AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

GCCOIN

DATED : 17 Jan 2023

MANUAL TESTING

Centralization – Enabling Trades.

Severity: High

Function: EnableTrading

Status: Open

Overview:

The EnableTrading 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 enabledTrading() external onlyOwner() {
require(!tradeEnable,"trading is already open");
    _SwapBackEnable = true;
     tradeEnable = true;
    emit TradingOpenUpdated();
  }
```

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 give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

AUDIT SUMMARY

Project name – GCCOIN

Date: 17 Jan 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	1	0	2	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/0xf60d9CE4d71aA03bDc219D06c1a98f024e4ED897#code

Token Information

Token Address:
0x2D8269Dae518e78D95110dbFADf1fb479b8152e7

Name: GCCOIN

Symbol: GCC

Decimals: 9

Network: BscScan

Token Type: BEP-20

Owner: 0x59b1E916ff33241b88De2907cBf3Df166A58c19e

Deployer:
0x59b1E916ff33241b88De2907cBf3Df166A58c19e

Token Supply: 1000000000

Checksum: A67acbefe2a12642d388659dffd20722

Testnet:
https://testnet.bscscan.com/address/0xf60d9CE4d71aA03bDc219D06c1a98f024e4ED897#code

TOKEN OVERVIEW

Fees:

Buy Fee: 5-5%

Sell Fee: 5-5%

Transfer Fee: 0-0%

Fees Privilege: Owner

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges:

- Whitelist to transfer without enabling trades
- Enabling trades

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.

auditace.tech

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

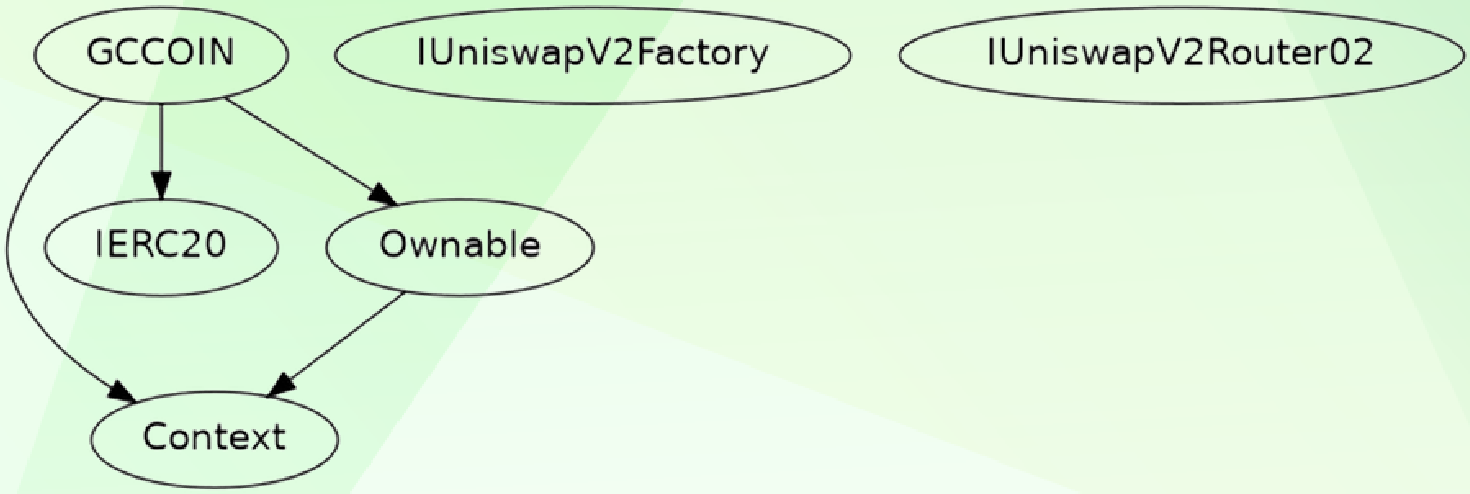
CLASSIFICATION OF RISK

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	0
◆ Low-Risk	2
◆ Gas Optimization / Suggestions	1

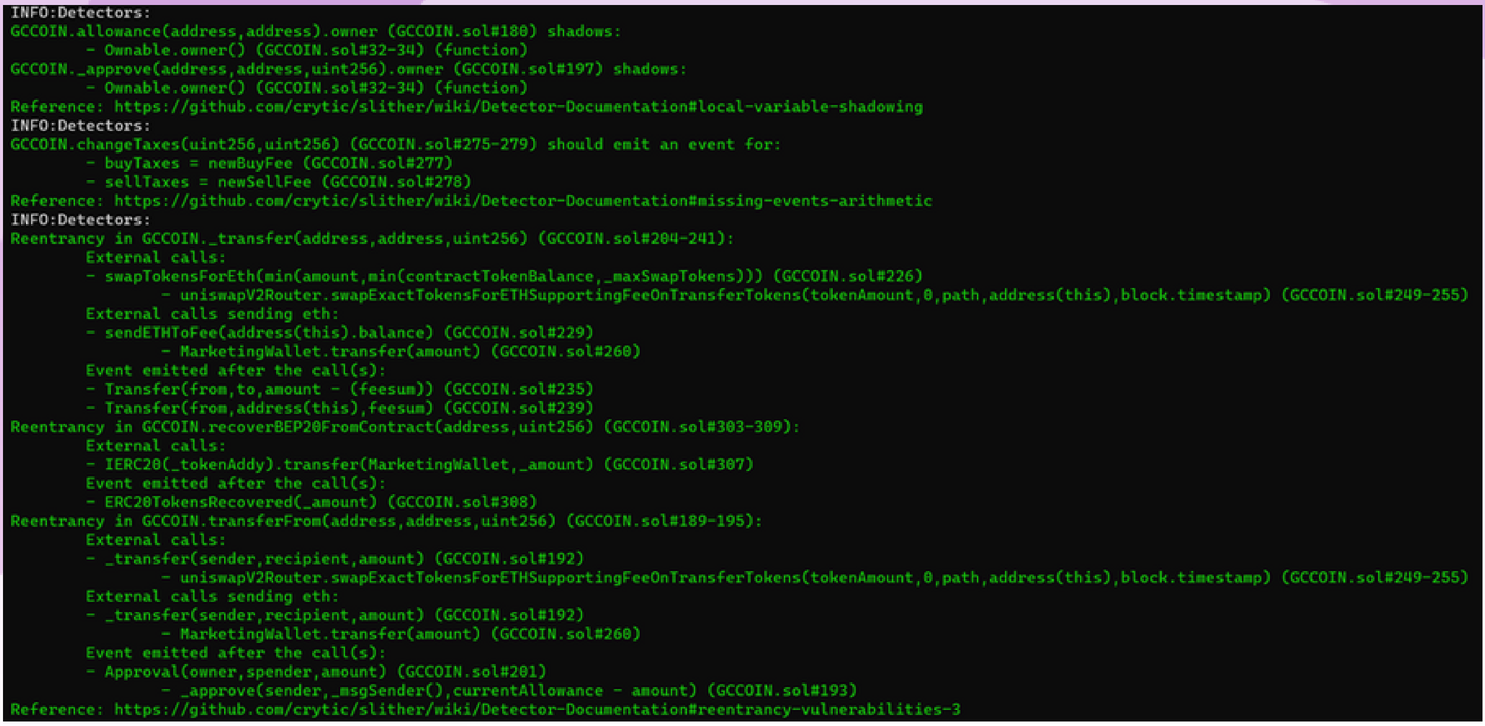
INHERITANCE TREE



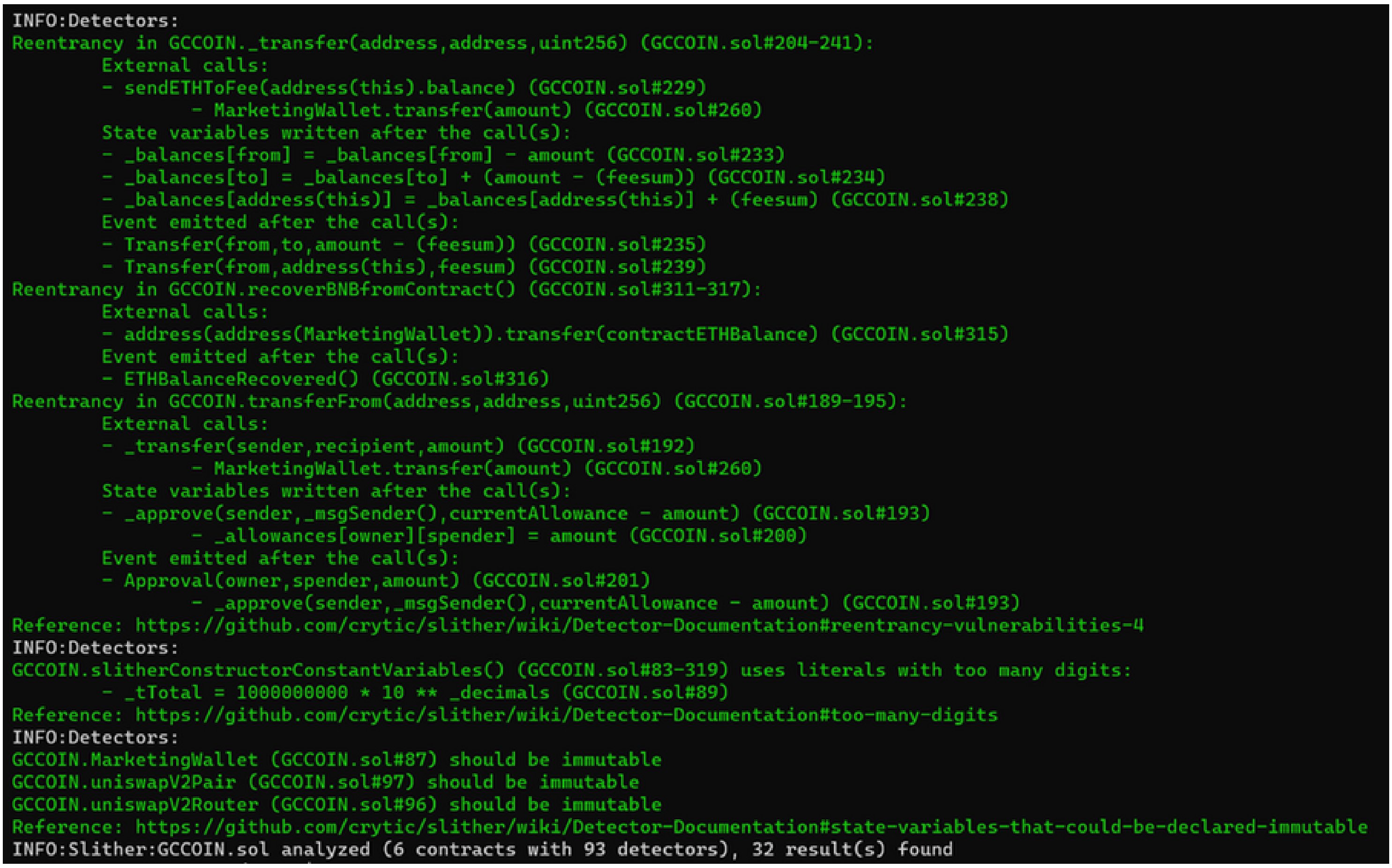
POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can Enable trading.
- The owner can set a whitelisted address.
- The owner can change buy/sell taxes not more than 5%.
- The owner can recover BEP20.

STATIC ANALYSIS



```
INFO:Detectors:
GCCOIN.allowance(address,address).owner (GCCOIN.sol#180) shadows:
        - Ownable.owner() (GCCOIN.sol#32-34) (function)
GCCOIN._approve(address,address,uint256).owner (GCCOIN.sol#197) shadows:
        - Ownable.owner() (GCCOIN.sol#32-34) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
GCCOIN.changeTaxes(uint256,uint256) (GCCOIN.sol#275-279) should emit an event for:
        - buyTaxes = newBuyFee (GCCOIN.sol#277)
        - sellTaxes = newSellFee (GCCOIN.sol#278)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Reentrancy in GCCOIN._transfer(address,address,uint256) (GCCOIN.sol#204-241):
        External calls:
        - swapTokensForEth(min(amount,min(contractTokenBalance,_maxSwapTokens))) (GCCOIN.sol#226)
                - unismapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (GCCOIN.sol#249-255)
        External calls sending eth:
        - sendETHToFee(address(this).balance) (GCCOIN.sol#229)
                - MarketingWallet.transfer(amount) (GCCOIN.sol#260)
        Event emitted after the call(s):
        - Transfer(from,to,amount - (feesum)) (GCCOIN.sol#235)
        - Transfer(from,address(this),feesum) (GCCOIN.sol#239)
Reentrancy in GCCOIN.recoverBEP20FromContract(address,uint256) (GCCOIN.sol#303-309):
        External calls:
        - IERC20(_tokenAddy).transfer(MarketingWallet,_amount) (GCCOIN.sol#307)
        Event emitted after the call(s):
        - ERC20TokensRecovered(_amount) (GCCOIN.sol#308)
Reentrancy in GCCOIN.transferFrom(address,address,uint256) (GCCOIN.sol#189-195):
        External calls:
        - _transfer(sender,recipient,amount) (GCCOIN.sol#192)
                - unismapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (GCCOIN.sol#249-255)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (GCCOIN.sol#192)
                - MarketingWallet.transfer(amount) (GCCOIN.sol#260)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (GCCOIN.sol#201)
                - _approve(sender,_msgSender(),currentAllowance - amount) (GCCOIN.sol#193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```



```
INFO:Detectors:
Reentrancy in GCCOIN._transfer(address,address,uint256) (GCCOIN.sol#204-241):
        External calls:
        - sendETHToFee(address(this).balance) (GCCOIN.sol#229)
                - MarketingWallet.transfer(amount) (GCCOIN.sol#260)
        State variables written after the call(s):
        - _balances[from] = _balances[from] - amount (GCCOIN.sol#233)
        - _balances[to] = _balances[to] + (amount - (feesum)) (GCCOIN.sol#234)
        - _balances[address(this)] = _balances[address(this)] + (feesum) (GCCOIN.sol#238)
        Event emitted after the call(s):
        - Transfer(from,to,amount - (feesum)) (GCCOIN.sol#235)
        - Transfer(from,address(this),feesum) (GCCOIN.sol#239)
Reentrancy in GCCOIN.recoverBNBfromContract() (GCCOIN.sol#311-317):
        External calls:
        - address(address(MarketingWallet)).transfer(contractETHBalance) (GCCOIN.sol#315)
        Event emitted after the call(s):
        - ETHBalanceRecovered() (GCCOIN.sol#316)
Reentrancy in GCCOIN.transferFrom(address,address,uint256) (GCCOIN.sol#189-195):
        External calls:
        - _transfer(sender,recipient,amount) (GCCOIN.sol#192)
                - MarketingWallet.transfer(amount) (GCCOIN.sol#260)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),currentAllowance - amount) (GCCOIN.sol#193)
                - _allowances[owner][spender] = amount (GCCOIN.sol#200)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (GCCOIN.sol#201)
                - _approve(sender,_msgSender(),currentAllowance - amount) (GCCOIN.sol#193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
GCCOIN.slitherConstructorConstantVariables() (GCCOIN.sol#83-319) uses literals with too many digits:
        - _tTotal = 1000000000 * 10 ** _decimals (GCCOIN.sol#89)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
GCCOIN.MarketingWallet (GCCOIN.sol#87) should be immutable
GCCOIN.uniswapV2Pair (GCCOIN.sol#97) should be immutable
GCCOIN.uniswapV2Router (GCCOIN.sol#96) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:GCCOIN.sol analyzed (6 contracts with 93 detectors), 32 result(s) found
```

FUNCTIONAL TESTING

1- **Approve** (passed):
https://testnet.bscscan.com/tx/0x9b5e8283f388f2100a0c6b801a59dfc7509a5d0f7d378c1ceb3fc495aaef6841

2- **Change Taxes** (passed):
https://testnet.bscscan.com/tx/0x3a84ba6c770b0652e20c5fa59e1374e9c29bf6d49cabde2b1cc0d6a0c9a8de04

3- **set Swap Back Settings** (passed):
https://testnet.bscscan.com/tx/0xea143b57d50aa04c4e68fa969e78c9d393decb5bc73974951e876fb8655970b6

4- **Enable Trading** (passed):
https://testnet.bscscan.com/tx/0xc8f7954a83259d5e20454619de4aab395ebdacc96738d347381912537e730ed1

5- **Set Swap Token** (passed):
https://testnet.bscscan.com/tx/0x9e1de7bbbb8a16d73376d5fbe1ba4c5080f472e5af98162658202a0a4e06ae6c

MANUAL TESTING

Centralization – Enabling Trades.

Severity: High

Function: EnableTrading

Status: Open

Overview:

The EnableTrading 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 enabledTrading() external onlyOwner() {
require(!tradeEnable,"trading is already open");
    _SwapBackEnable = true;
     tradeEnable = true;
    emit TradingOpenUpdated();
  }
```

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 give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

MANUAL TESTING

Centralization – Missing Events

Severity: Low

Function: Missing Events

Status: Open

Overview:

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function changeTaxes(uint256 newBuyFee, uint256 newSellFee)
external onlyOwner {
require(newBuyFee <= 5 && newSellFee <= 5, "ERC20: wrong tax
value!");
    buyTaxes = newBuyFee;
    sellTaxes = newSellFee;
  }
```

MANUAL TESTING

Centralization – Local Variable Shadowing

Severity: Low

Status: Open

Function: _approve and allowance

Overview:

```
function allowance(address owner, address
spender) public view override returns (uint256) {
return _allowances[owner][spender];
  }
function _approve(address owner, address
spender, uint256 amount) private {
require(owner != address(0), "ERC20: approve
from the zero address");
require(spender != address(0), "ERC20: approve
to the zero address");
    _allowances[owner][spender] = amount;
emit Approval(owner, spender, amount);
  }
```

Suggestion:

Rename the local variable that shadows another component.

MANUAL TESTING

Optimization

Severity: Optimization

Subject: Remove unused code.

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice. though to avoid them.

```
event FeesUpdated(uint256 indexed
_feeAmount);
```

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