# Cyberscope

# Audit Report
# AI Minion

April 2023

Network    BSC

Address    0x3ed4246eF2052e66eE2594Ac08f1ed591279d914

Audited by  © cyberscope

# Table of Contents

# Review

| Contract Name | AIM |
|---|---|
| Compiler Version | v0.8.18+commit.87f61d96 |
| Optimization | 200 runs |
| Explorer | https://bscscan.com/address/0x3ed4246ef2052e66ee2594ac08f1ed591279d914 |
| Address | 0x3ed4246ef2052e66ee2594ac08f1ed591279d914 |
| Network | BSC |
| Symbol | AIM |
| Decimals | 9 |
| Total Supply | 25.000.000.000.000 |

## Audit Updates

| | |
|---|---|
| **Initial Audit** | 10 Apr 2023<br><br>https://github.com/cyberscope-io/audits/blob/main/2-aim/v1/audit.pdf |
| **Corrected Phase 2** | 13 Apr 2023<br><br>https://github.com/cyberscope-io/audits/blob/main/2-aim/v2/audit.pdf |
| **Corrected Phase 3** | 18 Apr 2023<br><br>https://github.com/cyberscope-io/audits/blob/main/2-aim/v3/audit.pdf |
| **Corrected Phase 4** | 21 Apr 2023<br><br>https://github.com/cyberscope-io/audits/blob/main/2-aim/v4/audit.pdf |
| **Corrected Phase 5** | 21 Apr 2023 |

## Source Files

| Filename | SHA256 |
|---|---|
| **AIM.sol** | f940a6189b4c44293b3c92791088b63bd735a8d4ac25c2cb1709158d1fbefdf7 |

# Findings Breakdown

| | | |
|---|---|---|
| ● Critical | 0 |
| ● Medium | 0 |
| ● Minor / Informative | 9 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|---|---|---|---|---|
| ● Critical | 0 | 0 | 0 | 0 |
| ● Medium | 0 | 0 | 0 | 0 |
| ● Minor / Informative | 9 | 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 |
|:---:|:---|:---|:---|
| ● | PVC | Price Volatility Concern | Unresolved |
| ● | L02 | State Variables could be Declared Constant | Unresolved |
| ● | 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 |
| ● | L19 | Stable Compiler Version | Unresolved |
| ● | L20 | Succeeded Transfer Check | Unresolved |

# PVC - Price Volatility Concern

| Criticality | Minor / Informative |
|---|---|
| Location | AIM.sol#L488 |
| Status | Unresolved |

## Description

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

```
function setMinTokenNumberToSell(uint256 _amount) public onlyOwner {
    minTokenNumberToSell = _amount;
}
```

## 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.

## L02 - State Variables could be Declared Constant

| Criticality | Minor / Informative |
| --- | --- |
| Location | AIM.sol#L38,39,284,288,289,290,296,298 |
| 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 payable private _previousOwner
uint256 private _lockTime
uint256 private _tTotal = 25e12 * 1e9
string private _name = "AI MINION"
string private _symbol = "AIM"
uint8 private _decimals = 9
address public burnAddress = 0x000000000000000000000000000000000000dEaD
uint256 public maxFee = 15
```

## 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 | AIM.sol#L101,102,119,139,484,489,494,498,503,509,516,522,629,630,631,632,633,652,653,654,655,656,675,676,677,678,679 |
| 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);
uint256 _amount
bool _state
address payable _marketWallet
address payable _charityWallet
IPancakeRouter02 _router
address _pair
uint _amount
IBEP20 _token
uint256 _redistributionFee
uint256 _liquidityFee

...
```

## 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 | AIM.sol#L38,39 |
| 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.

```
address payable private _previousOwner
uint256 private _lockTime
```

## 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** | AIM.sol#L485,635,658,681 |
| **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.

```
minTokenNumberToSell = _amount
redistributionFeeOnBuying = _redistributionFee
redistributionFeeOnSelling = _redistributionFee
redistributionFee = _redistributionFee
```

## 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** | AIM.sol#L907 |
| **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.

```solidity
function swapETHForTokens(
        address routerAddress,
        address recipient,
        uint256 ethAmount
    ) internal {
        IPancakeRouter02 pancakeRouter = IPancakeRouter02(routerAddress);
...
        pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
ethAmount}(
            0, // accept any amount of BNB
            path,
            address(recipient),
            block.timestamp + 300
        );
    }
```

## 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.

## Recommendation

## L13 - Divide before Multiply Operation

| Criticality | Minor / Informative |
| --- | --- |
| Location | AIM.sol#L537,538,561,562,570,571,578,579,587,588 |
| 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 a loss of prediction.

```
uint256 burnFee = tAmount*(_currentautoburnFee)/(1e2)
uint256 rBurnFee = burnFee*(currentRate)
```

## 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.

## L19 - Stable Compiler Version

| Criticality | Minor / Informative |
|---|---|
| Location | AIM.sol#L1 |
| 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 ^0.8.17;
```

## 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 | AIM.sol#L524 |
| 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.

```
_token.transfer(msg.sender,_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 |
| | | | | |
| **IBEP20** | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| **Context** | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| **Ownable** | Implementation | Context | | |
| | | Public | ✓ | - |
| | owner | Public | | - |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | | | | |

| IPancakeFactory | Interface | | | |
|---|---|---|---|---|
| | feeTo | External | | - |
| | feeToSetter | External | | - |
| | getPair | External | | - |
| | allPairs | External | | - |
| | allPairsLength | External | | - |
| | createPair | External | ✓ | - |
| | setFeeTo | External | ✓ | - |
| | setFeeToSetter | External | ✓ | - |
| | | | | |
| IPancakePair | 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 | | - |
| | mint | External | ✓ | - |
| | burn | External | ✓ | - |
| | swap | External | ✓ | - |
| | skim | External | ✓ | - |
| | sync | External | ✓ | - |
| | initialize | External | ✓ | - |
| | | | | |
| **IPancakeRouter01** | 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 | | - |
| | | | | |
| **IPancakeRouter02** | Interface | IPancakeRouter01 | | |
| | removeLiquidityETHSupportingFeeOnTransferTokens | External | ✓ | - |
| | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactTokensForTokensSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactETHForTokensSupportingFeeOnTransferTokens | External | Payable | - |
| | swapExactTokensForETHSupportingFeeOnTransferTokens | External | ✓ | - |
| | | | | |

| AIM | Implementation | Context, IBEP20, Ownable | | |
|---|---|---|---|---|
| | | Public | ✓ | - |
| | name | External | | - |
| | symbol | External | | - |
| | decimals | External | | - |
| | totalSupply | External | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |
| | isExcludedFromReward | Public | | - |
| | totalFees | Public | | - |
| | deliver | Public | ✓ | - |
| | reflectionFromToken | Public | | - |
| | tokenFromReflection | Public | | - |
| | excludeFromReward | Public | ✓ | onlyOwner |
| | includeInReward | External | ✓ | onlyOwner |
| | excludeFromFee | Public | ✓ | onlyOwner |
| | includeInFee | Public | ✓ | onlyOwner |
| | setMinTokenNumberToSell | Public | ✓ | onlyOwner |

| | | | | |
|---|---|---|---|---|
| setSwapAndLiquifyEnabled | Public | ✓ | | onlyOwner |
| setReflectionFees | External | ✓ | | onlyOwner |
| setmarketWallet | External | ✓ | | onlyOwner |
| setcharityWallet | External | ✓ | | onlyOwner |
| setRoute | External | ✓ | | onlyOwner |
| withdrawBNB | External | ✓ | | onlyOwner |
| withdrawToken | External | ✓ | | onlyOwner |
| | External | Payable | | - |
| totalFeePerTx | Internal | | | |
| _reflectFee | Private | ✓ | | |
| _getRate | Private | | | |
| _getCurrentSupply | Private | | | |
| _takeLiquidityPoolFee | Internal | ✓ | | |
| _takeMarketFee | Internal | ✓ | | |
| _takeCharityFee | Internal | ✓ | | |
| _takeBurnFee | Internal | ✓ | | |
| removeAllFee | Private | ✓ | | |
| setBuyFee | Private | ✓ | | |
| setSellFee | Private | ✓ | | |
| setNormalFee | Private | ✓ | | |
| setBuyFeePercent | External | ✓ | | onlyOwner |
| setSellFeePercent | External | ✓ | | onlyOwner |
| setNormalFeePercent | External | ✓ | | onlyOwner |

| | | | | |
|---|---|---|---|---|
| | isExcludedFromFee | Public | | - |
| | _approve | Private | ✓ | |
| | _transfer | Private | ✓ | |
| | _tokenTransfer | Private | ✓ | |
| | _transferStandard | Private | ✓ | |
| | _transferToExcluded | Private | ✓ | |
| | _transferFromExcluded | Private | ✓ | |
| | _transferBothExcluded | Private | ✓ | |
| | swapAndLiquify | Private | ✓ | |
| | | | | |
| **Utils** | Library | | | |
| | swapTokensForEth | Internal | ✓ | |
| | swapETHForTokens | Internal | ✓ | |
| | addLiquidity | Internal | ✓ | |

# Inheritance Graph
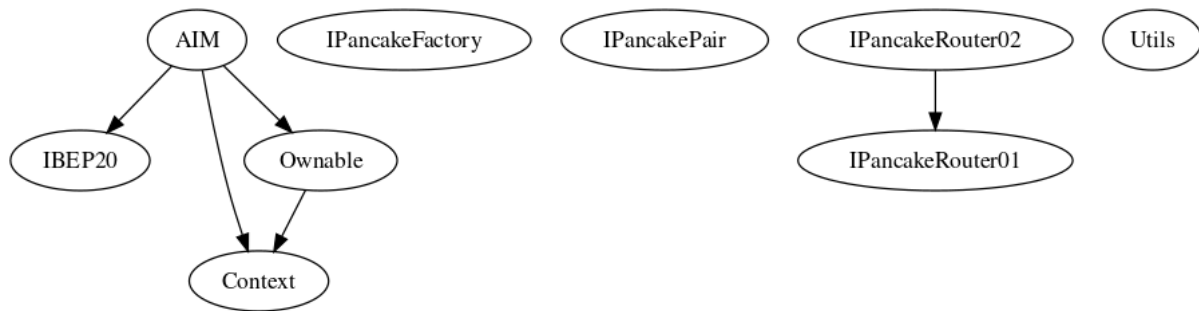
# Flow Graph
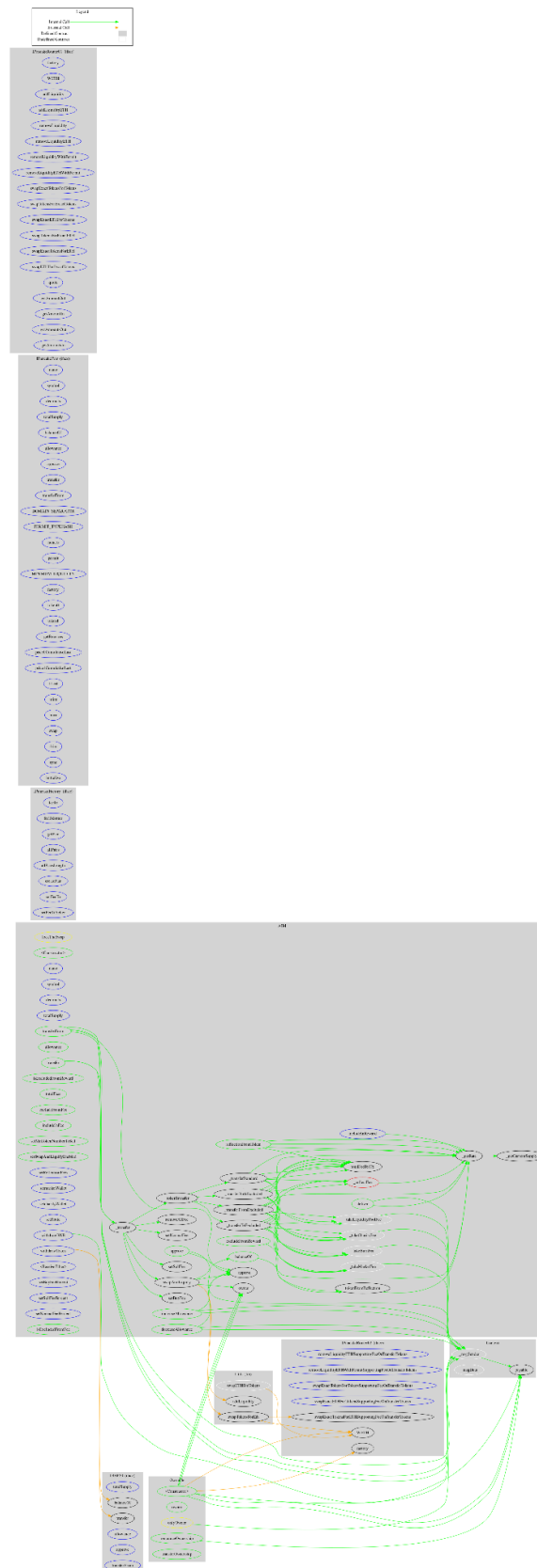
# Summary

AI Minion contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. AI Minion 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 15% buy, sell, and transfer 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