Cyberscope

Audit Report

BasedSwap

April 2024

Network BASE

Address 0x314da69dE85145FDd5b7580971e9DB0388A2cDc4

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

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

Diagnostics

Severity	Code	Description	Status
●	MEE	Missing Events Emission	Unresolved
●	PAMAR	Pair Address Max Amount Restriction	Renounced
●	RSML	Redundant SafeMath Library	Unresolved
●	OCTD	Transfers Contract's Tokens	Renounced
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L05	Unused State Variable	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Contract Name	BASEDSWAP
Compiler Version	v0.8.9+commit.e5eed63a
Optimization	200 runs
Explorer	https://basescan.org/token/0x314da69dE85145FDd5b7580971e9DB0388A2cDc4
Address	0x314da69dE85145FDd5b7580971e9DB0388A2cDc4
Network	BASE
Symbol	BSW
Decimals	18
Total Supply	100,000,000

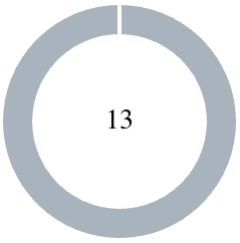
Audit Updates

Initial Audit	13 Apr 2024

Source Files

Filename	SHA256
BASEDSWAP.sol	026ea2c56cf59e4213fe75dbb6101a71718d0236a853c8acbb6831cbd5a125b1

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	13

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	10	0	3	0

ELFM - Exceeds Fees Limit

Criticality	Minor / Informative
Location	BASEDSWAP.sol#L644
Status	Renounced

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 `updateFees` function with a percentage up to 50%.

```
function updateFees(uint8 _marketingFeeBuy, uint8
_liquidityFeeBuy,uint8 _devFeeBuy,uint8 _marketingFeeSell, uint8
_liquidityFeeSell,uint8 _devFeeSell) external onlyOwner{
    _fees.buyMarketingFee = _marketingFeeBuy;
    _fees.buyLiquidityFee = _liquidityFeeBuy;
    _fees.buyDevFee = _devFeeBuy;
    _fees.buyTotalFees = _fees.buyMarketingFee +
_fees.buyLiquidityFee + _fees.buyDevFee;

    _fees.sellMarketingFee = _marketingFeeSell;
    _fees.sellLiquidityFee = _liquidityFeeSell;
    _fees.sellDevFee = _devFeeSell;
    _fees.sellTotalFees = _fees.sellMarketingFee +
_fees.sellLiquidityFee + _fees.sellDevFee;
    require(_fees.buyTotalFees <= 50, "Must keep fees at 50% or
less");
    require(_fees.sellTotalFees <= 50, "Must keep fees at 50% or
less");
}
```

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.

OCTD - Transfers Contract's Tokens

Criticality	Minor / Informative
Location	BASEDSWAP.sol#L678
Status	Renounced

Description

The contract accumulates the tokens from the taxes on every transaction. These tokens are transferred to the contract address and increase the corresponding variables tokensForLiquidity, tokensForMarketing, tokensForDev. These variables are used to swap the accumulated tokens in the swapBack method. If the owner executed the `rescueERC20` method providing the contract address then the swapBack() will revert since there will be inconsistency between the state variables and the actual amount.

```
tokensForLiquidity += fees * _fees.sellLiquidityFee /
_fees.sellTotalFees;
tokensForMarketing += fees * _fees.sellMarketingFee /
_fees.sellTotalFees;
tokensForDev += fees * _fees.sellDevFee / _fees.sellTotalFees;

function swapBack() private {
    uint256 contractTokenBalance = balanceOf(address(this));
    uint256 toSwap = tokensForLiquidity + tokensForMarketing +
tokensForDev;

function rescueERC20(address tokenAdd, uint256 amount) external
onlyOwner {
    IERC20(tokenAdd).transfer(owner(), amount);
}
```

Recommendation

The team is advised to update the contract functionality so it will not be able to produce inconsistency between the state variables and the actual state.

PAMAR - Pair Address Max Amount Restriction

Criticality	Minor / Informative
Location	BASEDSWAP.sol#L719
Status	Renounced

Description

The contract owner has the authority to set the maximum wallet amount that the recipient can receipt. This percentage could be more than 1%. If the owner excludes the pair address from the `_isExcludedMaxWalletAmount` then the sale will stop since the pair addresses usually accumulate a large amount of tokens that is much more than 1%. This behavior may lead the contract to become a honeypot.

```
function excludeFromWalletLimit(address account, bool excluded)
public onlyOwner {
    _isExcludedMaxWalletAmount[account] = excluded;
}
    ...
if (!_isExcludedMaxWalletAmount[recipient]) {
    require(amount + balanceOf(recipient) <= maxWalletAmount, "Max
wallet exceeded");
}
```

Recommendation

The team is adviced to conduct a thorough review of the criteria used to exclude addresses from the maximum wallet amount limit. Consider the implications of excluding pair addresses and assess whether alternative approaches or additional safeguards are necessary to address the unintended consequences described.

MEE - Missing Events Emission

Criticality	Minor / Informative
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 excludeFromFees(address account, bool excluded) public
onlyOwner {
    _isExcludedFromFees[account] = excluded;
}
...
function excludeFromMaxTransaction(address updAds, bool isEx)
public onlyOwner {
    _isExcludedMaxTransactionAmount[updAds] = isEx;
}
```

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.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	BASEDSWAP.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 in cases where the explanatory error message is not used.

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

Recommendation

The team is advised to remove the SafeMath library in cases where the revert error message is not used. 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	BASEDSWAP.sol#L356,525,545,546,644,682
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);

Fees public _fees = Fees({
        buyTotalFees: 0,
        buyMarketingFee: 0,
        buyDevFee:0,
        buyLiquidityFee: 0,

        sellTotalFees: 0,
        sellMarketingFee: 0,
        sellDevFee:0,
        sellLiquidityFee: 0
    })
mapping(address => bool) public _isExcludedMaxTransactionAmount
mapping(address => bool) public _isExcludedMaxWalletAmount

...
```

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	BASEDSWAP.sol#L291
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.

```
int256 private constant MAX_INT256 = ~(int256(1) << 255)
```

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.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	BASEDSWAP.sol#L626,634,641
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.

```
thresholdSwapAmount = newAmount
maxBuyAmount = (totalSupply() * newMaxBuy) / 1000
maxWalletAmount = (totalSupply() * newPercentage) / 1000
```

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.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	BASEDSWAP.sol#L333,339,346
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function abs(int256 a) internal pure returns(int256) {
        require(a != MIN_INT256);
        return a < 0 ? -a : a;
    }

function toUint256Safe(int256 a) internal pure returns(uint256) {
...
    }

function toInt256Safe(uint256 a) internal pure returns(int256) {
    int256 b = int256(a);
        require(b >= 0);
        return b;
    }
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	BASEDSWAP.sol#L756,757,758,759,763,764,765,766
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 prediction.

```
fees = amount.mul(_fees.buyTotalFees).div(100)
tokensForMarketing += fees * _fees.buyMarketingFee /
_fees.buyTotalFees
```

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.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	BASEDSWAP.sol#L581
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
uint256 totalSupply = 1e8 * 1e18
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	BASEDSWAP.sol#L683,684
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
marketingWallet = _marketingWallet
devWallet = _devWallet
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	BASEDSWAP.sol#L679
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(tokenAdd).transfer(owner(), amount)
```

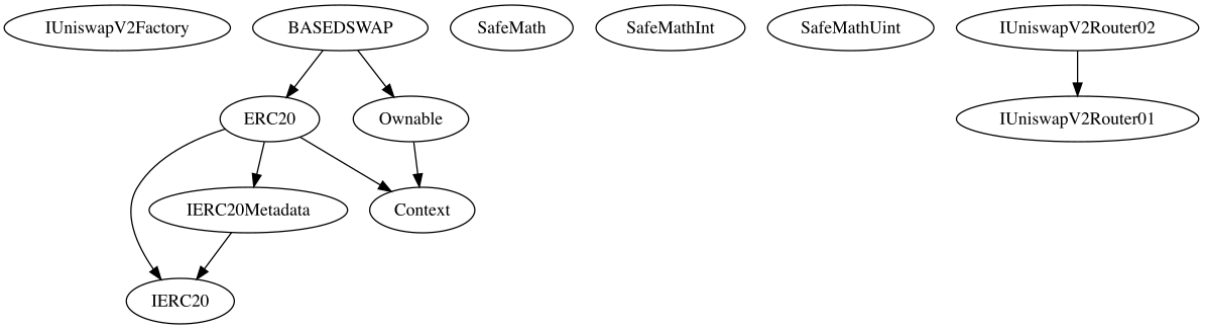
Recommendation

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.

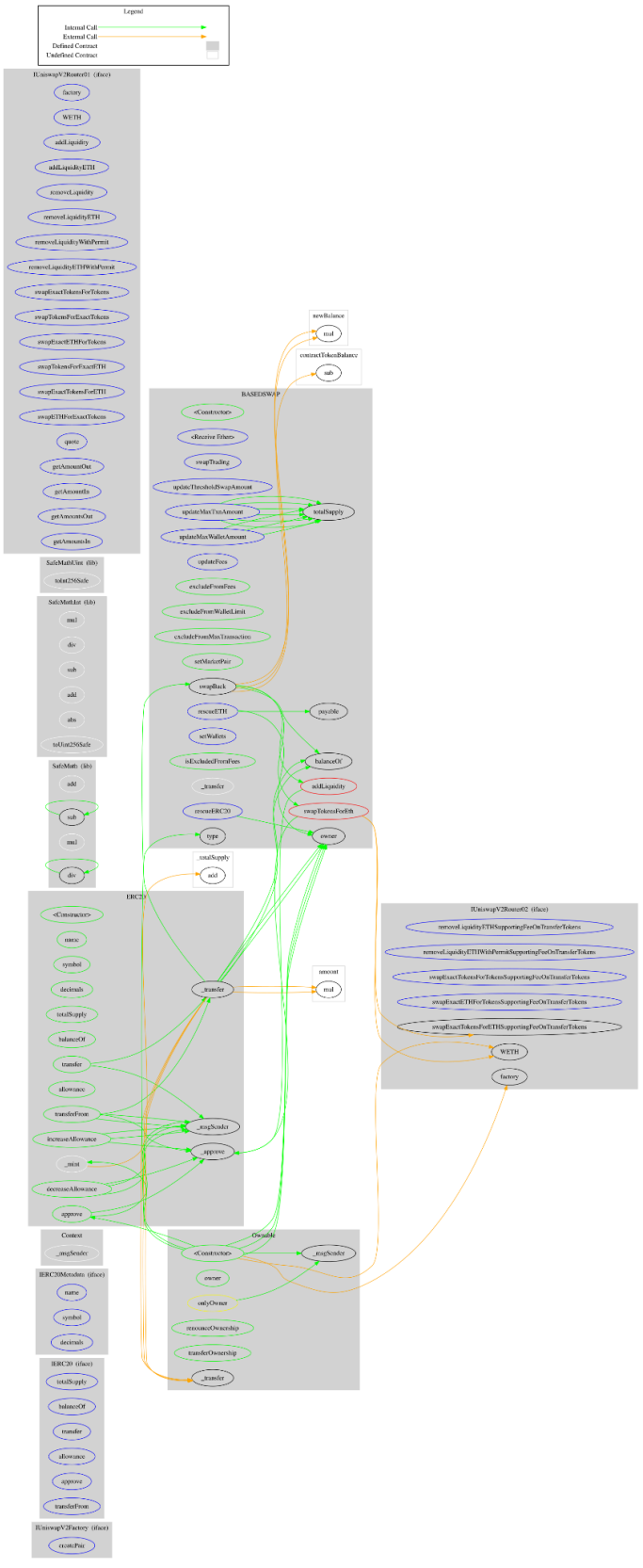
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
BASEDSWAP	Implementation	ERC20, Ownable		
		Public	✓	ERC20
		External	Payable	-
	swapTrading	External	✓	onlyOwner
	updateThresholdSwapAmount	External	✓	onlyOwner
	updateMaxTxnAmount	External	✓	onlyOwner
	updateMaxWalletAmount	External	✓	onlyOwner
	updateFees	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	excludeFromWalletLimit	Public	✓	onlyOwner
	excludeFromMaxTransaction	Public	✓	onlyOwner
	setMarketPair	Public	✓	onlyOwner
	rescueETH	External	✓	onlyOwner
	rescueERC20	External	✓	onlyOwner
	setWallets	External	✓	onlyOwner
	isExcludedFromFees	Public		-
	_transfer	Internal	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	swapBack	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
SafeMathUint (lib)
toInt256Safe
SafeMathInt (lib)
mul
div
sub
add
abs
toUint256Safe
SafeMath (lib)
add
sub
mul
div
IERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
increaseAllowance
_msgSender
_approve
_mint
decreaseAllowance
approve
Context
_msgSender
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IUniswapV2Factory (iface)
createPair
newBalance
real
contractTokenBalance
sub
BASEDSWAP
<Constructor>
<Receive Ether>
swapTrading
updateThresholdSwapAmount
updateMaxTxnAmount
updateMaxWalletAmount
totalSupply
updateFees
excludeFromFees
excludeFromWalletLimit
excludeFromMaxTransaction
setMarketPair
swapBack
rescueETH
payable
setWallets
isExcludedFromFees
balanceOf
_transfer
addLiquidity
rescueERC20
swapTokensForEth
type
owner
totalSupply
add
_transfer
amount
real
Ownable
<Constructor>
_msgSender
owner
onlyOwner
renounceOwnership
transferOwnership
_transfer
IUniswapV2Router02 (iface)
removeLiquidityETHSupportingFeeOnTransferTokens
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory

Summary

BasedSwap contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. The contract owner has renounced the ownership.

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