



Cyberscope

Audit Report

Mythic Ore

January 2023

Network BSC

Address 0xd6163ceC51F2e7C5974F3F4Ce8FDb9c80ABF142e

Audited by © cyberscope

Table of Contents

Table of Contents	1
Contract Review	3
Audit Updates	4
Source Files	4
Contract Architecture Concern	5
Team's Reply 29 November 2022	5
Team's Update 14 December 2022	5
Contract Analysis	6
US - Untrusted Source	7
Description	7
Recommendation	7
TSD - Total Supply Diversion	8
Description	8
Recommendation	8
Team Update	8
RLS - Redundant Liquidity Swaps	10
Description	10
Recommendation	10
Team Update	10
RM - Reflection Mechanism	11
Description	11
Recommendation	11
Team Update	11
TUU - Time Units Usage	12
Description	12
Recommendation	12

L04 - Conformance to Solidity Naming Conventions	13
Description	13
Recommendation	14
L13 - Divide before Multiply Operation	15
Description	15
Recommendation	15
L20 - Succeeded Transfer Check	16
Description	16
Recommendation	16
Contract Functions	17
Inheritance Graph	24
Contract Flow	25
Domain Info	26
Summary	27
Team Update 29 November 2022	27
Team's Update 14 December 2022	27
Disclaimer	28
About Cyberscope	29

Contract Review

Contract Name	MORE
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	1024 runs
Explorer	https://bscscan.com/address/0xd6163cec51f2e7c5974f3f4ce8fdb9c80abf142e
Address	0xd6163cec51f2e7c5974f3f4ce8fdb9c80abf142e
Network	BSC
Symbol	MORE
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	24th November 2022
Corrected Phase 1	29th November 2022
Corrected Phase 2	30th November 2022
Corrected Phase 3	6th December 2022
Corrected Phase 4	07 Jan 2023
Corrected Phase 5	11 Jan 2023

Source Files

Filename	SHA256
Interfaces/IAgent.sol	513457f2c6850d6c7195aa548e6285916f45cff3592db31dcb6897a7e9cd974a
Interfaces/IERC20.sol	597b75adbc8cea6b702b97509ef3c2576b3ee79fd0260e30eedd8bc41217b641
Interfaces/IUniswap.sol	4182998eb45fcf08cec84ad68108149ba375dd5404e3252e624e2d163bf32c38
More.sol	ff12303069bb3f1d6c0d1b53f35d0e959d37d95ef56e6b3c70b0a5eaaa368853

Contract Architecture Concern

The contract implements an ERC20 token enriched with some features like reflections and autogenerated liquidity pool. The implementation of the contract is custom and it is not based on any well-known implementation. As a result, some concepts and methodologies like [reflections](#), [gas optimization](#), etc. are not could be more well-structured. The team is advised to fork a well-known ERC20 implementation that contains the same features and apply their requirements.

Team's Reply 29 November 2022

The team has acknowledged that this is not a security issue and states:

"We are not using Safemoon-fork reflections system cause:

- 1. It is expensive in terms of gas too.*
- 2. It is harder to implement considering we have a different method of share calculations.*
- 3. So all gas optimizations are targeted mainly to compensate costly reflections system."*

Team's Update 14 December 2022

"7.5% locked for liquidity injection. Liquidity gathered from fees can be unlocked but we have a separate lock mechanism for that again in the contract."

Contract Analysis

● Critical ● Medium ● Minor

Severity	Code	Description	Status
●	US	Untrusted Source	Unresolved
●	TSD	Total Supply Diversion	Acknowledged
●	RLS	Redundant Liquidity Swaps	Acknowledged
●	RM	Reflection Mechanism	Acknowledged
●	TUU	Time Units Usage	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

US - Untrusted Source

Criticality	minor
Status	Unresolved

Description

The contract uses an external contract in order to determine the transaction's flow. The external contract is untrusted. As a result, it may produce security issues and harm the transactions.

```
Agent = IAgent(newAgent);  
...  
Agent.delegate{value: agentPotToSend}(buyPot, sellPot, transferPot, teamPot,  
referrerPot, tokensUsedForReferrersPot);  
...  
Agent.tryToLock(msg.sender, amount, lockConfig.nextLockIndex, duration);
```

Recommendation

The contract should use a trusted external source. A trusted source could be either a commonly recognized or an audited contract. The pointing addresses should not be able to change after the initialization. Additionally, as a safety measure, all external calls to untrusted addresses should be wrapped in a try-catch statement.

TSD - Total Supply Diversion

Criticality	minor
Location	More.sol#L1335
Status	Acknowledged

Description

The total supply of a token is the total number of tokens that have been created, while the balances of individual accounts represent the number of tokens that an account owns. The total supply and the balances of individual accounts are two separate concepts that are managed by different variables in a smart contract. These two entities should be equal to each other.

In the contract, the amount that is added to the total supply does not equal the amount that is added to the balances. As a result, the sum of balances is diverse from the total supply.

```
_balanceOf[account] += _reflected;
```

Recommendation

The total supply and the balance variables are separate and independent from each other. The total supply represents the total number of tokens that have been created, while the balance mapping stores the number of tokens that each account owns. The sum of balances should always equal the total supply.

Team Update

The team has acknowledged that this is not a security issue and states:

"This amount is deducted from the total supply in notifyTaxSystem() then its continuously being added here until reflection cycle is finished while it is true that the sum of all balances won't be equal to the total supply in most cases in the end, they will become equal (minus some small amount as a result of rounding down on divisions) but sum of all balances will not ever be greater than total supply so your comment that "The amount that is added to the total supply does not equal the amount that is added to the balances" is not true because tokens here are not

actually being minted or added to the total supply but as stated in a comment above it is true that sum of all balances will be slightly lower than total supply personally I do not think this is a major issue.”

RLS - Redundant Liquidity Swaps

Criticality	Minor
Location	More.sol#L1476,1494
Status	Acknowledged

Description

In order to accumulate tokenLiquidityReserves the contract swap tokens for BNB and then swap back the proportional BNB for tokens.

```
function addLiquidityFromTokenReserves() private
{...}

function refillLiquidityTokenReserves() private
{...}
```

Recommendation

The contract could accumulate the tokens directly from the liquidity fees.

Team Update

The team has acknowledged that this is not a security issue and states:

"Initially tokenLiquidityReserves is filled so no redundant swaps then after some time it is true that redundant swap are gonna happen, but: it saves just a little bit of gas to not write to tokenLiquidityReserves on every taxed tx it is not useful to write every time to tokenLiquidityReserves as initially it is filled we were planning to fill it after deployment but it is surely more reasonable to do it here that way instead of using 50/50 system we forward 100% bnb from tax to liquidity and it works while tokenLiquidityReserves is filled not as a result of a swap."

RM - Reflection Mechanism

Criticality	Minor
Location	More.sol#L329,1017
Status	Acknowledged

Description

The contract uses a complicated technique to send the reflected tokens to each account. On every transfer, the sender's and the receiver's balance is updated according to the corresponding reflected amount. This process produces a large amount of gas cost proportionally to the number of transfers.

```
updateReflections(sender);  
...  
updateReflections(recipient);
```

Recommendation

The contract could use a simpler reflections mechanism that is based on a classic safemoon fork.

<https://github.com/safemoonprotocol/Safemoon.sol/blob/main/Safemoon.sol>

Team Update

The team has acknowledged that this is not a security issue.

The team response is mentioned in the **Contract Architecture** section.

TUU - Time Units Usage

Criticality	Minor / Informative
Location	More.sol#L426
Status	Unresolved

Description

The contract is using arbitrary numbers to form time-related values. As a result, it decreases the readability of the codebase and prevents the compiler to optimize the source code.

```
require(lock.unlockTimestamp + additionalTime - block.timestamp < 86400 * 366  
+ 1, "LET2");
```

Recommendation

It is a good practice to use the time units reserved keywords like `seconds`, `minutes`, `hours`, `days`, `weeks` and `years` to process time-related calculations.

It's important to note that these time units are simply a shorthand notation for representing time in seconds, and do not have any effect on the actual passage of time or the execution of the contract. The time units are simply a convenience for expressing time in a more human-readable form.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	More.sol#L52,54,85,103,105,106,128,129,131,144,199,201,202,203
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.

```
uint256 private constant _decimals = 18
uint256 private constant _totalSupply = MAX_SUPPLY
mapping(address => ModifiersData) public Modifiers
mapping(address => uint256) private ReflectionsPerSharePaid
ListAddress.ListStruct private Shareholders
ListAddress.ListStruct private AuthorizedContracts
mapping(address => LockConfig) public LockConfigs
mapping(address => mapping(uint256 => LockInstance)) public Locks
mapping(uint256 => uint256) public MaxCompoundingIterations
mapping(uint256 => TaxData) private Taxes
IUniswapV2Router02 private SwapRouter
address public SwapAgent
address public MainAccount
IAgent public Agent
```

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>.

L13 - Divide before Multiply Operation

Criticality	Minor
Location	More.sol#L1207,1209,1211,1217,1220,1226,1244,1387,1388,1389,1390,1391,1418,1419,1420,1421,1422,1425
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of precision.

```
transferTokens = toSwap * _tokenPots.transfer / totalCompressedTokens
_potsBNB.transfer += uint80(receivedBNB * transferTokens / toSwap)
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L20 - Succeeded Transfer Check

Criticality	Minor
Location	More.sol#L756
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IERC20(tokenContract).transfer(msg.sender, balance)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the [Openzeppelin library](#).

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IAgent	Interface			
	delegate	External	Payable	-
	marketplaceDelegate	External	Payable	-
	notifyTransferListener	External	✓	-
	notifyTransferListener	External	✓	-
	tryToLock	External	✓	-
	tryToLockExtra	External	✓	-
	tryToUnlock	External	✓	-
IERC20	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-

	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	swap	External	✓	-

	sync	External	✓	-
	initialize	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-

	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
ListAddress	Library			
	add	Internal	✓	
	remove	Internal	✓	
MORE	Implementation	IERC20		
	_onlyMain	Private		
	_onlyAuthorized	Private		
	_onlySwap	Private		
	_flagCheck	Private		
		Public	✓	-
		External	Payable	-
	transferFrom	External	✓	-
	transfer	External	✓	-
	lightningTransfer	External	✓	onlyAuthorized
	prepareReferralSwap	External	✓	onlySwap
	approve	External	✓	-
	toggleLocks	External	✓	flagCheck
	lockTokens	External	✓	-
	lockExtraTokens	External	✓	-
	_beforeLock	Private	✓	
	unlockTokens	External	✓	-
	unlockTokensEarly	External	✓	-
	_beforeUnlock	Private	✓	
	tryToUnlock	External	✓	-
	setModifiers	External	✓	onlyAuthorized
	setModifiers	External	✓	onlyAuthorized

	addMultiplier	External	✓	onlyAuthorized
	setBuyTaxReduction	External	✓	onlyAuthorized
	setSellTaxReduction	External	✓	onlyAuthorized
	addTokensToLiquidityReservesFromContract	External	✓	onlyAuthorized
	addBNBToLiquidityPot	External	Payable	-
	buybackAndBurn	External	Payable	-
	buybackAndLockToLiquidity	External	Payable	-
	addAuthorized	External	✓	onlyMain
	removeAuthorized	External	✓	onlyMain
	lockLiquidityFromFees	External	✓	onlyMain
	withdrawLiquidityFromFees	External	✓	onlyMain
	toggleSellAddress	External	✓	onlyMain flagCheck
	toggleAccountTaxExclusion	External	✓	onlyMain
	toggleAccountMaxAccountRuleExclusion	External	✓	onlyMain flagCheck
	setReferralTaxReduction	External	✓	onlyMain
	setMaxAccountAndMaxMultiplier	External	✓	onlyMain
	setReflectionsDelayAndDistributingPart	External	✓	onlyMain
	setMaxCompoundingIterations	External	✓	onlyMain
	setMinGasForWork	External	✓	onlyMain
	setTax	External	✓	onlyMain
	setWorkAmounts	External	✓	onlyMain
	setMainAccount	External	✓	onlyMain
	setAgents	External	✓	onlyMain
	addToReflectionsFromContract	External	✓	onlyAuthorized
	withdrawFreeBNB	External	✓	onlyMain
	withdrawFreeTokens	External	✓	onlyMain
	launchToken	External	✓	onlyMain
	balanceOf	External		-

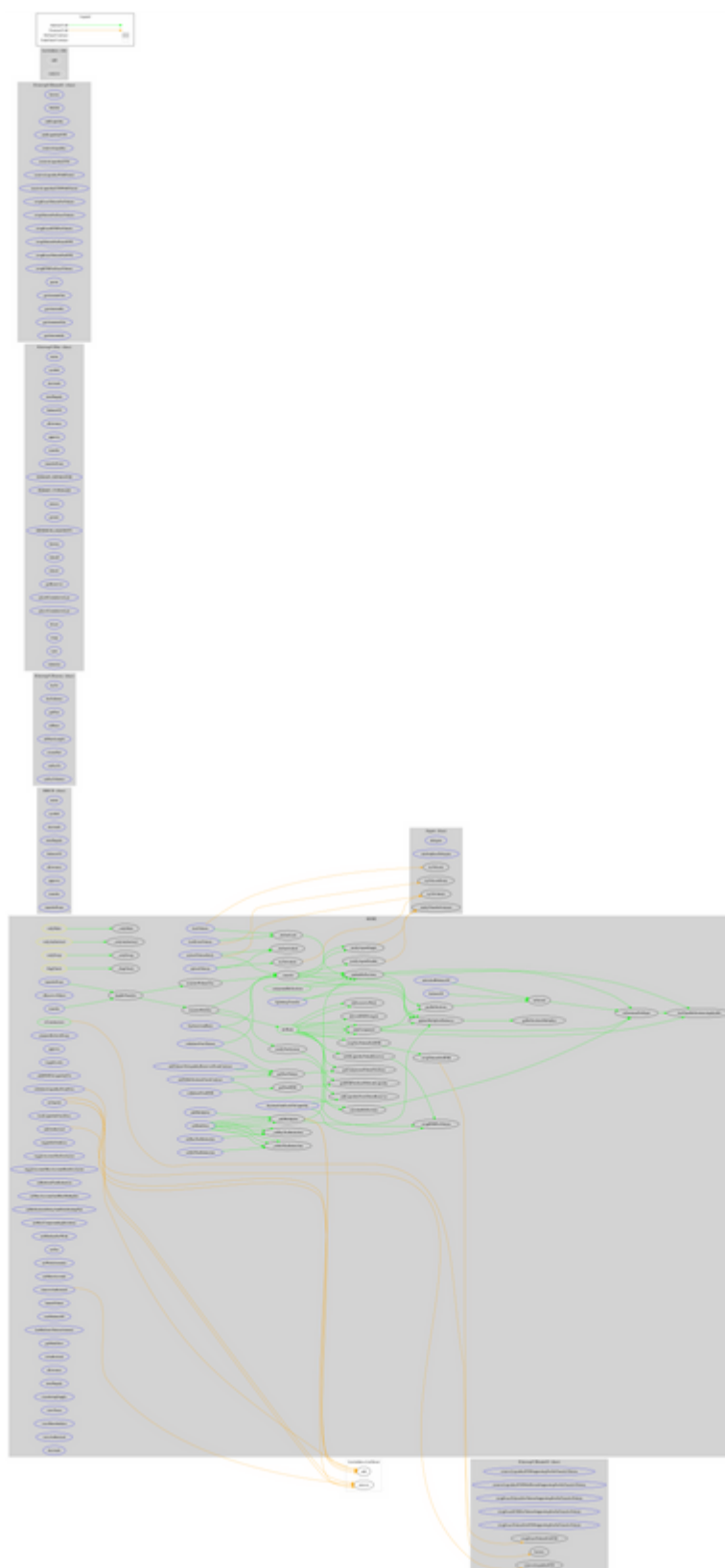
	unlockedBalanceOf	External		-
	rawBalanceOf	External		-
	lastReferrerTokensAmount	External		-
	getModifiers	External		-
	getModifiers	External		-
	isAuthorized	External		-
	allowance	External		-
	totalSupply	External		-
	circulatingSupply	External		-
	viewTaxes	External		-
	viewShareholders	External		-
	viewAuthorized	External		-
	decimals	External		-
	doWork	Public	✓	-
	doExcessiveWork	Private		
	autoCompound	Public	✓	-
	compoundReflections	Public	✓	-
	reflected	Public		-
	reflected	Private		
	getFreeTokens	Public		-
	getFreeBNB	Public		-
	_transfer	Internal	✓	
	notifyAgentSingle	External	✓	-
	notifyAgentDouble	External	✓	-
	handleTransfer	Private	✓	
	transferWithTax	Private	✓	
	transferWithoutTax	Private	✓	
	deliverBNBToAgent	External	✓	-
	notifyTaxSystem	Private	✓	

	calculateReflections	Private	✓	
	updateMultiplierBalances	Private	✓	
	updateReflections	Private	✓	
	payReflections	Private	✓	
	lastTimeReflectionsApplicable	Private		
	reflectionsPerShare	Private		
	getReflectionsMultiplier	Public		-
	swapTaxTokensForBNB	Private	✓	
	swapTokensForBNB	Private	✓	
	swapBNBForTokens	Private	✓	
	addLiquidityFromTokenReserves	Private	✓	
	refillLiquidityTokenReserves	Private	✓	
	_addMultiplier	Private	✓	
	_setBuyTaxReduction	Private	✓	
	_setSellTaxReduction	Private	✓	
	getCompressedTokenPotsSum	Private		
	getBNBPotsSumWithoutLiquidity	Private		

Inheritance Graph



Contract Flow



Domain Info

Domain Name	mythicore.io
Registry Domain ID	c06b3a9a39654b128ef40cd39b633b96-DONUTS
Creation Date	2022-07-22T16:37:58Z
Updated Date	2022-07-27T16:38:54Z
Registry Expiry Date	2023-07-22T16:37:58Z
Registrar WHOIS Server	whois.godaddy.com/
Registrar URL	http://www.godaddy.com/domains/search.aspx?ci=8990
Registrar	GoDaddy.com, LLC
Registrar IANA ID	146

The domain was created 5 months before the creation of the audit. It will expire in 8 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

The Mythic Ore contract implements an ERC20 token. This audit investigates security issues, business logic concerns and potential improvements.

Team Update 29 November 2022

The team has replied to all of the findings and has acknowledged that the remaining are not security issues.

Team's Update 14 December 2022

"7.5% locked for liquidity injection. Liquidity gathered from fees can be unlocked but we have a separate lock mechanism for that again in the contract."

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>