AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Grok X Grok

DATED : 18 Dec 23'

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 enableTrading() external onlyOwner{
  require(tradingEnabled == false, "Trading is already enabled");
  tradingEnabled = true;
}
```

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.

AUDIT SUMMARY

Project name – Grok X Grok

Date: 18 Dec, 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	1	2
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/0xb5f2f7d4c97959456d71f5c6cf3f9fa02d5c5ccd#code

Token Information

Token Address:
0xE2C14977f8012357a82Fa1944D509B2a21d2F399

Name: Grok X Grok

Symbol: Grok X Grok

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner: 0x1bc3b6Fed224b2050052a739acA259Da02fc75E0

Deployer: 0x1bc3b6Fed224b2050052a739acA259Da02fc75E0

Total Supply: 1,000,000

Checksum: 0e1c3a4fbb6e83e8393a57617b5a5b53

Testnet:
https://testnet.bscscan.com/address/0xb5f2f7d4c97959456d71f5c6cf3f9fa02d5c5ccd#code

TOKEN OVERVIEW

Buy Fee: 0-0%

Sell Fee: 0-0%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: Yes

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.

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

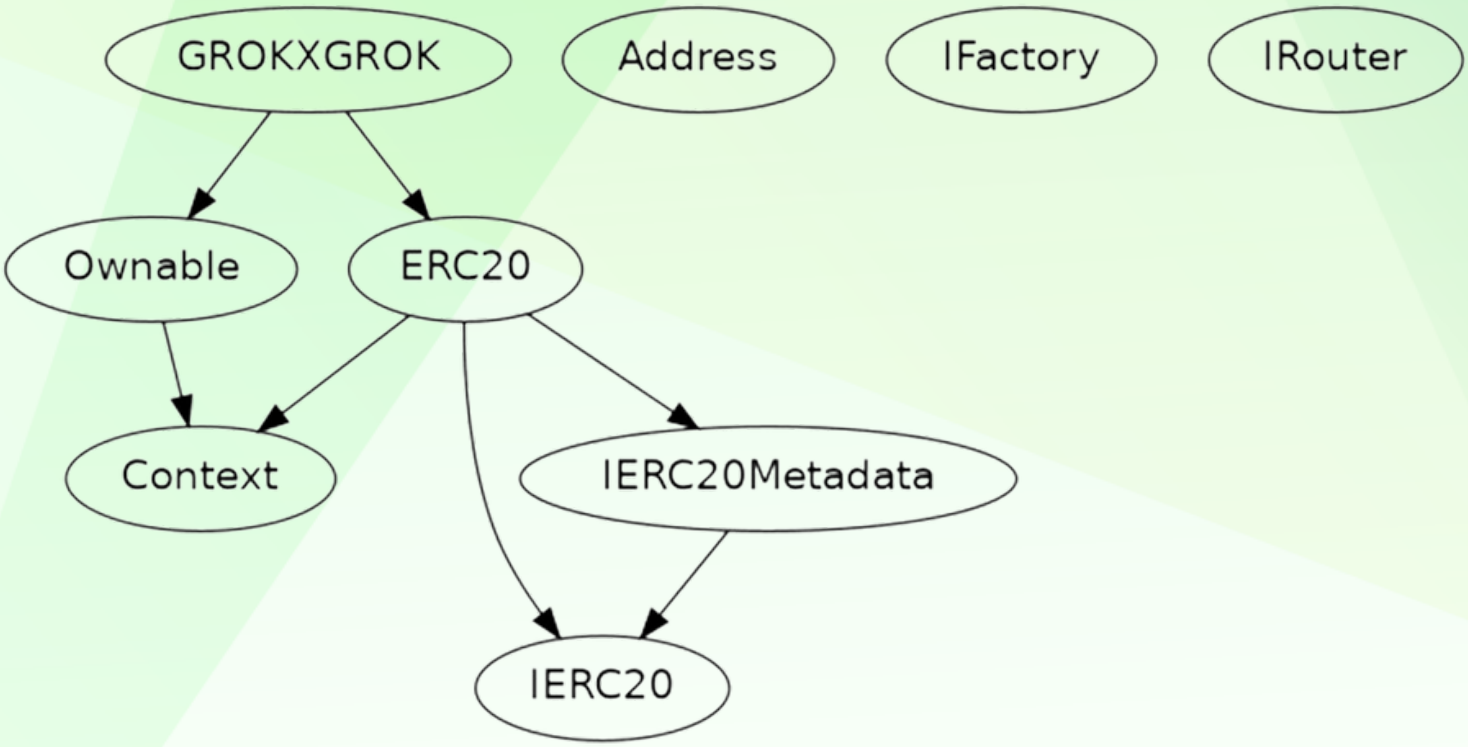
CLASSIFICATION OF RISK

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	1
Gas Optimization / Suggestions	2

INHERITANCE TREE



POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can rescue bnb.
- The owner can enable/disable trading.
- The owner can update dead line.
- The owner can market the wallet address.
- The owner can update the exempt fee

STATIC ANALYSIS

```
INFO:Detectors:
GROKXGROK.Liquify(uint256,GROKXGROK.Taxes) (GROKXGROK.sol#513-547) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - swapTaxes.liquidity) (GROKXGROK.sol#534)
        - ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity (GROKXGROK.sol#535)
GROKXGROK.Liquify(uint256,GROKXGROK.Taxes) (GROKXGROK.sol#513-547) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - swapTaxes.liquidity) (GROKXGROK.sol#534)
        - mrkettxAmt = unitBalance * 2 * swapTaxes.mrkettx (GROKXGROK.sol#542)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
GROKXGROK._transfer(address,address,uint256).currentTaxes (GROKXGROK.sol#471) is a local variable never initialized
GROKXGROK._transfer(address,address,uint256).feesum (GROKXGROK.sol#469) is a local variable never initialized
GROKXGROK._transfer(address,address,uint256).feeswap (GROKXGROK.sol#468) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
GROKXGROK.addLiquidity(uint256,uint256) (GROKXGROK.sol#567-580) ignores return value by router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GROKXGROK.sol#572-579
)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
GROKXGROK._transfer(address,address,uint256).fee (GROKXGROK.sol#470) is written in both
        fee = 0 (GROKXGROK.sol#479)
        fee = (amount * feesum) / 100 (GROKXGROK.sol#495)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write
INFO:Detectors:
GROKXGROK.updateLiquidityTreshhold(uint256) (GROKXGROK.sol#586-589) should emit an event for:
        - tokenLiquidityThreshold = new_amount * 10 ** decimals() (GROKXGROK.sol#588)
GROKXGROK.updatedeadline(uint256) (GROKXGROK.sol#598-602) should emit an event for:
        - deadline = _deadline (GROKXGROK.sol#601)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Modifier GROKXGROK.lockTheSwap() (GROKXGROK.sol#373-379) does not always execute _; or revertReference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-modifier
INFO:Detectors:
Reentrancy in GROKXGROK.Liquify(uint256,GROKXGROK.Taxes) (GROKXGROK.sol#513-547):
        External calls:
        - swapTokensForETH(toSwap) (GROKXGROK.sol#531)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (GROKXGROK.sol#558-564)
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (GROKXGROK.sol#539)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GROKXGROK.sol#572-579)
        External calls sending eth:
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (GROKXGROK.sol#539)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GROKXGROK.sol#572-579)
        State variables written after the call(s):
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (GROKXGROK.sol#539)
                - _allowances[owner][spender] = amount (GROKXGROK.sol#248)
Reentrancy in GROKXGROK.transferFrom(address,address,uint256) (GROKXGROK.sol#406-421):
        External calls:
        - _transfer(sender,recipient,amount) (GROKXGROK.sol#411)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GROKXGROK.sol#572-579)
                - (success) = recipient.call{value: amount}() (GROKXGROK.sol#260)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (GROKXGROK.sol#558-564)
                - address(mrkettxWallet).sendValue(mrkettxAmt) (GROKXGROK.sol#544)
```

```
INFO:Detectors:
Reentrancy in GROKXGROK.Liquify(uint256,GROKXGROK.Taxes) (GROKXGROK.sol#513-547):
        External calls:
        - swapTokensForETH(toSwap) (GROKXGROK.sol#531)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (GROKXGROK.sol#558-564)
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (GROKXGROK.sol#539)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GROKXGROK.sol#572-579)
        External calls sending eth:
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (GROKXGROK.sol#539)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GROKXGROK.sol#572-579)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (GROKXGROK.sol#249)
                - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (GROKXGROK.sol#539)
Reentrancy in GROKXGROK._transfer(address,address,uint256) (GROKXGROK.sol#457-511):
        External calls:
        - Liquify(feeswap,currentTaxes) (GROKXGROK.sol#500)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GROKXGROK.sol#572-579)
                - (success) = recipient.call{value: amount}() (GROKXGROK.sol#260)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (GROKXGROK.sol#558-564)
                - address(mrkettxWallet).sendValue(mrkettxAmt) (GROKXGROK.sol#544)
        External calls sending eth:
        - Liquify(feeswap,currentTaxes) (GROKXGROK.sol#500)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GROKXGROK.sol#572-579)
                - (success) = recipient.call{value: amount}() (GROKXGROK.sol#260)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (GROKXGROK.sol#222)
                - super._transfer(sender,recipient,amount - fee) (GROKXGROK.sol#503)
        - Transfer(sender,recipient,amount) (GROKXGROK.sol#222)
                - super._transfer(sender,address(this),feeAmount) (GROKXGROK.sol#508)
Reentrancy in GROKXGROK.transferFrom(address,address,uint256) (GROKXGROK.sol#406-421):
        External calls:
        - _transfer(sender,recipient,amount) (GROKXGROK.sol#411)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GROKXGROK.sol#572-579)
                - (success) = recipient.call{value: amount}() (GROKXGROK.sol#260)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (GROKXGROK.sol#558-564)
                - address(mrkettxWallet).sendValue(mrkettxAmt) (GROKXGROK.sol#544)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (GROKXGROK.sol#411)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GROKXGROK.sol#572-579)
                - (success) = recipient.call{value: amount}() (GROKXGROK.sol#260)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (GROKXGROK.sol#249)
                - _approve(sender,_msgSender(),currentAllowance - amount) (GROKXGROK.sol#418)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (GROKXGROK.sol#15-18) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.19 (GROKXGROK.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
```

STATIC ANALYSIS

```
INFO:Detectors:
Context._msgData() (GROKXGROK.sol#15-18) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.19 (GROKXGROK.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (GROKXGROK.sol#254-265):
        - (success) = recipient.call{value: amount}() (GROKXGROK.sol#260)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IRouter.WETH() (GROKXGROK.sol#317) is not in mixedCase
Function GROKXGROK.Liquify(uint256,GROKXGROK.Taxes) (GROKXGROK.sol#513-547) is not in mixedCase
Parameter GROKXGROK.updateLiquidityTreshhold(uint256).new_amount (GROKXGROK.sol#586) is not in mixedCase
Function GROKXGROK.EnableTrading() (GROKXGROK.sol#591-596) is not in mixedCase
Parameter GROKXGROK.updatedeadline(uint256)._deadline (GROKXGROK.sol#598) is not in mixedCase
Parameter GROKXGROK.updateExemptFee(address,bool)._address (GROKXGROK.sol#609) is not in mixedCase
Variable GROKXGROK.genesis_block (GROKXGROK.sol#356) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (GROKXGROK.sol#16)" inContext (GROKXGROK.sol#10-19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
GROKXGROK.launchtax (GROKXGROK.sol#358) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
GROKXGROK.pair (GROKXGROK.sol#348) should be immutable
GROKXGROK.router (GROKXGROK.sol#347) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:GROKXGROK.sol analyzed (9 contracts with 93 detectors), 32 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/0x3297946658dba0e9b2797f8518f78a7a369ea22e87deb5f940976d4bce87f9ed

2- Increase Allowance (passed):

https://testnet.bscscan.com/tx/0x092fd2a779c561253e05ecba6ceaa51fa9542d640959ba92e8e5ab7a0f699c08

3- Decrease Allowance (passed):

https://testnet.bscscan.com/tx/0xbe690b8ba9d14e42cfe387c513abd3aab798aa25a5c7cb0ea63eec2dc3551359

4- Enable Trading (passed):

https://testnet.bscscan.com/tx/0x04b901c150ce3544078a8876408c7cfadd411f13793d046a827a7a346f969055

5- Update Exempt Fee (passed):

https://testnet.bscscan.com/tx/0xabaf74f5ceffef3c13c9d08b5d5219029356bba1e4d3ba8859e4ddb16f41411c

6- Update Liquidity Provide (passed):

https://testnet.bscscan.com/tx/0x1d4b891837427526263 8fa1b0f820b362879cdd712219f7b35ecae2e143e49c1

7- Update Liquidity Treshhold (passed):

https://testnet.bscscan.com/tx/0xbcfa33f608abc703da417415d51948ffebe8b1afb4e6c776c604f5941cafe31e

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 enableTrading() external onlyOwner{
  require(tradingEnabled == false, "Trading is already enabled");
  tradingEnabled = true;
}
```

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.

MANUAL TESTING

Centralization – Missing Events

Severity: Low

subject: 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 updateLiquidityProvide(bool state) external onlyOwner {
    providingLiquidity = state;
  }
 function updatedeadline(uint256 _deadline) external onlyOwner {
    require(!tradingEnabled, "Can't change when trading has started");
    require(_deadline < 5, "Deadline should be less than 5 Blocks");
    deadline = _deadline;
  }
 function updateMktWallet(address newWallet) external onlyOwner {
    require(newWallet != address(0), "Fee Address cannot be zero address");
    mrkettxWallet = newWallet;
  }
function updateExemptFee(address _address, bool state) external onlyOwner {
    exemptFee[_address] = state;
  }
function bulkExemptFee(address[] memory accounts, bool state) external
onlyOwner {
    for (uint256 i = 0; i < accounts.length; i++) {
      exemptFee[accounts[i]] = state;
    }
  }
```

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

```
  function _msgData() internal view virtual returns (bytes calldata) {
    this; // silence state mutability warning without generating bytecode - see https://github.com/ethereum/solidity/issues/2691
    return msg.data;
  }
}
```

MANUAL TESTING

Optimization

Severity: Informational

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 ^0.8.19;

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.

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