Cyberscope

Audit Report

LunarSphinx

January 2024

SHA256 117fb6e0328e1e64c6d395cbc97c01909f55125d3f226dab7bbdd2c2d4ce0606

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	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	IDI	Immutable Declaration Improvement	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved

Table of Contents

Review

Contract Name	LunarSphinx
Compiler Version	v0.8.0+commit.c7dfd78e
Optimization	200 runs
Testing Deploy	https://mumbai.polygonscan.com/address/0xba22e222d68a4c1b44004f7e0add6a8ec429f067
Address	0xba22e222d68a4c1b44004f7e0add6a8ec429f067
Network	MATIC Mumbai
Symbol	Pry
Decimals	18
Total Supply	10,000,000
Badge Eligibility	Yes

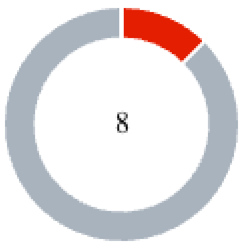
Audit Updates

Initial Audit	11 Jan 2024 https://github.com/cyberscope-io/audits/blob/main/lunarsphinx/v1/audit.pdf
Corrected Phase 2	15 Jan 2024

Source Files

Filename	SHA256
contracts/LunarSphinx.sol	117fb6e0328e1e64c6d395cbc97c01909f55125d3f226dab7bbdd2c2d4ce0606

Findings Breakdown



●	Critical	1
●	Medium	0
●	Minor / Informative	7

Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	1	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	7	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	contracts/LunarSphinx.sol
Status	Unresolved

Description

The contract owner has the authority to stop the sales for all users excluding the owner, as described in detail in the PLPI and PVC sections. As a result, the contract may operate as a honeypot.

Recommendation

The team is strongly encouraged to adhere to the recommendations outlined in the respective sections. By doing so, the contract can eliminate any potential of operating as a honeypot.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	contracts/LunarSphinx.sol#L523,525,534,540,546
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
_decimals
_tTotal
_teamAddress
swapAndLiquifyEnabled
uniswapV2Pair
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	contracts/LunarSphinx.sol#L750,755
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.

```
_isExcludedFromFee[account] = true;
_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.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	contracts/LunarSphinx.sol#L1040
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.

```
function swapTokensForEth(uint256 tokenAmount) private {
    // generate the uniswap pair path of token -> weth
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = uniswapV2Router.WETH();

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

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

Recommendation

To mitigate this potential vulnerability, it is advised to ensure that the primary pair between the contract's token and the native currency is consistently provided with liquidity. The team should actively verify and sustain adequate liquidity in the primary pair to safeguard against any disruptions in the token swapping functionality. Furthermore, implementing a mechanism within the contract to check if the pair has adequate liquidity provisions during runtime is advised. This feature allows the contract to omit the swap functionality if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures. Additionally, to enhance robustness, the contract should 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-revertable manner, thereby ensuring a more resilient and predictable operation.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	contracts/LunarSphinx.sol#L785
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `numTokensSellToAddToLiquidity` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function setSwapBackSettings(uint256 _amount) external onlyOwner {
    require(
        _amount >= totalSupply().mul(5).div(10**4),
        "Swapback amount should be at least 0.05% of total supply"
    );
    numTokensSellToAddToLiquidity = _amount;
    emit SwapAndLiquifyAmountUpdated(_amount);
}
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	contracts/LunarSphinx.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/LunarSphinx.sol#L395,469,472,475,480,777,785,924,928,936
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 _taxFee
uint256 public _liquidityFee
uint256 public _teamFee
address public _teamAddress
uint256 TeamFeeBps
uint256 _amount
```

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.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	contracts/LunarSphinx.sol#L759,770,778
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
_taxFee = taxFeeBps
_liquidityFee = liquidityFeeBps
_teamFee = TeamFeeBps
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	

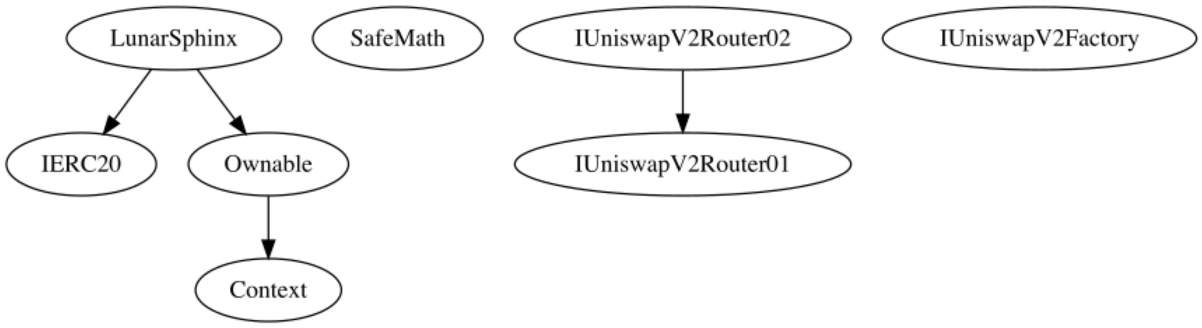
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
IUniswapV2Router02	Interface	IUniswapV2 Router01		

	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IUniswapV2Factory	Interface			
	createPair	External	✓	-
LunarSphinx	Implementation	IERC20, Ownable		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	isExcludedFromReward	Public		-
	totalFees	Public		-
	deliver	Public	✓	-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-

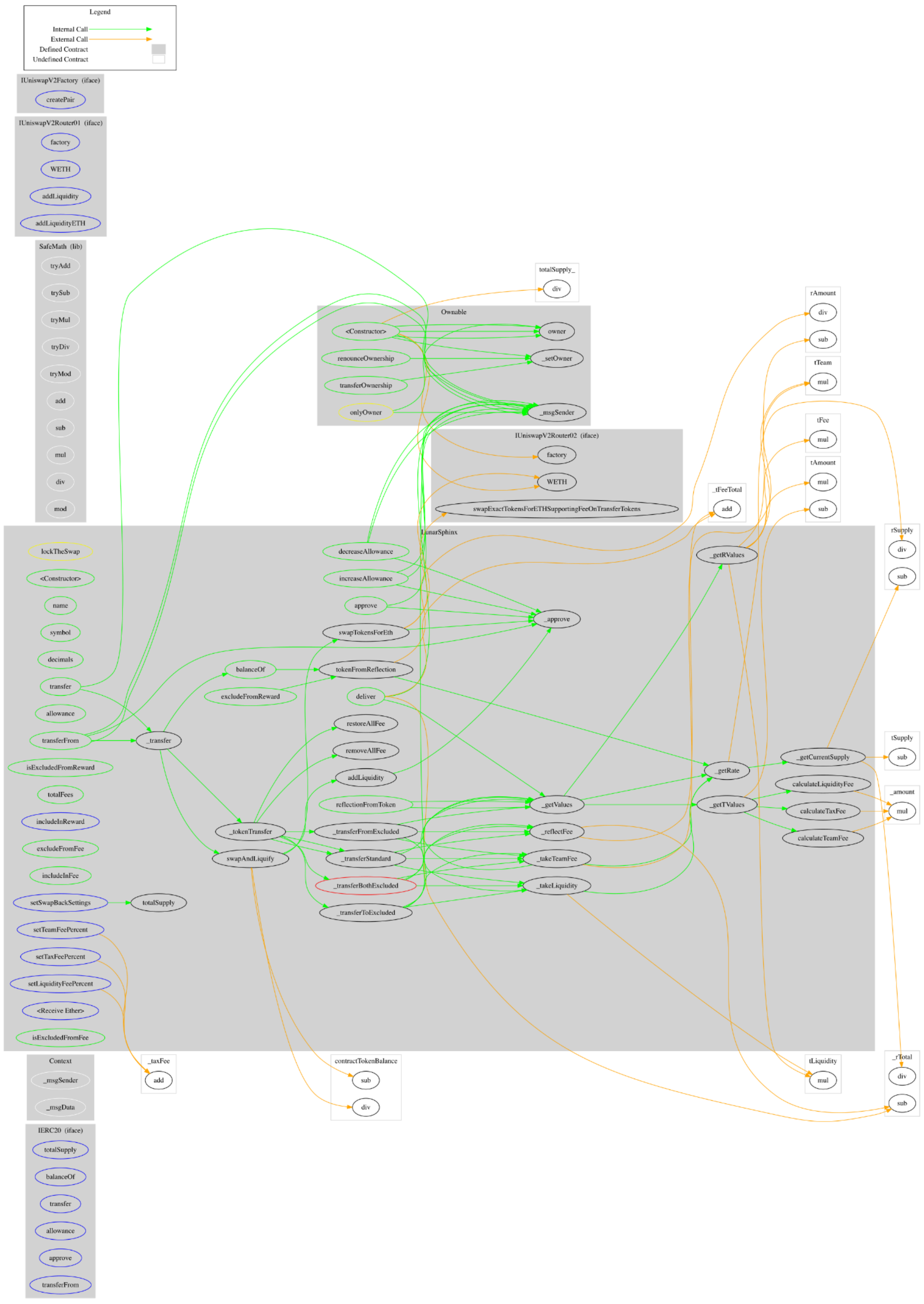
	excludeFromReward	Public	✓	onlyOwner
	includeInReward	External	✓	onlyOwner
	_transferBothExcluded	Private	✓	
	excludeFromFee	Public	✓	onlyOwner
	includeInFee	Public	✓	onlyOwner
	setTaxFeePercent	External	✓	onlyOwner
	setLiquidityFeePercent	External	✓	onlyOwner
	setTeamFeePercent	External	✓	onlyOwner
	setSwapBackSettings	External	✓	onlyOwner
		External	Payable	-
	_reflectFee	Private	✓	
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidity	Private	✓	
	_takeTeamFee	Private	✓	
	calculateTaxFee	Private		
	calculateLiquidityFee	Private		
	calculateTeamFee	Private		
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	

	isExcludedFromFee	Public		-
	_approve	Private	✓	
	_transfer	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
IUniswapV2Factory (iface)
createPair
IUniswapV2Router01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
totalSupply_
div
Ownable
<Constructor>
owner
renounceOwnership
_setOwner
transferOwnership
onlyOwner
_msgSender
IUniswapV2Router02 (iface)
factory
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
rAmount
div
sub
tTeam
mul
tFee
mul
tAmount
mul
sub
_tFeeTotal
add
rSupply
div
sub
LunarSphinx
lockTheSwap
<Constructor>
name
symbol
decimals
transfer
allowance
transferFrom
isExcludedFromReward
totalFees
includeInReward
excludeFromFee
includeInFee
setSwapBackSettings
setTeamFeePercent
setTaxFeePercent
setLiquidityFeePercent
<Receive Ether>
isExcludedFromFee
decreaseAllowance
increaseAllowance
approve
swapTokensForEth
balanceOf
tokenFromReflection
excludeFromReward
deliver
_transfer
restoreAllFee
removeAllFee
addLiquidity
reflectionFromToken
_tokenTransfer
swapAndLiquify
_transferFromExcluded
_transferStandard
_transferBothExcluded
_transferToExcluded
totalSupply
_approve
_getRValues
_getRate
_getCurrentSupply
_getValues
_getTValues
_reflectFee
_takeTeamFee
_takeLiquidity
calculateLiquidityFee
calculateTaxFee
calculateTeamFee
tSupply
sub
_amount
mul
Context
_msgSender
_msgData
_taxFee
add
contractTokenBalance
sub
div
tLiquidity
mul
_rTotal
div
sub
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom

Summary

LunarSphinx contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. LunarSphinx is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 25% fees.

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