AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Audinals

DATED : 15 Aug 23'

High Risk

Centralization – Enabling Trades

Severity: High
function: launch
Status: Open
Overview:
The **launch** function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function launch() external onlyOwner {
    require(tradingActiveTime == 0);
    tradingActiveTime = block.number;
}
```

Suggestion

To reduce centralization and potential manipulation, consider one of the following approaches:

1. Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2. If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner.

High Risk

Centralization – Blacklisting

Severity: High
function: transferProtection
Status: Open
Overview:
Owner is able to disable sell and wallet to wallet transfers for an arbitrary wallet by using **transferProtection** function.

```
function transferProtection(
    address[] calldata _wallets,
    uint256 _enabled
) external onlyOwner {
    for (uint256 i = 0; i < _wallets.length; i++) {
        walletProtection[_wallets[i]] = _enabled;
    }
}

function _beforeTokenTransfer(address from, address to) internal view {
    require(
        walletProtection[from] == 0 || to == owner(),
        "Wallet protection enabled, please contact support"
    );
}
```

Blacklisting liquidity pool using this function can blacklist buy trnasactions.

Suggestion
Implement an automated method for blacklisting malicious wallets and define all the actions that might blacklist a wallet

High Risk

Logical – Prepare function doesn't work

Severity: High
function: prepare
Status: Open
Overview:
prepare function will not be working prior to enabling trades (calling "**launch**" function), this is due to limitations which prevents wallet from sending or receiving tokens unless one side is "**owner**" since contract is a a non-owner address and is interacting with liquidity pool, prepare function will not be working.

_transfer function:

```
if (tradingActiveTime == 0) {
    require(from == owner() || to == owner(), "Trading not yet active");
    super._transfer(from, to, amount);
}
```

Suggestion
Create a mapping which allows certain wallets to be able to transfer tokens prior to launch.

Example code:
mapping(address=>bool) allowed;

_transfer function:

```
if (tradingActiveTime == 0) {
    require(allowed[from] || allowed[to], "Trading not yet active");
    super._transfer(from, to, amount);
}
```

High Risk

Centralization – Updating dividend tracker

Severity: High
function: setDistributor
Status: Open
Overview:
setDistributor function allows owner to update dividend tracker to a new contract. Updating dividend tracker to a malicious contract could result in unexpected behaviour while trying to transfer / buy / sell tokens.

```
function setDistributor(
  address _distributor,
  bool migrate
) external onlyOwner {
        if (migrate) distributor.migrate(_distributor);
        distributor = IDividendDistributor(_distributor);
        distributor.initialize();
}
```

Suggestion
Since dividend tracker is one of core components of the contract, updating it to a malicious address could affect transfer/buy/sell transactions. Its suggested to implement a more decentralized method for updating dividend tracker such as using a governance model or a time lock contract.

High Risk

Centralization – Pool can not be finalized prior to launch

Severity: High
function: _transfer
Status: Open
Overview:
Prior to launch _transfer function prevents all wallets from sending / receiving tokens if one of sides is not owner's wallet. This prevents presale pool from being finalized prior to launch

Suggestion
Create a mapping which allows certain wallets to be able to transfer tokens prior to launch.

Example code:
mapping(address=>bool) allowed;

_transfer function:

```
if (tradingActiveTime == 0) {
  require(allowed[from] || allowed[to], "Trading not yet active");
  super._transfer(from, to, amount);
 }
```

AUDIT SUMMARY

Project name – Audinals

Date: 15 Aug, 2023

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

Audit Status: Failed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	5	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/token/0x13b16EaBD4e8cE352233e8b7fABE828DFEAc1693

Token Information

Token Name : Audinals

Token Symbol: AUDO

Decimals: 9

Token Supply: 1,000,000,000

Token Address:
0x2a52368E42a081BB46453Ffc4D562A2014438D98

Checksum:
de8a0e13482feb8a1e391439f8e92323e706c58c

Owner:
0x389346E15bd2D4CFB046E1C70911Dc1D9b9B639B
(at time of writing the audit)

Deployer:
0x389346E15bd2D4CFB046E1C70911Dc1D9b9B639B

TOKEN OVERVIEW

Fees:

Buy Fees: 5%

Sell Fees: 5%

Transfer Fees: 0%

Fees Privilege: Static

Ownership: owner

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: Initial distribution of the tokens

modifying fees

enabling trades

limiting wallet from sell/transfer

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

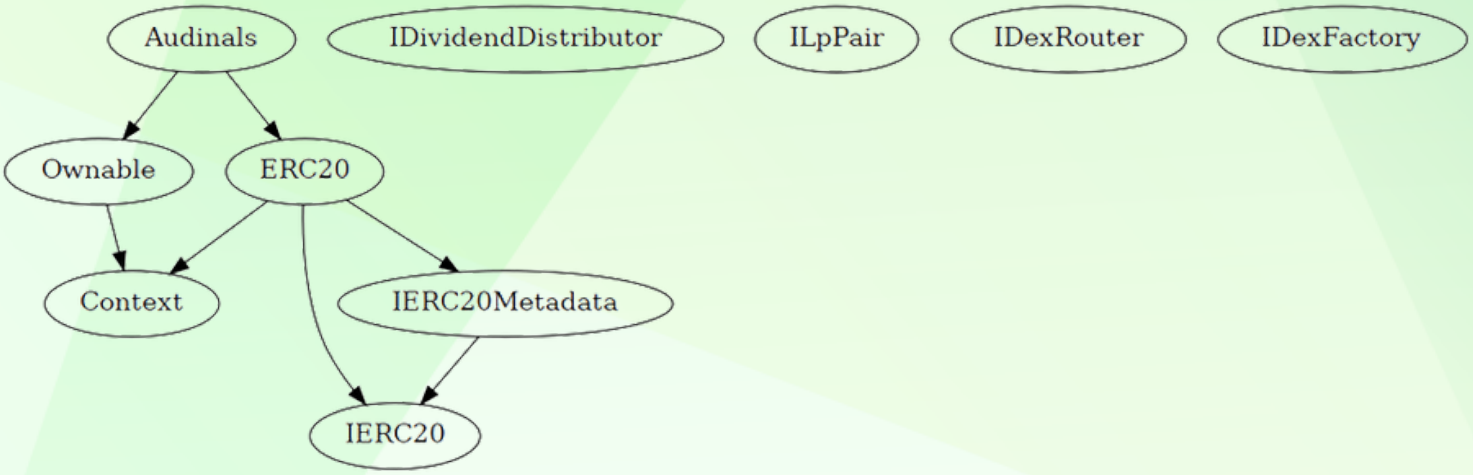
CLASSIFICATION OF RISK

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	5
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- **Owner is not able to change current fees (5% buy/sell and 0% transfer)**
- Owner is able to blacklist an arbitrary wallet
- Owner is able to disable trades
- Owner is not able to mint new tokens
- Owner is not able to set maximum wallet and maximum buy/sell/transfer limits
- **Owner must enable trades manually**

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	_transfer	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_initialTransfer	Internal 🔒	🛑	

CONTRACT ASSESMENT

Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
IDividendDistributor	Interface			
└	initialize	External ❗	🛑	NO❗
└	setDistributionCriteria	External ❗	🛑	NO❗
└	setShare	External ❗	🛑	NO❗
└	deposit	External ❗	💵	NO❗
└	claimDividend	External ❗	🛑	NO❗
└	getUnpaidEarnings	External ❗		NO❗
└	getPaidDividends	External ❗		NO❗
└	getTotalPaid	External ❗		NO❗
└	getClaimTime	External ❗		NO❗
└	getLostRewards	External ❗		NO❗
└	getTotalDividends	External ❗		NO❗
└	getTotalDistributed	External ❗		NO❗
└	getTotalSacrificed	External ❗		NO❗
└	countShareholders	External ❗		NO❗
└	migrate	External ❗	🛑	NO❗
ILpPair	Interface			
└	sync	External ❗	🛑	NO❗
IDexRouter	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactETHForTokens	External ❗	💵	NO❗
└	swapETHForExactTokens	External ❗	💵	NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	getAmountsOut	External ❗		NO❗

CONTRACT ASSESMENT

IDexFactory	Interface			
└	createPair	External ❗	🛑	NO❗
Audinals	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	<Receive Ether>	External ❗	💵	NO❗
└	decimals	Public ❗		NO❗
└	updateSwapTokens	External ❗	🛑	onlyOwner
└	toggleSwap	External ❗	🛑	onlyOwner
└	setPair	External ❗	🛑	onlyOwner
└	getSellFees	Public ❗		NO❗
└	getBuyFees	Public ❗		NO❗
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	setDividendExempt	External ❗	🛑	onlyOwner
└	_transfer	Internal 🔒	🛑	
└	swapTokensForEth	Private 🔐	🛑	
└	swapBack	Private 🔐	🛑	
└	withdrawStuckETH	External ❗	🛑	onlyOwner
└	prepare	External ❗	💵	onlyOwner
└	launch	External ❗	🛑	onlyOwner
└	setDistributor	External ❗	🛑	onlyOwner
└	setDistributionCriteria	External ❗	🛑	onlyOwner
└	manualDeposit	External ❗	💵	NO❗
└	getPoolStatistics	External ❗		NO❗
└	myStatistics	External ❗		NO❗
└	checkClaimTime	External ❗		NO❗
└	claim	External ❗	🛑	NO❗
└	airdropToWallets	External ❗	🛑	onlyOwner
└	transferProtection	External ❗	🛑	onlyOwner
└	_beforeTokenTransfer	Internal 🔒		

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Audinals._transfer(address,address,uint256) (contracts/Token.sol#532-587):
        External calls:
        - swapBack(amount) (contracts/Token.sol#548)
                - dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#594-600)
                - distributor.deposit{value: ethBalance}() (contracts/Token.sol#615)
        - distributor.setShare(from,0) (contracts/Token.sol#563)
        External calls sending eth:
        - swapBack(amount) (contracts/Token.sol#548)
                - distributor.deposit{value: ethBalance}() (contracts/Token.sol#615)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (contracts/Token.sol#255)
                - super._transfer(from,to,amount) (contracts/Token.sol#576)
        - Transfer(sender,recipient,amount) (contracts/Token.sol#255)
                - super._transfer(from,address(this),fees) (contracts/Token.sol#570)
Reentrancy in Audinals.prepare(uint256,uint256) (contracts/Token.sol#626-650):
        External calls:
        - lpPair = IDexFactory(dexRouter.factory()).createPair(ETH,address(this)) (contracts/Token.sol#633-636)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (contracts/Token.sol#255)
                - super._transfer(msg.sender,address(this),tokens * _decimalFactor) (contracts/Token.sol#640)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Audinals._transfer(address,address,uint256) (contracts/Token.sol#532-587) has a high cyclomatic complexity (18).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
Context._msgData() (contracts/Token.sol#21-24) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.17 (contracts/Token.sol#14) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Audinals.withdrawStuckETH() (contracts/Token.sol#619-624):
        - (success,None) = address(msg.sender).call{value: address(this).balance}() (contracts/Token.sol#621-623)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IDexRouter.WETH() (contracts/Token.sol#365) is not in mixedCase
Parameter Audinals.setDistributor(address,bool)._distributor (contracts/Token.sol#658) is not in mixedCase
Parameter Audinals.setDistributionCriteria(uint256,uint256,uint256)._minPeriod (contracts/Token.sol#668) is not in mixedCase
Parameter Audinals.setDistributionCriteria(uint256,uint256,uint256)._minDistribution (contracts/Token.sol#669) is not in mixedCase
Parameter Audinals.setDistributionCriteria(uint256,uint256,uint256)._claimAfter (contracts/Token.sol#670) is not in mixedCase
Parameter Audinals.transferProtection(address[],uint256)._wallets (contracts/Token.sol#738) is not in mixedCase
Parameter Audinals.transferProtection(address[],uint256)._enabled (contracts/Token.sol#739) is not in mixedCase
Constant Audinals._decimals (contracts/Token.sol#427) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Audinals._decimalFactor (contracts/Token.sol#428) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (contracts/Token.sol#22)" inContext (contracts/Token.sol#16-25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Slither:./contracts/Token.sol analyzed (10 contracts with 88 detectors), 27 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- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0xb2de25427412c30d8b994600deed255808f806eac8f9c96707b06cd6343a11d5

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

https://testnet.bscscan.com/tx/0x27d1413ba1d99b4de77576849755c0b245c10a8464fda69ec1a0d3073c72e866

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

https://testnet.bscscan.com/tx/0x0e6bbcc15101a515e2dd0cf259c1b4e672672c0890a3c820a13ece28572c7c5a

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

https://testnet.bscscan.com/tx/0x555ca953a856d3d3fa80ce20a8efd33a079a6cde75a0528591bfc040b7f6e19d

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

https://testnet.bscscan.com/tx/0xa0d211b1f09bc0f29c1736e03cbabb02cea2cd831afd90f7bcc2d7f9152c13e3

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

https://testnet.bscscan.com/tx/0x826d6d03c0537eb9163c6fc20648029b8ff20b60e41c1d7ac14bc9317ac4893c

FUNCTIONAL TESTING

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

https://testnet.bscscan.com/tx/0x05530a91bc7b6a1cfc8c2a3e681d0df7f1d6e8d4ea243012f5ad7c9863e521eb

8- Internal swap (ETH sent to dividend tracker) (passed):

https://testnet.bscscan.com/tx/0x826d6d03c0537eb9163c6fc20648029b8ff20b60e41c1d7ac14bc9317ac4893c

High Risk

Centralization – Enabling Trades

Severity: High
function: launch
Status: Open
Overview:
The **launch** function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function launch() external onlyOwner {
    require(tradingActiveTime == 0);
    tradingActiveTime = block.number;
}
```

Suggestion

To reduce centralization and potential manipulation, consider one of the following approaches:

1. Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2. If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner.

High Risk

Centralization – Blacklisting

Severity: High
function: transferProtection
Status: Open
Overview:
Owner is able to disable sell and wallet to wallet transfers for an arbitrary wallet by using **transferProtection** function.

```
function transferProtection(
    address[] calldata _wallets,
    uint256 _enabled
) external onlyOwner {
    for (uint256 i = 0; i < _wallets.length; i++) {
        walletProtection[_wallets[i]] = _enabled;
    }
}

function _beforeTokenTransfer(address from, address to) internal view {
    require(
        walletProtection[from] == 0 || to == owner(),
        "Wallet protection enabled, please contact support"
    );
}
```

Blacklisting liquidity pool using this function can blacklist buy trnasactions.

Suggestion
Implement an automated method for blacklisting malicious wallets and define all the actions that might blacklist a wallet

High Risk

Logical – Prepare function doesn't work

Severity: High
function: prepare
Status: Open
Overview:
prepare function will not be working prior to enabling trades (calling "**launch**" function), this is due to limitations which prevents wallet from sending or receiving tokens unless one side is "**owner**" since contract is a a non-owner address and is interacting with liquidity pool, prepare function will not be working.

_transfer function:

```
if (tradingActiveTime == 0) {
    require(from == owner() || to == owner(), "Trading not yet active");
    super._transfer(from, to, amount);
}
```

Suggestion
Create a mapping which allows certain wallets to be able to transfer tokens prior to launch.

Example code:
mapping(address=>bool) allowed;

_transfer function:

```
if (tradingActiveTime == 0) {
    require(allowed[from] || allowed[to], "Trading not yet active");
    super._transfer(from, to, amount);
}
```

High Risk

Centralization – Updating dividend tracker

Severity: High
function: setDistributor
Status: Open
Overview:
setDistributor function allows owner to update dividend tracker to a new contract. Updating dividend tracker to a malicious contract could result in unexpected behaviour while trying to transfer / buy / sell tokens.

```
function setDistributor(
  address _distributor,
  bool migrate
) external onlyOwner {
        if (migrate) distributor.migrate(_distributor);
        distributor = IDividendDistributor(_distributor);
        distributor.initialize();
}
```

Suggestion
Since dividend tracker is one of core components of the contract, updating it to a malicious address could affect transfer/buy/sell transactions. Its suggested to implement a more decentralized method for updating dividend tracker such as using a governance model or a time lock contract.

High Risk

Centralization – Pool can not be finalized prior to launch

Severity: High
function: _transfer
Status: Open
Overview:
Prior to launch _transfer function prevents all wallets from sending / receiving tokens if one of sides is not owner's wallet. This prevents presale pool from being finalized prior to launch

Suggestion
Create a mapping which allows certain wallets to be able to transfer tokens prior to launch.

Example code:
mapping(address=>bool) allowed;

_transfer function:

```
if (tradingActiveTime == 0) {
  require(allowed[from] || allowed[to], "Trading not yet active");
  super._transfer(from, to, amount);
 }
```

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