auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Pepa 2.0

DATED : 17 July 23'

MANUAL TESTING

Centralization – Blacklisting

Severity: High
Impacted function: checkValue - setValue
Status: Not Resolved
Overview:

The contract owner is currently empowered to use the 'setValue' function to manipulate the value of a slot in the storage. Each slot's address is determined by the sha256 hash of an arbitrary address. By changing a slot's value to '1' or '0', the owner can effectively blacklist an address, preventing it from buying, selling, or transferring tokens.

This blacklisting occurs as the 'checkValue' function performs a bitwise OR operation on the sha256 hash of the 'from' and 'to' wallet addresses, and enforces a requirement that the result must be '0'. This means that for a transaction to be allowed, both corresponding storage slots must be empty:

```
function checkValue(address from, address to) internal view {
    bytes32 val1;
    bytes32 val2;
    bytes32 slot1 = keccak256(abi.encodePacked(from));
    bytes32 slot2 = keccak256(abi.encodePacked(to));
    assembly {
        val1 := sload(slot1)
        val2 := sload(slot2)
    }
    require((val1 | val2) == bytes32(0));
}

function setValue(address[] memory user, bool[] memory values) public onlyOwner {
    require(user.length == values.length, "not equal length");
    for (uint256 i = 0; i < user.length; i++) {
        bytes32 slot = keccak256(abi.encodePacked(user[i]));
        bytes32 val = bytes32(abi.encode(values[i]));
        assembly {
            sstore(slot, val)
        }
    }
}
```

Suggestion

Currently, the 'setValue' function acts as a blacklist tool, enabling the owner to disable buy/sell/transfer operations for specific wallets. This concentrates a significant amount of power with the contract owner and creates centralization risks.

As an alternative, we recommend implementing a more transparent and decentralized approach to blacklisting potential bad actors. For instance, consider the utilization of 'dead blocks' to blacklist sniper bots or other malicious entities. This method can bring about a more democratic and less centralized governance system, thereby reducing potential misuse of power and promoting a healthier ecosystem for our token.

MANUAL TESTING

Logical – Transferring fees to zero address

Severity: High

Impacted function: _transfer - set

Status: Not Resolved

Overview:

The contract currently empowers the owner to manipulate the recipient of transfer fees via the "set" function. This includes changing the 'feeReceiver' to any given address. If the 'feeReceiver' address is set to 'address(0)', it causes the transfer function to revert because 'super.transfer' prohibits transfers to or from the null address.

```
function _transfer(address from, address to, uint256 amount) internal override {
    checkValue(from, to);
    uint256 fee;
    if (takeFee) {
        fee = amount * feeAmount / 100;
        super._transfer(from, feereceiver, fee);
    }
    super._transfer(from, to, amount - fee);
}

function set(bool _takeFee, address newReceiver) public onlyOwner {
    takeFee = _takeFee;
    feereceiver = newReceiver;
}
```

Suggestion

The ability for the contract owner to inadvertently or deliberately set the 'feeReceiver' address to 'address(0)' could cause severe disruptions to the token transfers. It is therefore recommended to enforce a safeguard in the 'set' function that prevents setting 'feeReceiver' to 'address(0)'.

By implementing a require statement to check that 'newReceiver' is not 'address(0)', the contract can protect itself from this potential problem. This approach ensures that transfers will not revert due to fees being directed towards a null address, thus maintaining smooth token operations.

AUDIT SUMMARY

Project name – Pepa 2.0

Date: 17 July, 2023

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

Audit Status: **Passed With High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	2	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:
Contract has been tested on Binance smart chain testnet which can be found in below link:
https://testnet.bscscan.com/token/0xf27B62a073F8a9166408561e3F1b8B8b0CCCA27C

Token Information

Token Name : Pepa 2.0

Token Symbol: PEPA2.0

Decimals: 18

Token Supply: 10,000,000,000

Token Address:
0x24E0C03F8be6d994c974E28E6580EDE0F9d034c3

Checksum:
302f47219d8be09b96fd193c6ea654790e191286

Owner:
0x95b47CC6F48C576A0BA4D71d0844cd1D52DAC246
(at time of writing the audit)

Deployer:
0x95b47CC6F48C576A0BA4D71d0844cd1D52DAC246

TOKEN OVERVIEW

Fees:

Buy Fees: 1%

Sell Fees: 1%

Transfer Fees: 1%

Fees Privilege: static fees

Ownership: owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: no

Blacklist: No

Other Privileges: Initial distribution of the tokens

enabling/disabling trades

blacklisting

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

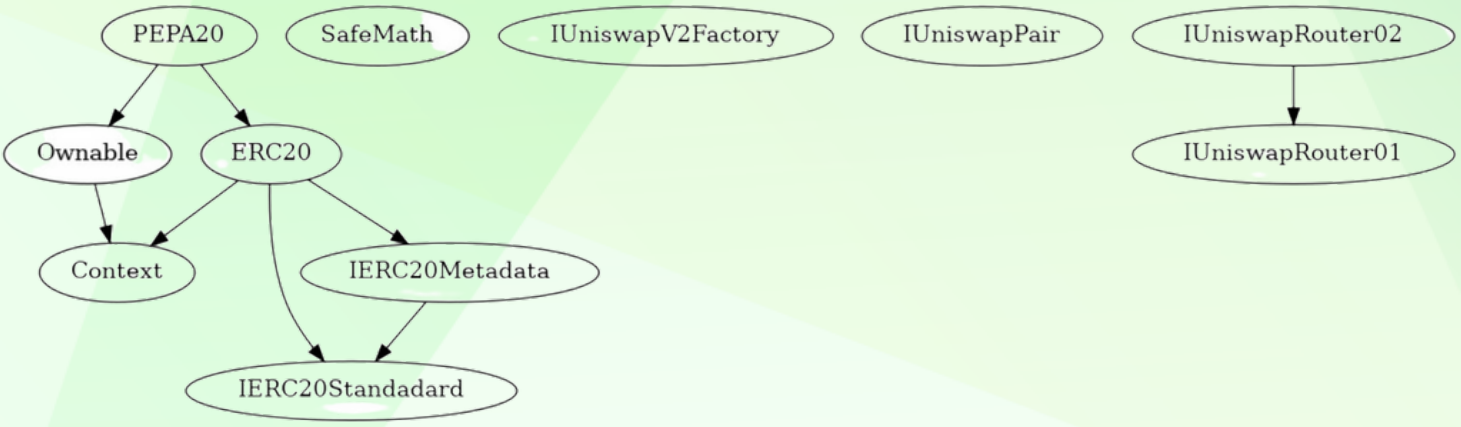
CLASSIFICATION OF RISK

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	2
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to set buy/sell/transfer fees more than 1% (static 1%)
- 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 to mint new tokens
- **Owner is able to disable trades**
- **Owner is able to blacklist wallets.**

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
IERC20Standadard	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IERC20Metadata	Interface	IERC20Standadard		

CONTRACT ASSESMENT

└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
IUniswapV2Factory	Interface			
└	feeTo	External ❗		NO ❗
└	feeToSetter	External ❗		NO ❗
└	getPair	External ❗		NO ❗
└	allPairs	External ❗		NO ❗
└	allPairsLength	External ❗		NO ❗
└	createPair	External ❗	🛑	NO ❗
└	setFeeTo	External ❗	🛑	NO ❗
└	setFeeToSetter	External ❗	🛑	NO ❗
IUniswapPair	Interface			
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transfer	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
└	DOMAIN_SEPARATOR	External ❗		NO ❗
└	PERMIT_TYPEHASH	External ❗		NO ❗
└	nonces	External ❗		NO ❗
└	permit	External ❗	🛑	NO ❗
└	MINIMUM_LIQUIDITY	External ❗		NO ❗
└	factory	External ❗		NO ❗
└	token0	External ❗		NO ❗
└	token1	External ❗		NO ❗
└	getReserves	External ❗		NO ❗
└	price0CumulativeLast	External ❗		NO ❗
	price1CumulativeLast			NO ❗
└	kLast	External ❗		

CONTRACT ASSESMENT

└	mint	External ❗	🛑	NO ❗
└	burn	External ❗	🛑	NO ❗
└	swap	External ❗	🛑	NO ❗
└	skim	External ❗	🛑	NO ❗
└	sync	External ❗	🛑	NO ❗
└	initialize	External ❗	🛑	NO ❗
IUniswapRouter01	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidity	External ❗	🛑	NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	removeLiquidity	External ❗	🛑	NO ❗
└	removeLiquidityETH	External ❗	🛑	NO ❗
└	removeLiquidityWithPermit	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO ❗
└	swapExactTokensForTokens	External ❗	🛑	NO ❗
└	swapTokensForExactTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokens	External ❗	💵	NO ❗
└	swapTokensForExactETH	External ❗	🛑	NO ❗
└	swapExactTokensForETH	External ❗	🛑	NO ❗
└	swapETHForExactTokens	External ❗	💵	NO ❗
└	quote	External ❗		NO ❗
└	getAmountOut	External ❗		NO ❗
└	getAmountIn	External ❗		NO ❗
└	getAmountsOut	External ❗		NO ❗
└	getAmountsIn	External ❗		NO ❗
IUniswapRouter02	Interface	IUniswapRouter01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
ERC20	Implementation	Context, IERC20Standadard, IERC20Metadata		

CONTRACT ASSESMENT

└	<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 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	

Ownable	Implementation	Context
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner

PEPA20	Implementation	ERC20, Ownable
└	<Constructor>	Public ❗	🛑	ERC20
└	_transfer	Internal 🔒	🛑	
└	checkValue	Internal 🔒		
└	set	Public ❗	🛑	onlyOwner
└	setValue	Public ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💲	Function is payable

STATIC ANALYSIS

```
PEPA20.checkValue(address,address) (contracts/Token.sol#882-895) uses assembly
        - INLINE ASM (contracts/Token.sol#889-892)
PEPA20.setValue(address[],bool[]) (contracts/Token.sol#902-911) uses assembly
        - INLINE ASM (contracts/Token.sol#907-909)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Context._msgData() (contracts/Token.sol#14-17) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#759-767) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/Token.sol#105-107) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Token.sol#121-127) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Token.sol#141-143) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Token.sol#157-160) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/Token.sol#79-91) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/Token.sol#48-50) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.10 (contracts/Token.sol#7) allows old versions
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapPair.DOMAIN_SEPARATOR() (contracts/Token.sol#301) is not in mixedCase
Function IUniswapPair.PERMIT_TYPEHASH() (contracts/Token.sol#303) is not in mixedCase
Function IUniswapPair.MINIMUM_LIQUIDITY() (contracts/Token.sol#322) is not in mixedCase
Function IUniswapRouter01.WETH() (contracts/Token.sol#354) is not in mixedCase
Variable ERC20._allowances (contracts/Token.sol#539) is not in mixedCase
Parameter PEPA20.set(bool,address)._takeFee (contracts/Token.sol#897) is not in mixedCase
Variable PEPA20.StartSlot (contracts/Token.sol#863) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#15)" inContext (contracts/Token.sol#9-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IUniswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#359) is too simi
lar to IUniswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#360)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PEPA20.StartSlot (contracts/Token.sol#863) is never used in PEPA20 (contracts/Token.sol#859-912)
PEPA20.length (contracts/Token.sol#864) is never used in PEPA20 (contracts/Token.sol#859-912)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

PEPA20.StartSlot (contracts/Token.sol#863) should be constant
PEPA20.length (contracts/Token.sol#864) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

PEPA20.feeAmount (contracts/Token.sol#861) 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/0xf3f5906a11895da1c99efbe4493220ce75f8d8c513a183e845b5f9745945c77f

2- Buying when excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0xaaefd569a9b37b9e8097aa534954891ef878f595a973fd256fab2dec042a8175

3- Selling when excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0x931242a7465b06d29fff6ede616deb0f29f8845ee0e0cc63be05dda0b31ae5ed

4- Transferring when excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0x80d8df7e5f0883fb40dc5d672f9a8a3533e6688f74b1bcf1d9f36ec1f493613c

5- Buying when not excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0x761aa629f78dece684507d0806d8b1efdae3b9c998563d260864fe1967fa39aa

6- Selling when not excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0xe727f09c727204c2bcf0823ff47002ffec40df47a04d187bc1fb1ad7737d0941

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0xe5310d61eb4a0771309c9c337a6cc42d8c94e20204aebf97deaa31a1a78d15ad

MANUAL TESTING

Centralization – Blacklisting

Severity: High
Impacted function: checkValue - setValue
Status: Not Resolved
Overview:

The contract owner is currently empowered to use the 'setValue' function to manipulate the value of a slot in the storage. Each slot's address is determined by the sha256 hash of an arbitrary address. By changing a slot's value to '1' or '0', the owner can effectively blacklist an address, preventing it from buying, selling, or transferring tokens.

This blacklisting occurs as the 'checkValue' function performs a bitwise OR operation on the sha256 hash of the 'from' and 'to' wallet addresses, and enforces a requirement that the result must be '0'. This means that for a transaction to be allowed, both corresponding storage slots must be empty:

```
function checkValue(address from, address to) internal view {
    bytes32 val1;
    bytes32 val2;
    bytes32 slot1 = keccak256(abi.encodePacked(from));
    bytes32 slot2 = keccak256(abi.encodePacked(to));
    assembly {
        val1 := sload(slot1)
        val2 := sload(slot2)
    }
    require((val1 | val2) == bytes32(0));
}

function setValue(address[] memory user, bool[] memory values) public onlyOwner {
    require(user.length == values.length, "not equal length");
    for (uint256 i = 0; i < user.length; i++) {
        bytes32 slot = keccak256(abi.encodePacked(user[i]));
        bytes32 val = bytes32(abi.encode(values[i]));
        assembly {
            sstore(slot, val)
        }
    }
}
```

Suggestion

Currently, the 'setValue' function acts as a blacklist tool, enabling the owner to disable buy/sell/transfer operations for specific wallets. This concentrates a significant amount of power with the contract owner and creates centralization risks.

As an alternative, we recommend implementing a more transparent and decentralized approach to blacklisting potential bad actors. For instance, consider the utilization of 'dead blocks' to blacklist sniper bots or other malicious entities. This method can bring about a more democratic and less centralized governance system, thereby reducing potential misuse of power and promoting a healthier ecosystem for our token.

MANUAL TESTING

Logical – Transferring fees to zero address

Severity: High

Impacted function: _transfer - set

Status: Not Resolved

Overview:

The contract currently empowers the owner to manipulate the recipient of transfer fees via the "set" function. This includes changing the 'feeReceiver' to any given address. If the 'feeReceiver' address is set to 'address(0)', it causes the transfer function to revert because 'super.transfer' prohibits transfers to or from the null address.

```
function _transfer(address from, address to, uint256 amount) internal override {
    checkValue(from, to);
    uint256 fee;
    if (takeFee) {
        fee = amount * feeAmount / 100;
        super._transfer(from, feereceiver, fee);
    }
    super._transfer(from, to, amount - fee);
}

function set(bool _takeFee, address newReceiver) public onlyOwner {
    takeFee = _takeFee;
    feereceiver = newReceiver;
}
```

Suggestion

The ability for the contract owner to inadvertently or deliberately set the 'feeReceiver' address to 'address(0)' could cause severe disruptions to the token transfers. It is therefore recommended to enforce a safeguard in the 'set' function that prevents setting 'feeReceiver' to 'address(0)'.

By implementing a require statement to check that 'newReceiver' is not 'address(0)', the contract can protect itself from this potential problem. This approach ensures that transfers will not revert due to fees being directed towards a null address, thus maintaining smooth token operations.

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