753)
FARMPEPE.projectTransferFee (contracts/FarmPepe/Token.sol#264) is never used in FARMPEPE (contracts/FarmPepe/Token.sol#222-753)
FARMPEPE.valueForSwap (contracts/FarmPepe/Token.sol#276) is never used in FARMPEPE (contracts/FarmPepe/Token.sol#222-753)
FARMPEPE.token (contracts/FarmPepe/Token.sol#286) is never used in FARMPEPE (contracts/FarmPepe/Token.sol#222-753)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

DividendDistributor.WBNB (contracts/FarmPepe/Token.sol#40) should be constant
DividendDistributor.dividendsPerShareAccuracyFactor (contracts/FarmPepe/Token.sol#53) should be constant
FARMPEPE.DEAD (contracts/FarmPepe/Token.sol#227) should be constant
FARMPEPE.REWARD (contracts/FarmPepe/Token.sol#225) should be constant
FARMPEPE.StartTokens (contracts/FarmPepe/Token.sol#284) should be constant
FARMPEPE.TransferFeeDenominator (contracts/FarmPepe/Token.sol#266) should be constant
FARMPEPE.WBNB (contracts/FarmPepe/Token.sol#226) should be constant
FARMPEPE.ZERO (contracts/FarmPepe/Token.sol#228) should be constant
FARMPEPE._totalSupply (contracts/FarmPepe/Token.sol#233) should be constant
FARMPEPE.buyFeeDenominator (contracts/FarmPepe/Token.sol#252) should be constant
FARMPEPE.buyfeeburning (contracts/FarmPepe/Token.sol#253) should be constant
FARMPEPE.liquidityBuyFee (contracts/FarmPepe/Token.sol#248) should be constant
FARMPEPE.liquiditySellFee (contracts/FarmPepe/Token.sol#255) should be constant
FARMPEPE.liquidityTransferFee (contracts/FarmPepe/Token.sol#262) should be constant
FARMPEPE.marketingBuyFee (contracts/FarmPepe/Token.sol#249) should be constant
FARMPEPE.marketingSellFee (contracts/FarmPepe/Token.sol#256) should be constant
FARMPEPE.marketingTransferFee (contracts/FarmPepe/Token.sol#263) should be constant
FARMPEPE.maxfee (contracts/FarmPepe/Token.sol#246) should be constant
FARMPEPE.projectBuyFee (contracts/FarmPepe/Token.sol#250) should be constant
FARMPEPE.projectSellFee (contracts/FarmPepe/Token.sol#257) should be constant
FARMPEPE.projectTransferFee (contracts/FarmPepe/Token.sol#264) should be constant
FARMPEPE.reflectionBuyFee (contracts/FarmPepe/Token.sol#272) should be constant
FARMPEPE.reflectionSellFee (contracts/FarmPepe/Token.sol#273) should be constant
FARMPEPE.sellFeeDenominator (contracts/FarmPepe/Token.sol#259) should be constant
FARMPEPE.sellfeeburning (contracts/FarmPepe/Token.sol#260) should be constant
FARMPEPE.token (contracts/FarmPepe/Token.sol#286) should be constant
FARMPEPE.totalBuyFee (contracts/FarmPepe/Token.sol#251) should be constant
FARMPEPE.totalSellFee (contracts/FarmPepe/Token.sol#258) should be constant
FARMPEPE.totalTransferFee (contracts/FarmPepe/Token.sol#265) should be constant
FARMPEPE.valueForSwap (contracts/FarmPepe/Token.sol#276) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

Auth.devwallet (contracts/FarmPepe/auth.sol#11) should be immutable
DividendDistributor._token (contracts/FarmPepe/Token.sol#31) should be immutable
DividendDistributor.router (contracts/FarmPepe/Token.sol#41) should be immutable
FARMPEPE.Project (contracts/FarmPepe/Token.sol#280) should be immutable
FARMPEPE._maxWalletToken (contracts/FarmPepe/Token.sol#236) should be immutable
FARMPEPE.circulatingsupply_ (contracts/FarmPepe/Token.sol#234-235) should be immutable
FARMPEPE.distributor (contracts/FarmPepe/Token.sol#290) should be immutable
FARMPEPE.launchedAt (contracts/FarmPepe/Token.sol#282) should be immutable
FARMPEPE.launchedAtTimestamp (contracts/FarmPepe/Token.sol#283) should be immutable
FARMPEPE.pair (contracts/FarmPepe/Token.sol#279) should be immutable
FARMPEPE.router (contracts/FarmPepe/Token.sol#278) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

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- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0x00b8f44ea1d3582abcbcbfd8e09adcf76e790ab08d95be998f56dafc0ce93a6b

2- Buying when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0xc4c97bdcf80ea4a71a669096892a7f746252ed6e4770411640066ee61a414208

3- Selling when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x1362f04d9af9f3e41b0046ba841633051c5a30da3bf0a126ff79f99046102263

4- Transferring when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0xcf84bdc9a09513941c42c2e9ad956f728dc67561ac0f1ff5d18cc475fc92117b

5- Buying when not excluded from fees (6% tax) (passed):

https://testnet.bscscan.com/tx/0x0127283290deacf3de790388abf5b911c489859b860ad230482acf4c5efa1ba0

6- Selling when not excluded from fees (6% tax) (passed):

https://testnet.bscscan.com/tx/0xa1b23ce451ec3091f126ad4d1ffac5f2fa9f58c467f4055e5230ecef9ec5a1a0

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x9e519884ebd73cfb4731bd902e206c31461f492f8fd214b71628fe3233b1e065

8- Internal swap (passed):

fee wallets received BNB + Burning

https://testnet.bscscan.com/tx/0xa1b23ce451ec3091f126ad4d1ffac5f2fa9f58c467f4055e5230ecef9ec5a1a0

MANUAL TESTING

Logical – Setting swap threshold to 0 or too high

Severity: High

function: setSwapBackSettings

Status: Not Resolved

Overview:

If swapThreshold is set to 0 and there are 0 tokens in the contract, the sell/transfers would be failed for both owner of the contract and holders of the token (everyone). Additionaly if swapThreshold is set to a very high value, trade slippages would be increased significantly

```
function setSwapBackSettings(
    bool _enabled,
    uint256 _amount
) external devwall {
    swapEnabled = _enabled;
    swapThreshold = _amount;
```

Suggestion

To mitigate this issue make sure that swapThreshold is always greater than 1/1000000 and less than 1% of total supply.

MANUAL TESTING

Logical – dev wallet can not be transferred

Severity: Medium

Status: Not Resolved

Overview:

There is a function to change the owner, but devWallet (which has significant control over contract) can not be changed.

Suggestion

To mitigate this issue implement a function to be able to transfer devWallet as well.

MANUAL TESTING

Gas – Redundant code

Severity: Low

Status: Not Resolved

Overview:

There are many sections of the code that do not have any use in the contract and are disabled for ever this includes:

- isOverLiquified
- getLiquidityBacking
- setTargetLiquidity
- launched
- sections of swapBack function that are related to auto-liqudity
- all the codes related to max wallet and time lock

Suggestion

To mitigate this issue, delete all redundatnt sections of the code

MANUAL TESTING

Suggestions

- Change name of **savetokens** function to something that identifies its action, like "withdrawTokens"
- There are two functions that are doing exact same thing. One of this functions can be removed; **shouldTakeFee**, **shouldTakeFeer**
- Redundant function : **buyTokens**

  Event if burn and contract are not receiving any tokens (from tax) a Transfer event is still emitted, this may cause confusion for investors
- **emit Transfer(sender, address(this), feeamount2);**

  emit Transfer(sender, DEAD, amounttoburn);

  Some features are disabled in the contract, like auto-liqudity – burns and rewards, having redundant code related to this features only increases gas usage with no purpose

  Fees can not be changed, consider adding a function to be able to change fees within a safe range

Gas optimizations

Define **router** variable as constant

- Define **pair** variable as constant
- Define **REWARD**, **WBNB**, **DEAD**, **ZERO** as constant variables

Redundant code here:

```
_balances[Project] = ((_totalSupply * 100) / 100);
```

- Can be changed to

```
_balances[Project] = _totalSupply;
```

Redundant code at _transfer function

```
if (sender != pair && !isOwner(sender)) {}
```

- Redundant code at _transfer function, since emergBlock is always set to true

```
if (!authorizations[sender] && !authorizations[recipient]) {
        require(emergBlock, "Trading not open yet");
}
```

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