Cyberscope

Audit Report

Daily FOMO

May 2023

Network	BSC
Address	0x93b2186ADbc3f3ea85EA4c6AC0FB5017975A7c07
Audited by	© cyberscope

Table of Contents

Review

Contract Name	FOMO
Compiler Version	v0.8.2+commit.661d1103
Optimization	200 runs
Explorer	https://bscscan.com/address/0x93b2186adbc3f3ea85ea4c6ac0fb5017975a7c07
Address	0x93b2186adbc3f3ea85ea4c6ac0fb5017975a7c07
Network	BSC
Symbol	$FOMO
Decimals	18
Total Supply	1,000,000,000

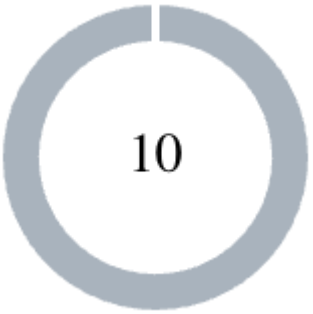
Audit Updates

Initial Audit	20 May 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249a a4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/token/ERC20/ERC20.sol	5031430cc2613c32736d598037d3075985 a2a09e61592a013dbd09a5bc2041b8
@openzeppelin/contracts/token/ERC20/extensions /IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166 689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db800 3d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a2 3a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/math/SafeMath.sol	0dc33698a1661b22981abad8e5c6f5ebca 0dfe5ec14916369a2935d888ff257a
contracts/FOMO.sol	bc02954629be1baa963dcc40450a3aa84a 15942eee1789a685503d7c82abf5f2

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	10

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

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
	CR	Code Repetition	Unresolved
	PTRP	Potential Transfer Revert Propagation	Unresolved
	PVC	Price Volatility Concern	Unresolved
	IDI	Immutable Declaration Improvement	Unresolved
	RSML	Redundant SafeMath Library	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L09	Dead Code Elimination	Unresolved
	L13	Divide before Multiply Operation	Unresolved
	L18	Multiple Pragma Directives	Unresolved
	L19	Stable Compiler Version	Unresolved

CR - Code Repetition

Criticality	Minor / Informative
Location	contracts/FOMO.sol#L235
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

The fee collection method repeats the same code segments for both buy and sell statements.

```
if(sell && saleFeeEnable)
{
    uint256 newLiquidityFee = amount.mul(liquidityFee[1]).div(100);
    uint256 newMarketingFee = amount.mul(marketingFee[1]).div(100);
    uint256 newTeamFee = amount.mul(teamFee[1]).div(100);

    liquidityFeeTotal += newLiquidityFee;
    marketingFeeTotal += newMarketingFee;
    teamFeeTotal += newTeamFee;
    return (newMarketingFee.add(newTeamFee).add(newLiquidityFee));
}
else if(buy && buyFeeEnable)
{
    uint256 newLiquidityFee = amount.mul(liquidityFee[0]).div(100);
    uint256 newMarketingFee = amount.mul(marketingFee[0]).div(100);
    uint256 newTeamFee = amount.mul(teamFee[0]).div(100);

    liquidityFeeTotal += newLiquidityFee;
    marketingFeeTotal += newMarketingFee;
    teamFeeTotal += newTeamFee;
    return (newMarketingFee.add(newTeamFee).add(newLiquidityFee));
}
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	contracts/FOMO.sol#L205
Status	Unresolved

Description

The contract sends funds to a `marketingWallet` as part of the transfer flow. This address can either be a wallet address or a contract. If the address belongs to a contract then it may revert from incoming payment. As a result, the error will propagate to the token's contract and revert the transfer.

```
payable(marketing).transfer(marketingPart);
payable(team).transfer(teamPart);
```

Recommendation

The contract should tolerate the potential revert from the underlying contracts when the interaction is part of the main transfer flow. This could be achieved by not allowing set contract addresses or by sending the funds in a non-revertable way.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	contracts/FOMO.sol#L180
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThreshold` 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.

```
bool canSwap = contractTokenBalance >= swapThreshold;
```

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 total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	@openzeppelin/contracts/token/ERC20/ERC20.sol#L55,56contracts/FOMO.sol#L81,82
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
_name
_symbol
router
pair
```

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.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	@openzeppelin/contracts/utils/math/SafeMath.solcontracts/FOMO.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/FOMO.sol#L40,263,278,290
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.

```
require(newWallet != address(0), "setTe
```

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.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	@openzeppelin/contracts/token/ERC20/ERC20.sol#L280
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 _burn(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: burn from the zero
address");

        _beforeTokenTransfer(account, address(0), amount);

        uint256 accountBalance = _balances[account];
...
        }
        _totalSupply -= amount;

        emit Transfer(account, address(0), amount);

        _afterTokenTransfer(account, address(0), amount);
    }
```

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	contracts/FOMO.sol#L184,195,201
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.

```
+= newTeamFee;
            return (newMarketingFee
ove(address(this), address(router), FOMOAmount);
        router.addLiquidi
```

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.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	contracts/FOMO.sol#L28@openzeppelin/contracts/utils/math/SafeMath.sol#L4@openzeppelin/contracts/utils/Context.sol#L4@openzeppelin/contracts/token/ERC20/IERC20.sol#L4@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#L4@openzeppelin/contracts/token/ERC20/ERC20.sol#L4@openzeppelin/contracts/access/Ownable.sol#L4
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 ^0.8.0;
0**18));
        _transferO
```

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	@openzeppelin/contracts/utils/math/SafeMath.sol#L4@openzeppelin/contracts/utils/Context.sol#L4@openzeppelin/contracts/token/ERC20/IERC20.sol#L4@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#L4@openzeppelin/contracts/token/ERC20/ERC20.sol#L4@openzeppelin/contracts/access/Ownable.sol#L4
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.

Functions Analysis

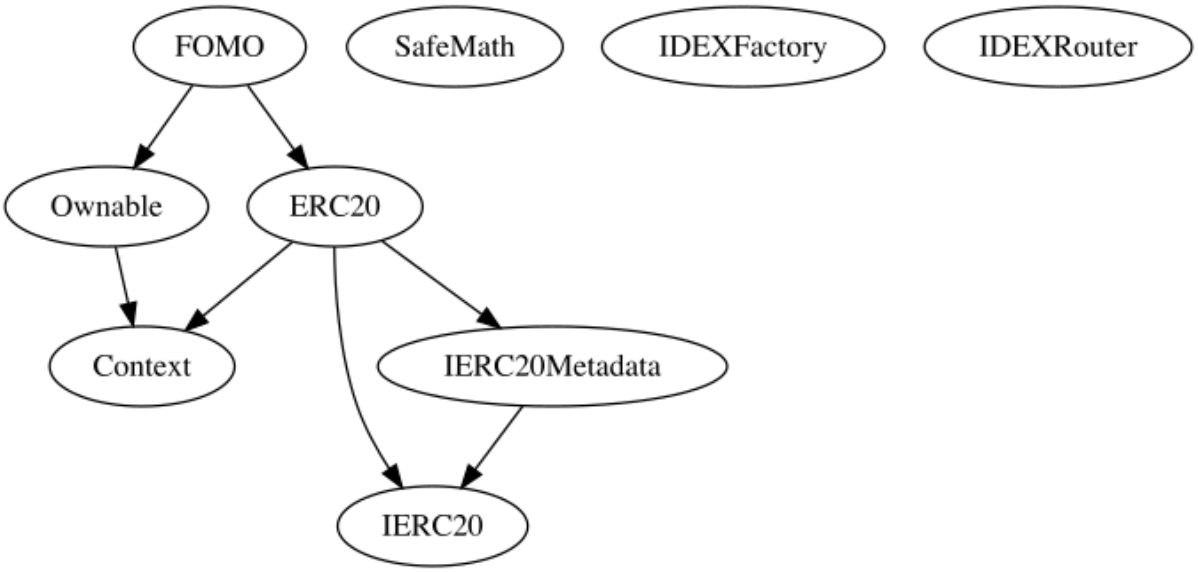
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ERC20	Implementation	Context, 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	✓	
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-

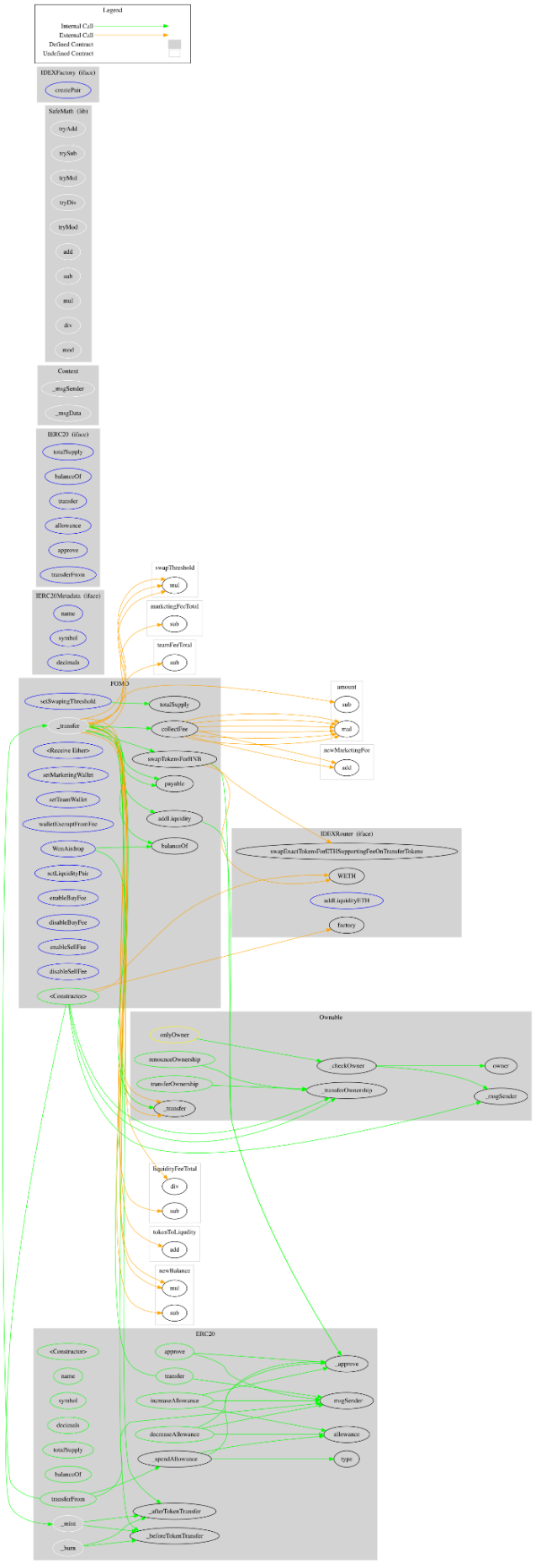
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
FOMO	Implementation	ERC20, Ownable		
		Public	✓	ERC20

		External	Payable	-
	setMarketingWallet	External	✓	onlyOwner
	setTeamWallet	External	✓	onlyOwner
	walletExemptFromFee	External	✓	onlyOwner
	setSwapingThreshold	External	✓	onlyOwner
	setLiquidityPair	External	✓	onlyOwner
	enableBuyFee	External	✓	onlyOwner
	disableBuyFee	External	✓	onlyOwner
	enableSellFee	External	✓	onlyOwner
	disableSellFee	External	✓	onlyOwner
	_transfer	Internal	✓	
	collectFee	Private	✓	
	swapTokensForBNB	Private	✓	
	addLiquidity	Private	✓	
	WenAirdrop	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IDEXFactory (iface)
createPair
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
Context
_msgSender
_msgData
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IERC20Metadata (iface)
name
symbol
decimals
swapThreshold
mul
marketingFeeTotal
sub
teamFeeTotal
sub
FOMO
setSwapingThreshold
_transfer
<Receive Ether>
serMarketingWallet
serTeamWallet
walletExemptFromFee
WonAirdrop
setLiquidityPair
enableBuyFee
disableBuyFee
enableSellFee
disableSellFee
<Constructor>
totalSupply
collectFee
swapTokensForBNB
payable
addLiquidity
balanceOf
amount
sub
mul
newMarketingFee
add
IDEXRouter (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
addLiquidityETH
factory
Ownable
onlyOwner
renounceOwnership
transferOwnership
_transfer
checkOwner
owner
transferOwnership
_msgSender
liquidityFeeTotal
div
sub
tokenToLiquidity
add
newBalance
mul
sub
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transferFrom
_mint
_burn
approve
transfer
increaseAllowance
decreaseAllowance
spendAllowance
_afterTokenTransfer
_beforeTokenTransfer
_approve
msgSender
allowance
type

Summary

Token is an interesting project that has a friendly and growing community. The Smart Daily FOMO contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Daily FOMO is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error 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. The fees are fixed to 5% in buys and 6% in sales.

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