auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Kirby Inu

DATED : 18 Apr 23'

AUDIT SUMMARY

Project name – Kirby Inu

Date: 18 April, 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	2	0	0	2
Acknowledged	0	0	0	0	0
Resolved	0	1	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 done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0x881F7c4e0cBAcac94A4bC0555525c665A5669a2C

Token Information

Token Name : Kirby Inu

Token Symbol: Kinu

Decimals: 18

Token Supply: 42,000,000,000,000

Token Address:
0xEa81224464c3173a38586157aB866eE63216e1Da

Checksum:
34d3a3e8f0b23d06aed78dcef65cd84ae4f0b5c6

Owner:
0x8a94cA37223Fae96a3696B48F4F55Af0C40B4e04

Deployer:
0x41B7b487B9F6726e114f9F4aFa9Fa7217219F805

TOKEN OVERVIEW

Fees:

Buy Fees: up to 10%

Sell Fees: up to 10%

Transfer Fees: 0%

Fees Privilige: none

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: Yes

Other Priviliges: excluding from fees - including in fees

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

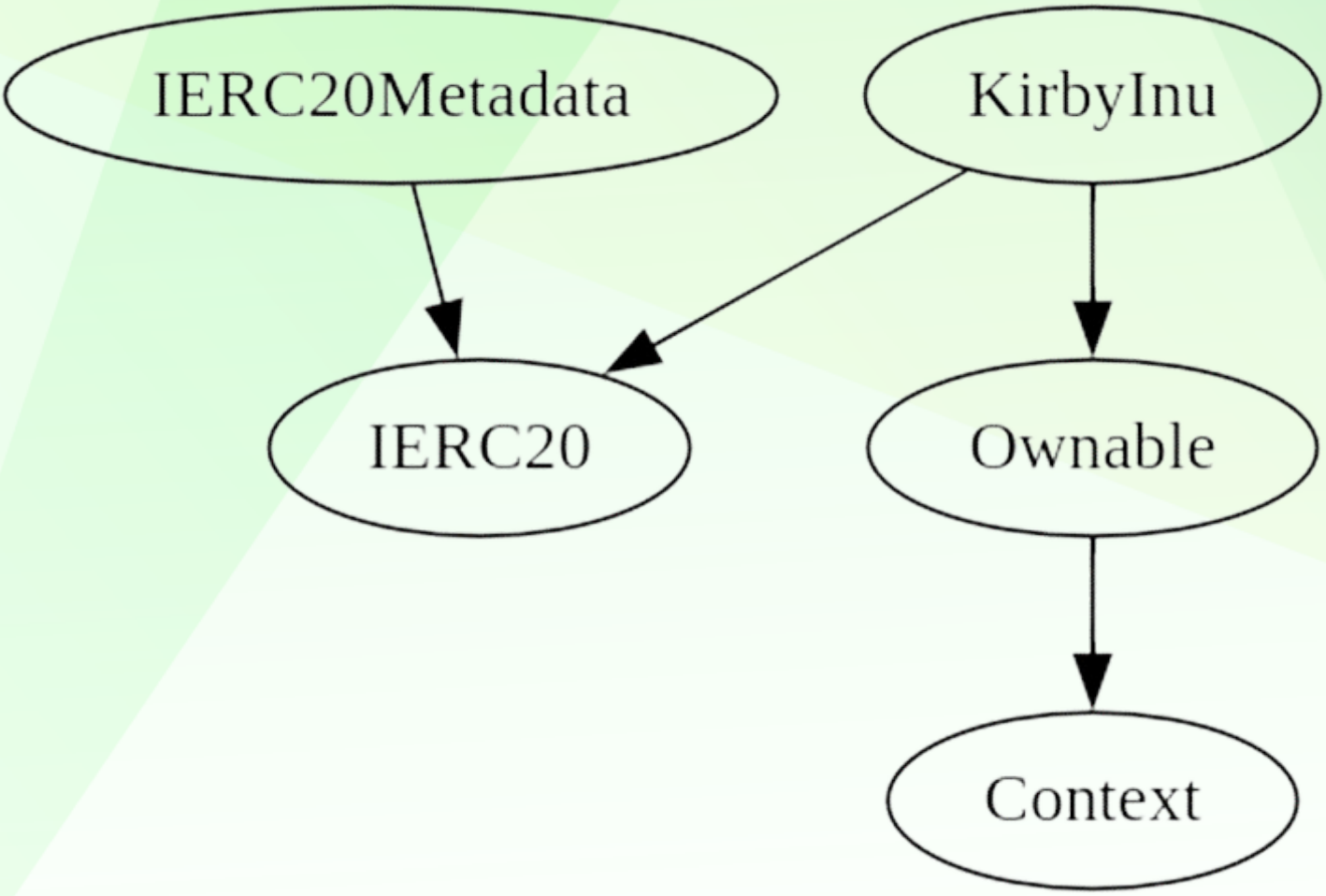
CLASSIFICATION OF RISK

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 (1 resolved)
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	2

INHERITANCE TREE



POINTS TO NOTE

- Owner is change fees (10% buy, 10% sell, 0% transfers)
- 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 disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
ISwapRouter	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	addLiquidity	External ❗	🛑	NO❗
ISwapFactory	Interface			
└	createPair	External ❗	🛑	NO❗
Ownable	Implementation			
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	transferOwnership	Public ❗	🛑	onlyOwner
TokenDistributor	Implementation			
└	<Constructor>	Public ❗	🛑	NO❗
KirbyInu	Implementation	IERC20, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗

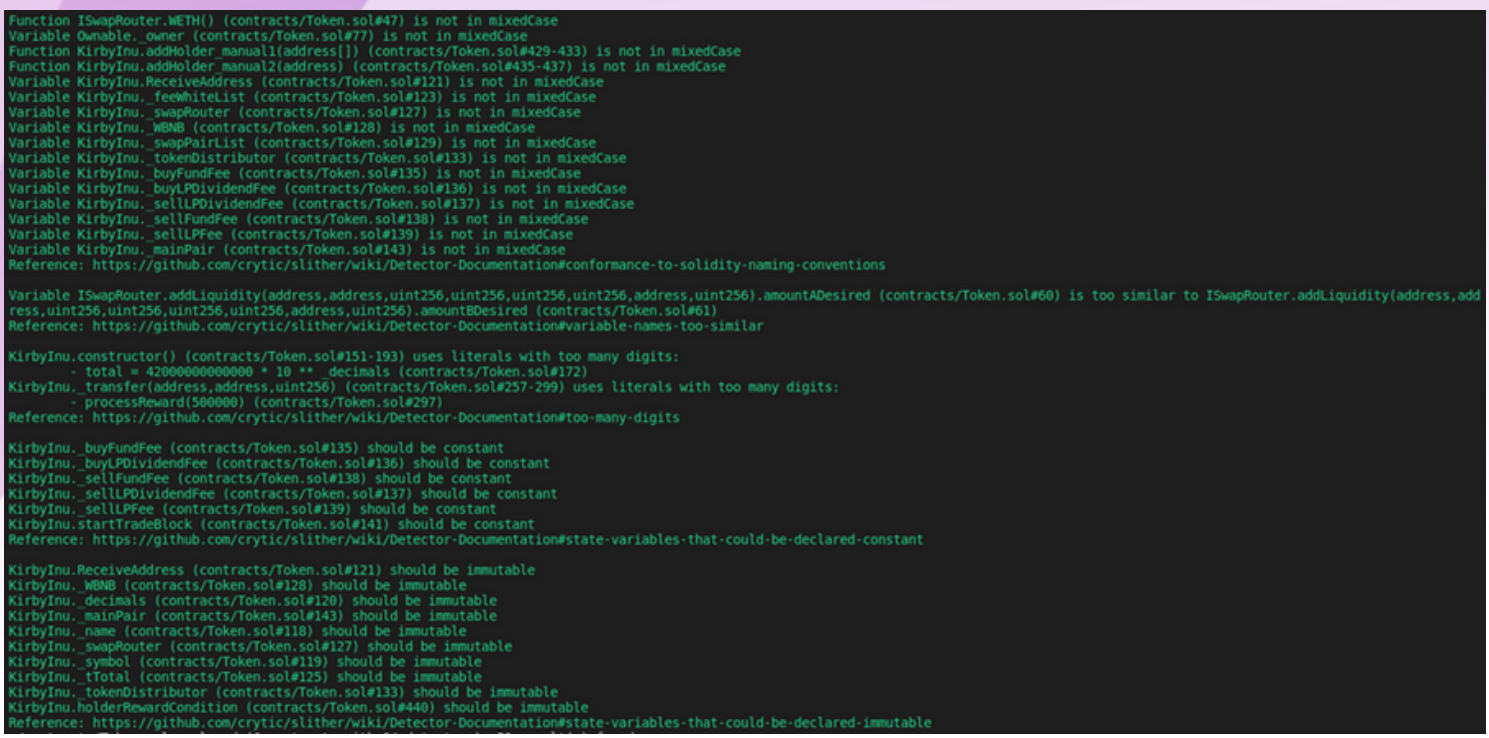
CONTRACT ASSESMENT

└	transferFrom	Public ❗	🛑		NO❗
└	_approve	Private 🔐	🛑		
└	_transfer	Private 🔐	🛑		
└	_tokenTransfer	Private 🔐	🛑		
└	swapTokenForFund	Private 🔐	🛑	lockTheSwap
└	_takeTransfer	Private 🔐	🛑		
└	setFundAddress	External ❗	🛑	onlyFunder
└	setFeeWhiteList	External ❗	🛑	onlyFunder
└	setSwapPairList	External ❗	🛑	onlyFunder
└	claimBalance	External ❗	🛑		NO❗
└	<Receive Ether>	External ❗	💵	NO❗
└	addHolder	Private 🔐	🛑		
└	addHolder_manual1	Public ❗	🛑	onlyFunder
└	addHolder_manual2	Public ❗	🛑	onlyFunder
└	processReward	Private 🔐	🛑		
DividendPayingTokenInterface	Interface			
└	dividendOf	External ❗		NO❗
└	distributeDividends	External ❗	💵	NO❗
└	withdrawableDividendOf	External ❗		NO❗
└	withdrawnDividendOf	External ❗		NO❗
└	accumulativeDividendOf	External ❗		NO❗
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❗
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner

CONTRACT ASSESMENT

Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
Function ISwapRouter.WETH() (contracts/Token.sol#47) is not in mixedCase
Variable Ownable._owner (contracts/Token.sol#77) is not in mixedCase
Function KirbyInu.addHolder_manual1(address[]) (contracts/Token.sol#429-433) is not in mixedCase
Function KirbyInu.addHolder_manual2(address) (contracts/Token.sol#435-437) is not in mixedCase
Variable KirbyInu.ReceiveAddress (contracts/Token.sol#121) is not in mixedCase
Variable KirbyInu._feeWhiteList (contracts/Token.sol#123) is not in mixedCase
Variable KirbyInu._swapRouter (contracts/Token.sol#127) is not in mixedCase
Variable KirbyInu._WBNB (contracts/Token.sol#128) is not in mixedCase
Variable KirbyInu._swapPairList (contracts/Token.sol#129) is not in mixedCase
Variable KirbyInu._tokenDistributor (contracts/Token.sol#133) is not in mixedCase
Variable KirbyInu._buyFundFee (contracts/Token.sol#135) is not in mixedCase
Variable KirbyInu._buyLPDividendFee (contracts/Token.sol#136) is not in mixedCase
Variable KirbyInu._sellLPDividendFee (contracts/Token.sol#137) is not in mixedCase
Variable KirbyInu._sellFundFee (contracts/Token.sol#138) is not in mixedCase
Variable KirbyInu._sellLPFee (contracts/Token.sol#139) is not in mixedCase
Variable KirbyInu._mainPair (contracts/Token.sol#143) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable ISwapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#60) is too similar to ISwapRouter.addLiquidity(address,add
ress,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#61)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

KirbyInu.constructor() (contracts/Token.sol#151-193) uses literals with too many digits:
        - total = 42000000000000 * 10 ** _decimals (contracts/Token.sol#172)
KirbyInu._transfer(address,address,uint256) (contracts/Token.sol#257-299) uses literals with too many digits:
        - processReward(500000) (contracts/Token.sol#297)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

KirbyInu._buyFundFee (contracts/Token.sol#135) should be constant
KirbyInu._buyLPDividendFee (contracts/Token.sol#136) should be constant
KirbyInu._sellFundFee (contracts/Token.sol#138) should be constant
KirbyInu._sellLPDividendFee (contracts/Token.sol#137) should be constant
KirbyInu._sellLPFee (contracts/Token.sol#139) should be constant
KirbyInu.startTradeBlock (contracts/Token.sol#141) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

KirbyInu.ReceiveAddress (contracts/Token.sol#121) should be immutable
KirbyInu._WBNB (contracts/Token.sol#128) should be immutable
KirbyInu._decimals (contracts/Token.sol#120) should be immutable
KirbyInu._mainPair (contracts/Token.sol#143) should be immutable
KirbyInu._name (contracts/Token.sol#118) should be immutable
KirbyInu._swapRouter (contracts/Token.sol#127) should be immutable
KirbyInu._symbol (contracts/Token.sol#119) should be immutable
KirbyInu._tTotal (contracts/Token.sol#125) should be immutable
KirbyInu._tokenDistributor (contracts/Token.sol#133) should be immutable
KirbyInu.holderRewardCondition (contracts/Token.sol#440) 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

Some features like auto-liquidity are disabled forever since taxes can not be changed, current tests are done to be compatible with contract at address 0xEa81224464c3173a38586157aB866eE63216e1Da however other tests for auto-liquidity and distribution also been done in a local testing environment (not included here)

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0xa615f73b0549e69fad27e81acdd5b68fb7204168ca17364bcb16c3d7c73747a3

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x23b5a5b6a9c6a8afc453a3bc84149a9cd62101ba6ec01c9fe30df32838799d49

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xd8e2e835edc6661efa4b3c35376791c1be73d3a00bb110c36b3652bb5c538cfc

4- Transferring when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x30051d67f55445adbd40ce5056730c4f6d2fb3c5548bf2dafb528baed3a6f032

5- Buying when not excluded (upto 10% tax) (passed):
https://testnet.bscscan.com/tx/0xa58f34b8d0f183949732d35354eae703614fbd51438573f6b65563b4791b9027

6- Selling when not excluded (upto 10% tax) (passed):
https://testnet.bscscan.com/tx/0x78b8d274825bea49d490d6490012d36493372b20eeab7e29484d5a91226eda72

FUNCTIONAL TESTING

7- Transferring when not excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x13a6f34e633a4a4e4d5f0c9af8123910816ffc23f9360dab104f8fc1eba54440

8- Internal swap (passed):
Fund wallet receiving BUSD (USDT in mainnet)
https://testnet.bscscan.com/address/0x8a94ca37223fae96a3696b48f4f55af0c40b4e04#tokentxns

MANUAL TESTING

Logical - Lack of allowance

Severity: High
Function: swapTokenForFund
Lines: 286
Status: Resolved
Overview:
Contract must approve router to spend its USDT tokens in order to be able to add liquidity.

```
if (lpAmount > 0) {
    uint256 lpWbnb = (WbnbBalance * lpFee) / swapFee;
    if (lpWbnb > 0) {
        _swapRouter.addLiquidity(
            address(this),
            _WBNB,
            lpAmount,
            lpWbnb,
            0,
            0,
            fundAddress,
            block.timestamp
        );
    }
}
```

Recommendation:
approve router to spend USDT tokens of the contract

```
WBNB.approve(address(_swapRouter), ~uint256(0))
```

Allevation:
Since LP tax is 0, this section of the code is disabled.

MANUAL TESTING

Logical - Dividend tracker

Severity: High
Function: processReward
Lines: 385
Status: Not Resolved
Overview:
There are several issuse in dividend tracker:

1- Wallet that are excluded from wallets still affect the calculations of each wallet's rewards

```
if (tokenBalance > 0 && !excludeHolder[shareHolder]) {
    amount = (balance * tokenBalance) / holdTokenTotal;

uint holdTokenTotal = holdToken.totalSupply(); // total shares
```

if a wallet is excluded from rewards, its balance must be deducted from total shares, otherwise a portion of reward tokens will be lost.
Recommendation:
you can create a variable to store total shares which is initialy equal to total supply of the holding token. If a wallet is excluded, the balance of the wallet gets deducted from total shares.

2- Hold token might be assigned to an invalid address:

```
IERC20 holdToken = IERC20(_mainPair);
uint holdTokenTotal = holdToken.totalSupply();
```

at current implementation of the code, LP tokens are used as holding token.
Recommendation:
if this implementation is not intentional, make sure to change the holdToken.

Team Explanation:
The handling fee will only be used for marketing repurchase

MANUAL TESTING

Informational - Withdrawing tokens

Status: Not Resolved
Overview:
Current implementation of the contract doesn't have a function to withdraw stuck ERC20 tokens.
Recommendation:
if this implementation is not intentional, make sure to change the holdToken.

```
function withdrawTokens(address _token) public onlyOwner {
        IERC20(_token).transfer(owner(), IERC20(_token).balanceOf(Address(this));
}
```

Informational - Naming convention

Status: Not Resolved
Overview:
USDT token is declared as **WBNB** in the contract, this can cause confusion for investors.
Recommendation:
Change **WBNB** to **USDT**

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