Cyberscope

Audit Report

Global Reserve Coin

May 2023

Network BSC

Address 0x8e4653Ef0f8Ce731653192bb642D5347Ccec2c6a

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	ZD	Zero Division	Unresolved
●	DDP	Decimal Division Precision	Unresolved
●	TUU	Time Units Usage	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Contract Name	GlobalReserveCoin
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://bscscan.com/address/0x8e4653ef0f8ce731653192bb642d5347ccec2c6a
Address	0x8e4653ef0f8ce731653192bb642d5347ccec2c6a
Network	BSC
Symbol	GRC
Decimals	9
Total Supply	100,000,000,000

Audit Updates

Initial Audit	28 May 2023

Source Files

Filename	SHA256
GlobalReserveCoin.sol	24903a8601574e25bb2e087e3d9de5f15940937cddf33fa97516f6cae82af0a7

Findings Breakdown

Critical	1
Medium	0
Minor / Informative	12

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	12	0	0	0

ZD - Zero Division

Criticality	Critical
Location	GlobalReserveCoin.sol#L1407
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. This can lead to unpredictable and potentially harmful results, such as a transaction revert.

The variable `_FeesTotal` could potentially hold the value of zero if both all fees are set to zero.

```
uint256 LP_Tokens       = Tokens * (_Fee__Buy_Liquidity +
_Fee__Sell_Liquidity) / _FeesTotal / 2;
```

Recommendation

It is important to handle division by zero appropriately in the code to avoid unintended behavior and to ensure the reliability and safety of the contract. The contract should ensure that the divisor is always non-zero before performing a division operation. It should prevent the variables to be set to zero, or should not allow the execution of the corresponding statements.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	GlobalReserveCoin.sol#L1419
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
uint256 BNB_Liquidity    = returned_BNB * (_Fee__Buy_Liquidity    +
_Fee__Sell_Liquidity)       / fee_Split;
uint256 BNB_Rewards      = returned_BNB * (_Fee__Buy_Rewards      +
_Fee__Sell_Rewards)      * 2 / fee_Split;
uint256 BNB_GOLD         = returned_BNB * (_Fee__Buy_Gold         +
_Fee__Sell_Gold)         * 2 / fee_Split;
```

Recommendation

The team is advised to take into consideration the rounding results that are produced from the solidity calculations. The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

TUU - Time Units Usage

Criticality	Minor / Informative
Location	GlobalReserveCoin.sol#L924
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(Minutes_Between_Payments <= 2880,"E05");
```

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.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	GlobalReserveCoin.sol#L883,901,1044,1045.1107,1161,1172,1183,1193
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract modifies the state of some variables without checking if the current state of these variables is the same as the one given as an argument. As a result, the contract performs redundant storage writes.

```
No_Fee_Transfers = true_or_false;
_isExcludedFromRewards[Wallet_Address] = true_or_false;
_isPair[Wallet_Address] = true_or_false;
_isLimitExempt[Wallet_Address] = true_or_false;
swapAndLiquifyEnabled = true_or_false;
...
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	GlobalReserveCoin.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 at https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	GlobalReserveCoin.sol#L592,593
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.

```
uniswapV2Pair
uniswapV2Router
```

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.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	GlobalReserveCoin.sol#L363,377
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.

```
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 public dividendsPerShareAccuracyFactor = 10 ** 36
```

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	GlobalReserveCoin.sol#L165,166,179,198,351,360,363,365,399,404,537,538,539,545,546,547,548,551,552,553,556,557,560,561,562,563,565,566,567,568,571,572,635,636,637,638,651,652,653,654,655,656,663,712,747,748,766,768,769,770,771,793,795,796,797,798,834,836,837,853,881,898,921,934,951,953,963,965,975,977,995,997,1006,1008,1017,1019,1038,1040,1041,1060,1081,1106,1112,1122,1124,1125,1136,1155,1157,1158,1166,1168,1169,1177,1179,1180,1187,1189,1190,1273,1400,1480
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 DOMAIN_SEPARATOR() external view returns (bytes32);
function PERMIT_TYPEHASH() external pure returns (bytes32);
function MINIMUM_LIQUIDITY() external pure returns (uint);
function WETH() external pure returns (address);
address _token
IERC20 RWDTOKEN = IERC20(0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56)
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
IUniswapV2Router02 public DivRouter

function Claim_Rewards() external {
        distributeDividend(msg.sender);
    }
uint256 _minDistribution
uint256 _minPeriod
address public Wallet_Liquidity =
0x486A72846547268B3A9BbDdb7C5B75D92eE66f67

...
```

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	GlobalReserveCoin.sol#L405,937,1139
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.

```
minPeriod = _minPeriod
distributorGas = Gas_Amount
swapTrigger = Transaction_Count + 1
```

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	GlobalReserveCoin.sol#L73,79,85,89,93,97,104,108,115,119,125
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 isContract(address account) internal view returns (bool) {
        uint256 size;
        assembly { size := extcodesize(account) }
        return size > 0;
    }

...
        (bool success, ) = recipient.call{ value: amount }("");
        require(success, "unable to send, recipient reverted");
    }

function functionCall(address target, bytes memory data) internal returns
(bytes memory) {
      return functionCall(target, data, "low-level call failed");
    }

...
```

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	GlobalReserveCoin.sol#L725
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.

```
return (TokenSupply,
                halfPrecent,
                TokenSupply / 100,
                TokenSupply / 100 + halfPrecent,
                TokenSupply / 50,
                TokenSupply / 50 + halfPrecent,
                TokenSupply / 100 * 3)
```

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.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	GlobalReserveCoin.sol#L75,130
Status	Unresolved

Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly { size := extcodesize(account) }

assembly {
                    let returndata_size := mload(returndata)
                    revert(add(32, returndata), returndata_size)
                }
```

Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	GlobalReserveCoin.sol#L492,1131
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.

```
RWDTOKEN.transfer(shareholder, amount)
IERC20(random_Token_Address).transfer(msg.sender, number_of_Tokens)
```

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**
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	mul	Internal		

	div	Internal		
	sub	Internal		
	div	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	_verifyCallResult	Private		
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		-
	burn	External	✓	-
	swap	External	✓	-
	skim	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	IUniswapV2 Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
		Public	✓	-
	Claim_Rewards	External	✓	-
	setDistributionCriteria	External	✓	onlyToken

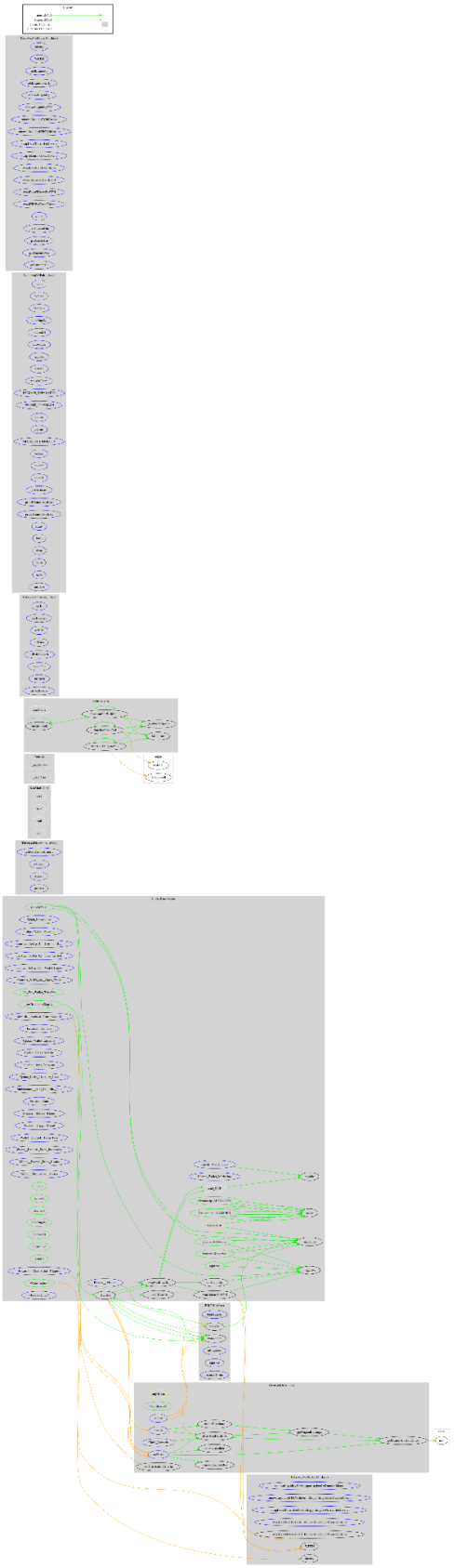
	setShare	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
GlobalReserve Coin	Implementation	Context, IERC20		
		Public	✓	-
	Token_Information	External		-
	Token_Wallet_Percents	External		-
	Contract_SetUp_01__Fees_on_Buy	External	✓	onlyOwner
	Contract_SetUp_02__Fees_on_Sell	External	✓	onlyOwner
	Contract_SetUp_03__Wallet_Limits	External	✓	onlyOwner
	Contract_SetUp_04__Open_Trade	External	✓	onlyOwner
	No_Fee_Wallet_Transfers	Public	✓	onlyOwner
	Rewards__Exclude_From_Rewards	External	✓	onlyOwner
	Rewards__Distribution_Triggers	External	✓	onlyOwner
	Rewards__Set_Gas	External	✓	onlyOwner
	Update_Wallet_Gold	External	✓	onlyOwner

	Update_Wallet_Liquidity	External	✓	onlyOwner
	Update_Wallet_Marketing	External	✓	onlyOwner
	Update_Links_Website	External	✓	onlyOwner
	Update_Links_Telegram	External	✓	onlyOwner
	Update_Links_Liquidity_Lock	External	✓	onlyOwner
	Maintenance__Add_Liquidity_Pair	External	✓	onlyOwner
	Ownership_TRANSFER	Public	✓	onlyOwner
	Ownership_RENOUNCE	Public	✓	onlyOwner
	Process__Auto	External	✓	onlyOwner
	Process__Manual	External	✓	onlyOwner
	Process__Rescue_Tokens	External	✓	onlyOwner
	Process__Trigger_Count	External	✓	onlyOwner
	Wallet__Exclude_From_Fees	External	✓	onlyOwner
	Wallet__Exclude_From_Processing	External	✓	onlyOwner
	Wallet__Exempt_From_Limits	External	✓	onlyOwner
	Wallet__Pre_Launch_Access	External	✓	onlyOwner
	owner	Public		-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-

	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	approve	Public	✓	-
	_approve	Private	✓	
	transfer	Public	✓	-
	transferFrom	Public	✓	-
	send_BNB	Internal	✓	
	getCirculatingSupply	Public		-
	_transfer	Private	✓	
	swapAndLiquify	Private	✓	
	swapTokensForBNB	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
		External	Payable	-

Inheritance Graph

Flow Graph



Summary

Global Reserve Coin contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. Global Reserve Coin is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors or critical issues. 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 12% buy and sell 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