Cyberscope

Audit Report

NexusGalaxy

February 2023

Type	BEP20
Network	BSC Testnet
Address	0xc84dC01c2f608852A490bC92D0E9FDBE3f251Ac3
Audited by	© cyberscope

Table of Contents

Review

Contract Name	NexusGalaxy
Compiler Version	v0.8.0+commit.c7dfd78e
Optimization	200 runs
Explorer	https://testnet.bscscan.com/address/0xc84dc01c2f608852a490bc92d0e9fdbe3f251ac3
Address	0xc84dc01c2f608852a490bc92d0e9fdbe3f251ac3
Network	BSC_TESTNET
Symbol	NXS
Decimals	18
Total Supply	1,000,000,000

Audit Updates

Initial Audit	08 Feb 2023

Source Files

Filename	SHA256
NexusGalaxy.sol	c3328304ab0ede8d5b549eeaa588649fe5aec1b643da82a18675aa0d25e267a0

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OCTD	Transfers Contract's Tokens	Unresolved
●	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	Unresolved
●	BC	Blacklists Addresses	Passed

OCTD - Transfers Contract's Tokens

Criticality	Minor / Informative
Location	NexusGalaxy.sol#L902
Status	Unresolved

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `withdrawTokens` function.

```
function withdrawTokens(address tokensAddrs, address to,  uint amount)
public lockWhileDistribution onlyOwner {
  IERC20(tokensAddrs).transfer(to, amount);
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

BT - Burns Tokens

Criticality	Critical
Location	NexusGalaxy.sol#L898
Status	Unresolved

Description

The contract owner has the authority to burn tokens from a specific address. The owner may take advantage of it by calling the `burnTokens` function. As a result, the targeted address will lose the corresponding tokens.

```
function burnTokens(address addr, uint amount) public onlyOwner {
  _burn(addr, amount);
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	CO	Code Optimization	Unresolved
●	DAV	Deployment Argument Validation	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

CO - Code Optimization

Criticality	Minor / Informative
Location	NexusGalaxy.sol#L882
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 statement `isLp[recentHoldersAddress]` is unnecessary since the liquidity pool address is never included in the `recentHoldersAddress`.

```
function updateAndPayRecentHolders(address addr, uint value) private
lockWhileDistribution{
    for(uint i=0; i<recentHoldersAddress.length; i++){
        if(recentHoldersAddress[i] != address(0) &&
!isLp[recentHoldersAddress[i]])
            super._transfer(address(this), recentHoldersAddress[i],
value/recentHoldersAddress.length);
    }

    if(isLp[addr]){
        return;
    }

    recentHoldersAddress[currentId] = addr;
    if(currentId == 4){
        currentId = 0;
    }else{
        currentId++;
    }
}
```

Recommendation

The team is advised to take into consideration these segments 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. It is recommended to remove redundant statements.

DAV - Deployment Argument Validation

Criticality	Minor / Informative
Location	NexusGalaxy.sol#L802
Status	Unresolved

Description

The contract does not properly sanitize the constructor arguments. These variables may produce vulnerability issues.

The `taxPercent` arguments might exceed the allowed limit of 25%. Additionally, the underlying percentages are not properly sanitized.

```
constructor(uint256 initialSupply, address _outTaxToken, address
_router, address _backToWalletAddress, address _teamAddress, address
_marketingAddress, address _developmentAddress, uint[] memory _taxes)
ERC20("Nexus Galaxy", "NXS") {
    _mint(msg.sender, initialSupply);
    outTaxToken = _outTaxToken;
    router = IUniswapV2Router02(_router);

    backToWalletAddress = _backToWalletAddress;
    teamAddress = _teamAddress;
    marketingAddress = _marketingAddress;
    developmentAddress = _developmentAddress;

    taxPercent = _taxes[0];
    backToWalletPercentage = _taxes[1];
    teamPercent = _taxes[2];
    marketingPercent = _taxes[3];
    developmentPercent = _taxes[4];
    burnPercent = _taxes[5];
    backToHoldersPercent = _taxes[6];

    toggleTaxes = true;
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

- The variable `taxPercent` shouldn't exceed the maximum acceptable value.
- The aggregation of the variables `backToWalletPercentage`, `teamPercent`, `marketingPercent`, `developmentPercent` shouldn't be set to values greater than 100.
- The aggregation of the variables `backToHoldersPercent`, `burnPercent` shouldn't be set to values greater than the `taxPercent`.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	NexusGalaxy.sol#L7,788,864
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);
bool _inTaxDistribution;
address _taxReciever,
uint _amount)

function SwapandPayTaxes(address _taxReciever, uint _amount) private{

        address[] memory path = new address[](2);
        path[0] = address(address(this));
        path[1] = address(outTaxToken);
        uint256 outputAmount = router.getAmountsOut(_amount, path)[1];

        router.swapExactTokensForTokens(_amount, outputAmount, path,
_taxReciever, block.timestamp);

    }
```

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	NexusGalaxy.sol#L861
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.

```
totalTax -= releaseAmount;
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	NexusGalaxy.sol#L800,803,804,805,806,911,914,917,920,924
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.

```
outTaxToken = _outTaxToken;
backToWalletAddress = _backToWalletAddress;
teamAddress = _teamAddress;
marketingAddress = _marketingAddress;
developmentAddress = _developmentAddress;
backToWalletAddress = addr;
teamAddress = addr;
marketingAddress = addr;
developmentAddress = addr;
outTaxToken = addr;
```

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.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	NexusGalaxy.sol#L3,101,150,177,262,347,377,760
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity >=0.6.2;
pragma solidity ^0.8.0;
pragma solidity 0.8.0;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	NexusGalaxy.sol#L150,177,262,347,377
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	NexusGalaxy.sol#L899
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(tokensAddrs).transfer(to, 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

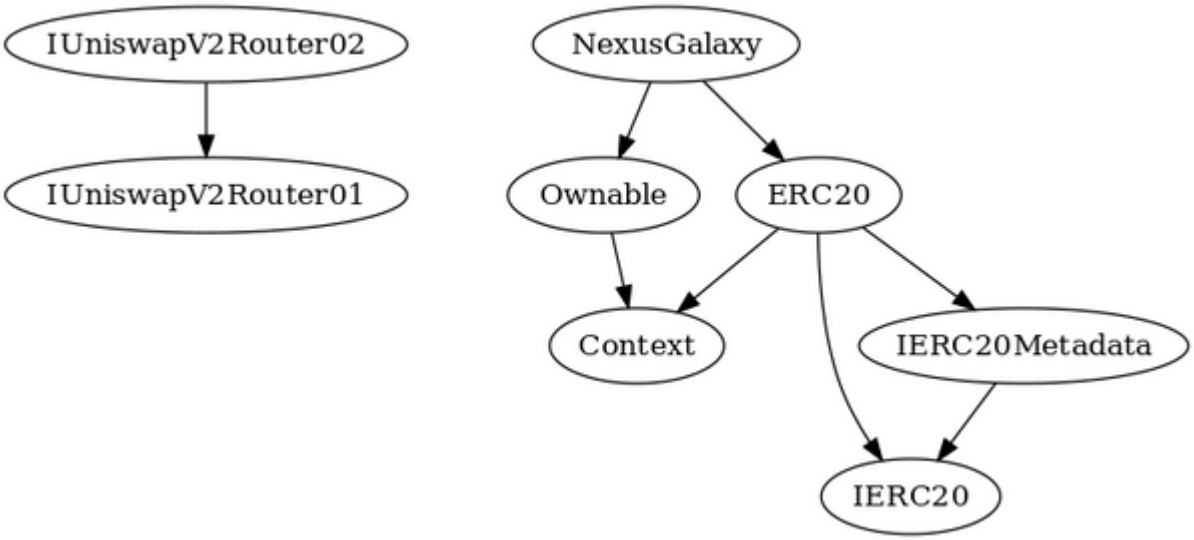
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	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	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-

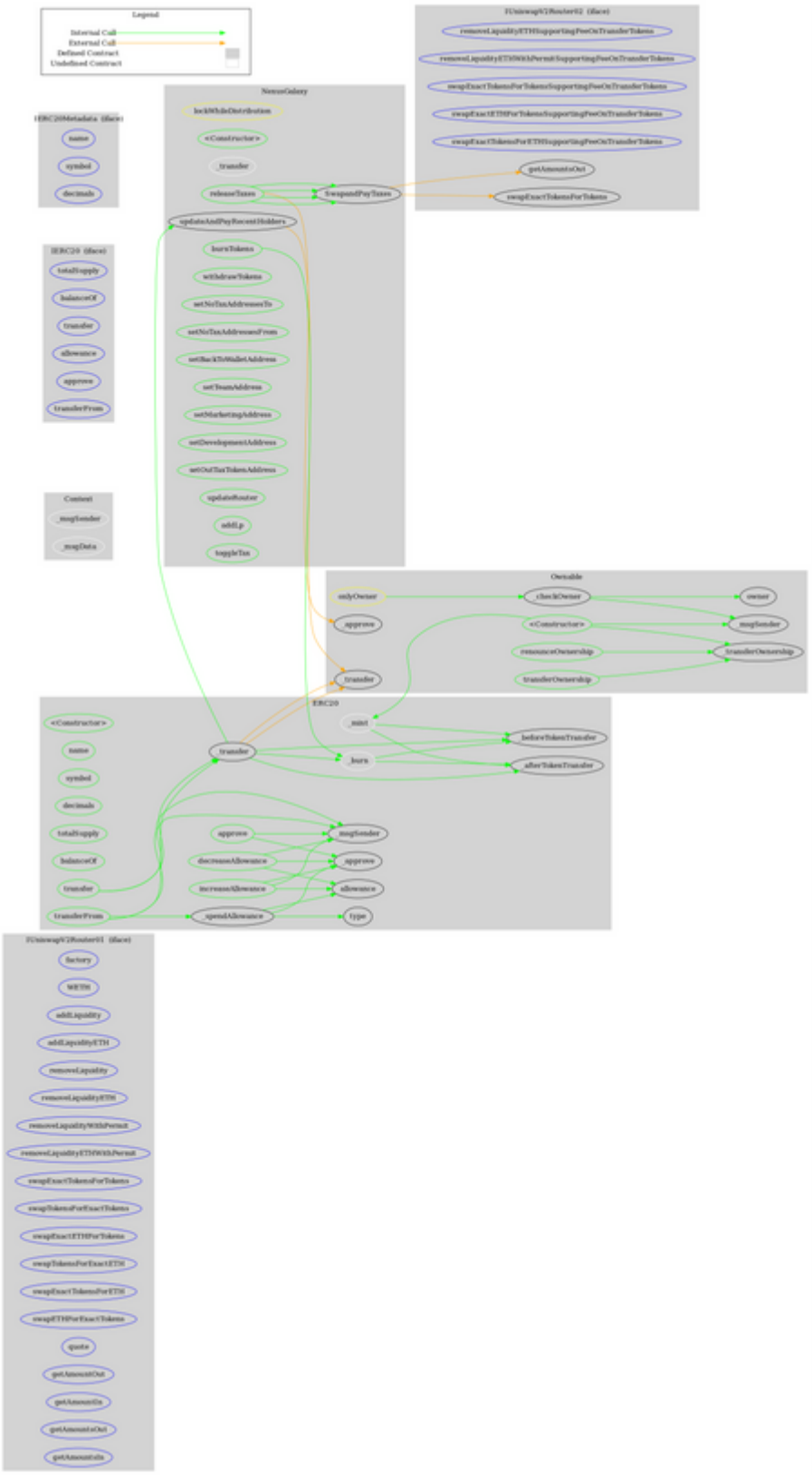
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
NexusGalaxy	Implementation	ERC20, Ownable		
		Public	✓	ERC20
	_transfer	Internal	✓	
	releaseTaxes	Public	✓	lockWhileDistribution onlyOwner

	SwapandPayTaxes	Private	✓	
	updateAndPayRecentHolders	Private	✓	lockWhileDistribution
	burnTokens	Public	✓	onlyOwner
	withdrawTokens	Public	✓	lockWhileDistribution onlyOwner
	setNoTaxAddressesTo	Public	✓	onlyOwner
	setNoTaxAddressesFrom	Public	✓	onlyOwner
	setBackToWalletAddress	Public	✓	onlyOwner
	setTeamAddress	Public	✓	onlyOwner
	setMarketingAddress	Public	✓	onlyOwner
	setDevelopmentAddress	Public	✓	onlyOwner
	setOutTaxTokenAddress	Public	✓	onlyOwner
	updateRouter	Public	✓	onlyOwner
	addLp	Public	✓	onlyOwner
	toggleTax	Public	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
NexusGalaxy
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData
lockWhileDistribution
<Constructor>
_transfer
releaseTaxes
updateAndPayRecentHolders
burnTokens
withdrawTokens
setNoTaxAddressesTo
setNoTaxAddressesFrom
setBackToWalletAddress
setTeamAddress
setMarketingAddress
setDevelopmentAddress
setOutTaxTokensAddress
updateRouter
addLp
toggleTax
IUniswapV2Router02 (iface)
removeLiquidityETHSupportingFeeOnTransferTokens
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
getAmountsOut
swapExactTokensForTokens
Ownable
onlyOwner
checkOwner
owner
approve
<Constructor>
msgSender
renounceOwnership
transferOwnership
transfer
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
transferFrom
_mint
_burn
beforeTokenTransfer
afterTokenTransfer
approve
decreaseAllowance
increaseAllowance
_spendAllowance
msgSender
approve
allowance
type
IUniswapV2Router01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
swapTokensForExactTokens
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn

Summary

There are some functions that can be abused by the owner like drain the contract's tokens, manipulate the fees and burn tokens from any address. if the contract owner abuses the burn functionality, then the users could lost their tokens. 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