Cyberscope

Audit Report

Eagle AI

May 2024

Network BASE

Address 0x42c1B8044Ff849d37782566fe7fc9C1E04Ec648f

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Unresolved
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	MEM	Misleading Error Messages	Unresolved
	MEE	Missing Events Emission	Unresolved
	MTEE	Missing Transfer Event Emission	Unresolved
	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
	L02	State Variables could be Declared Constant	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L05	Unused State Variable	Unresolved

Table of Contents

Review

Contract Name	EAGLEAI
Compiler Version	v0.8.24+commit.e11b9ed9
Optimization	200 runs
Explorer	https://basescan.org/address/0x42c1B8044Ff849d37782566fe7fc9C1E04Ec648f
Address	0x42c1B8044Ff849d37782566fe7fc9C1E04Ec648f
Network	BASE
Symbol	EAI
Decimals	18
Total Supply	100,000,000
Badge Eligibility	Must Fix Criticals

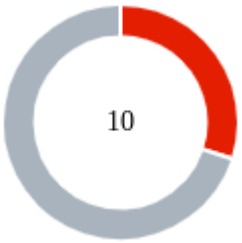
Audit Updates

Initial Audit	15 May 2024

Source Files

Filename	SHA256
EAGLEAI.sol	748fb13f342dbdbf407feb4d1013ee678951fb96147e44550bfc247429074a5f

Findings Breakdown



Critical	3
Medium	0
Minor / Informative	7

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	3	0	0	0
Medium	0	0	0	0
Minor / Informative	7	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	EAGLEAI.sol#L1074
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enable the owner will not be able to disable them again.

```
    function startTrading() external onlyOwner {
        tradeEnabled = true;
        emit TradeEnabled(tradeEnabled);
    }
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	EAGLEAI.sol#L1111,1131
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `updateBuyTaxPer` or `updateSellTaxPer` function with a high percentage value.

```
   function updateBuyTaxPer(uint256 reflectionPercent,uint256
coinOperartionPer,uint256 liquidityTaxPer,uint256 burnTaxPer) external
onlyOwner {
        uint256 totalTax = reflectionPercent + coinOperartionPer +
liquidityTaxPer + burnTaxPer;
        require(totalTax <= 100,"You can not set buy tax more then
100%");
        buyReflectionTax = reflectionPercent;
        buyCoinWalletTaxPer = coinOperartionPer;
        buyLiquidityTaxPer = liquidityTaxPer;
        buyBurnTaxPer = burnTaxPer;
        emit
BuyTaxUpdated(buyReflectionTax,buyCoinWalletTaxPer,buyLiquidityTaxPer,buy
BurnTaxPer);
    }

    function updateSellTaxPer(uint256 reflectionPercent,uint256
coinOperartionPer,uint256 liquidityTaxPer,uint256 burnTaxPer) external
onlyOwner {
        uint256 totalTax = reflectionPercent + coinOperartionPer +
liquidityTaxPer + burnTaxPer;
        require(totalTax <= 100,"You can not set sell tax more then
100%");
        sellReflectionTax = reflectionPercent;
        SellCoinWalletTaxPer = coinOperartionPer;
        sellLiquidityTaxPer = liquidityTaxPer;
        sellBurnTaxPer = burnTaxPer;
        emit
SellTaxUpdated(sellReflectionTax,SellCoinWalletTaxPer,sellLiquidityTaxPer
,sellBurnTaxPer);
    }
```

Recommendation

The contract could embody a check for the maximum acceptable value. The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Critical
Location	EAGLEAI.sol#L1084
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `addBlacklist` function.

```
    function addBlacklist(address account) external  onlyOwner {
        require(!blacklisted[account],"Account already
blacklisted");
        blacklisted[account] = true;
        emit AddedInBlacklist(account);
    }
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

MEM - Missing Error Messages

Criticality	Minor / Informative
Location	EAGLEAI.sol#L1173
Status	Unresolved

Description

The contract is missing error messages. These are no error messages to accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(tradeEnabled)
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	EAGLEAI.sol#L733,745
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
    function excludeFromFee(address account) external onlyOwner
{
        require(!_isExcludedFromFee[account],"Alreay excluded
from fee");
        _isExcludedFromFee[account] = true;
    }

    function includeInFee(address account) external onlyOwner {
        require(_isExcludedFromFee[account],"Alreay included in
fee");
        _isExcludedFromFee[account] = false;
    }
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

MTEE - Missing Transfer Event Emission

Criticality	Minor / Informative
Location	EAGLEAI.sol#L837
Status	Unresolved

Description

The contract does not emit an event when portions of the main amount are transferred during the transfer process. This lack of event emission results in decreased transparency and traceability regarding the flow of tokens, and hinders the ability of decentralized applications (dApps), such as blockchain explorers, to accurately track and analyze these transactions.

```
      function _takeCoinFund(uint256 tCoinOperation) private {
           uint256 currentRate =  _getRate();
         uint256 rCoinOperation = tCoinOperation * currentRate;
         _rOwned[fundWallet] = _rOwned[fundWallet] +
rCoinOperation;
         if(_isExcluded[fundWallet])
             _tOwned[fundWallet] = _tOwned[fundWallet] +
tCoinOperation;
         emit CoinFund(tCoinOperation);
     }
```

Recommendation

It is advisable to incorporate the emission of detailed event logs following each asset transfer. These logs should encapsulate key transaction details, including the identities of the sender and receiver, and the quantity of assets transferred. Implementing this practice will enhance the reliability and transparency of transaction tracking systems, ensuring accurate data availability for ecosystem participants.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	EAGLEAI.sol#L1301
Status	Unresolved

Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
_approve(address(this), address(uniswapV2Router), tokenAmount);

// make the swap
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTok
ens(
    tokenAmount,
    0, // accept any amount of ETH
    path,
    address(this),
    block.timestamp
);
```

Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	EAGLEAI.sol#L372
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint256 private startingHr;
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	EAGLEAI.sol#L217,396,760,772,788,988,999,1011,1024
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
function WETH() external pure returns (address);
uint256 public SellCoinWalletTaxPer=2;
function setFundWallet(address _fundWallet)
function setSwapAndLiquifyEnabled(bool _enabled)
...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L05 - Unused State Variable

Criticality	Minor / Informative
Location	EAGLEAI.sol#L372
Status	Unresolved

Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
uint256 private startingHr;
```

Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality, and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

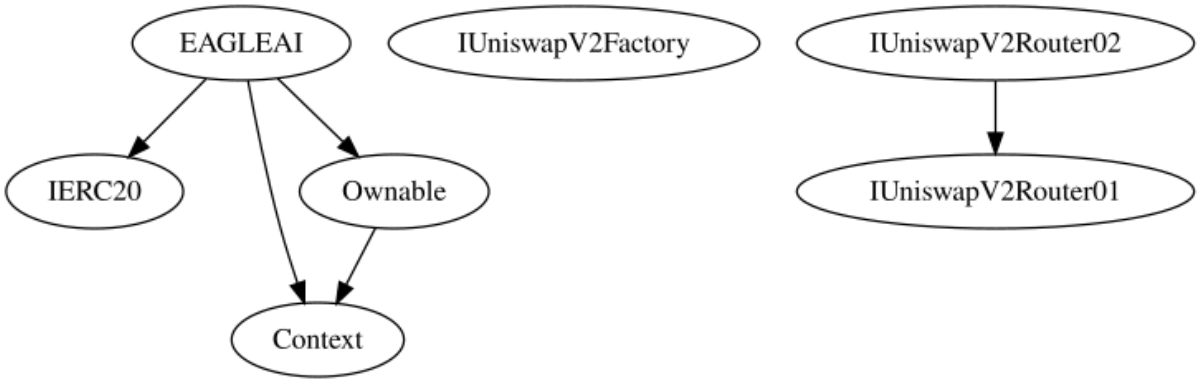
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
EAGLEAI	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	Public		-
	transfer	External	✓	-
	allowance	External		-
	approve	Public	✓	-
	transferFrom	External	✓	-
	increaseAllowance	External	✓	-
	decreaseAllowance	External	✓	-
	isExcludedFromReward	External		-
	totalFees	External		-
	deliver	External	✓	-
	reflectionFromToken	External		-
	tokenFromReflection	Public		-
	excludeFromReward	External	✓	onlyOwner

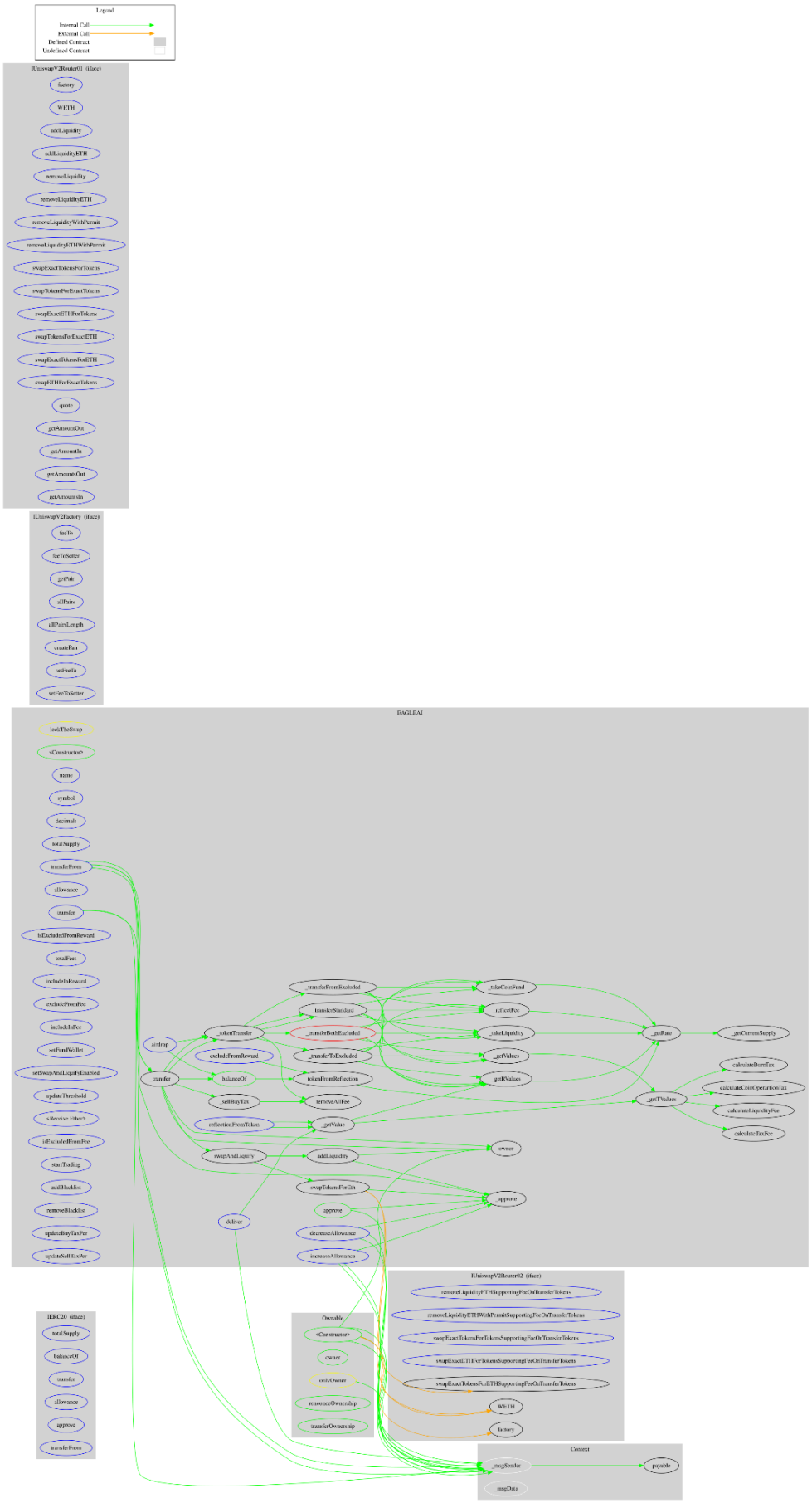
	includeInReward	External	✓	onlyOwner
	_transferBothExcluded	Private	✓	
	excludeFromFee	External	✓	onlyOwner
	includeInFee	External	✓	onlyOwner
	setFundWallet	External	✓	onlyOwner
	setSwapAndLiquifyEnabled	External	✓	onlyOwner
	updateThreshold	External	✓	onlyOwner
		External	Payable	-
	_reflectFee	Private	✓	
	_takeCoinFund	Private	✓	
	_getValues	Private		
	_getValue	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidity	Private	✓	
	calculateTaxFee	Private		
	calculateLiquidityFee	Private		
	calculateCoinOperartionTax	Private		
	calculateBurnTax	Private		
	removeAllFee	Private	✓	
	isExcludedFromFee	External		-
	_approve	Private	✓	
	startTrading	External	✓	onlyOwner

	addBlacklist	External	✓	onlyOwner
	removeBlacklist	External	✓	onlyOwner
	updateBuyTaxPer	External	✓	onlyOwner
	updateSellTaxPer	External	✓	onlyOwner
	_transfer	Private	✓	
	airdrop	External	✓	onlyOwner
	_sellBuyTax	Private	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Router01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
swapTokensForExactTokens
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn
IUniswapV2Factory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
EAGLEAI
lockTheSwap
<Constructor>
name
symbol
decimals
totalSupply
transferFrom
allowance
transfer
isExcludedFromReward
totalFees
includeInReward
excludeFromFee
includeInFee
setFundWallet
setSwapAndLiquifyEnabled
updateThreshold
<Receive Ether>
isExcludedFromFee
startTrading
addBlackList
removeBlackList
updateBuyTaxPer
updateSellTaxPer
_transferFromExcluded
_transferStandard
_transferBothExcluded
_transferToExcluded
_tokenTransfer
airdrop
excludeFromReward
_transfer
balanceOf
sellBuyTax
reflectionFromToken
swapAndLiquify
swapTokensForEth
deliver
_takeCostFund
_reflectFee
_takeLiquidity
_getValues
tokenFromReflection
removeAllFee
_getValue
addLiquidity
approve
decreaseAllowance
increaseAllowance
_getRate
_getCurrentSupply
_getTValues
_getRValues
calculateBurnTax
calculateCostOperationTax
calculateLiquidityFee
calculateTaxFee
owner
IUniswapV2Router02 (iface)
removeLiquidityETHSupportingFeeOnTransferTokens
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
<Constructor>
owner
onlyOwner
renounceOwnership
transferOwnership
Context
_msgSender
payable
_msgData

Summary

Eagle AI contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions, manipulate the fees and massively blacklist addresses. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io