Cyberscope

Audit Report

Axondao

September 2023

SHA256 9e3981c28290febdeb5977411fa8e7a5445531010e750007a6bb4fb2323f895d

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	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Critical	IFI	Incorrect Functions Implementation	Unresolved
Minor / Informative	EIS	Excessively Integer Size	Unresolved
Minor / Informative	EPC	Existing Pair Creation	Unresolved
Minor / Informative	MVN	Misleading Variables Naming	Unresolved
Minor / Informative	RSML	Redundant SafeMath Library	Unresolved
Minor / Informative	L04	Conformance to Solidity Naming Conventions	Unresolved
Minor / Informative	L07	Missing Events Arithmetic	Unresolved
Minor / Informative	L13	Divide before Multiply Operation	Unresolved
Minor / Informative	L14	Uninitialized Variables in Local Scope	Unresolved
Minor / Informative	L16	Validate Variable Setters	Unresolved
Minor / Informative	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Testing Deploy	https://testnet.bscscan.com/address/0x133fd8e3b291a810c78ec4204c121084c4663e4c

Audit Updates

Initial Audit	20 Sep 2023 https://github.com/cyberscope-io/audits/blob/main/axgt/v1/audit.pdf
Corrected Phase 2	26 Sep 2023 https://github.com/cyberscope-io/audits/blob/main/axgt/v2/audit.pdf
Corrected Phase 3	10 Oct 2023

Source Files

Filename	SHA256
AXGT.sol	9e3981c28290febdeb5977411fa8e7a5445531010e750007a6bb4fb2323f895d

Findings Breakdown

Critical	1
Medium	0
Minor / Informative	10

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	10	0	0	0

IFI - Incorrect Functions Implementation

Criticality	Critical
Location	contracts/AXGT.sol#L846,853,860
Status	Unresolved

Description

The contract contains functions to set specific wallet addresses for administrative and funding purposes. However, there are discrepancies in the implementation. Specifically, the `setFundWAllet` function, which is presumably intended to set the `fundWallet` address, is instead setting the `adminWallet` address. This can lead to unintended consequences and potential mismanagement of funds.

Additionally, in the `changeOwner` function, the `_address` passed as a parameter is firstly set not to be excluded from fees and then immediately set to be excluded. This means that the previous owner's address, which should ideally be excluded from fees after transferring ownership, remains unaffected.

```
    function setadminWallet (address _address) external onlyOwner {
        require(_address != address(0) && adminWallet != _address,
"Invalid or same address provided");
        _isExcluded[adminWallet] = false;
        adminWallet = _address;
        _isExcluded[adminWallet] = true;
    }

    function setFundWAllet (address _address) external onlyOwner {
        require(_address != address(0) && fundWallet != _address,
"Invalid or same address provided");
        _isExcluded[fundWallet] = false;
        adminWallet = _address;
        _isExcluded[fundWallet] = true;
    }

    function changeOwner (address _address) external onlyOwner {
        require(_address != address(0) && owner() != _address, "Invalid
or same address provided");
        _isExcluded[_address] = false;
        transferOwnership(_address);
        _isExcluded[_address] = true;
    }
```

Recommendation

It is recommended to reconsider the code implementation. Specifically, the team is advised to modify the `setFundWAllet` function to correctly set the `fundWallet` variable with the provided `_address` instead of the `adminWallet`. Additionally, in the `changeOwner` function, it is recommended to change the `_isExcluded[_address]` line to `_isExcluded[_owner]` to ensure that the previous owner's address is correctly set to be not excluded from fees after transferring ownership.

EIS - Excessively Integer Size

Criticality	Minor / Informative
Location	AXGT.sol#L682,693,883
Status	Unresolved

Description

The contract is using a bigger unsigned integer data type that the maximum size that is required. By using an unsigned integer data type larger than necessary, the smart contract consumes more storage space and requires additional computational resources for calculations and operations involving these variables. This can result in higher transaction costs, longer execution times, and potential scalability bottlenecks.

Specifically the variables `liquidity`, `FeeadminFee` and `fundFee` can be represent by using a `uint16`, since the maximum possible vallue is `2500`. Additionally, `PercentageOfMaximumTokensToAccumate` is a constant with a value of `10`. This can be represented using `uint8` since `uint8` can hold values between 0 and 255.

```
uint256 private constant PercentageOfMaximumTokensToAccumate =
10;
...
uint256 public liquidityFee    = 133;
uint256 public adminFee        = 133;
uint256 public fundFee         = 133;

    function setFee (uint256 _newLPFee, uint256 _newadminFee,
uint256 _newfundFee) external onlyOwner{
        require (0 < _newLPFee + _newadminFee + _newfundFee &&
_newLPFee + _newadminFee + _newfundFee< 2500, "need to smaller
than 100%");
        ...
    }
```

Recommendation

To address the inefficiency associated with using an oversized unsigned integer data type, it is recommended to accurately determine the required size based on the range of values the

variable needs to represent. The team is advised to use a `uint16` data type for the variables `liquidityFee` , `adminFee` and `fundFee` , and a `uint8` for the `PercentageOfMaximumTokensToAccumate` varriable.

EPC - Existing Pair Creation

Criticality	Minor / Informative
Location	AXGT.sol#L721
Status	Unresolved

Description

The contract contains a function that does not handle the scenario where a pair already exists prior to its execution. If a pair for the given tokens has already been established, the `createPair` function will revert and not proceed with the creation of a new pair. As a result, if a pair has been previously set up before the function is invoked, the contract will encounter an error when trying to call the `createPair` function. This will prevent the successful execution, essentially leading the function to revert.

```
   function setUniswapV2Router(address _uniswapV2Router)
external onlyOwner{
        IUniswapV2Router02 _UniswapV2Router =
IUniswapV2Router02(_uniswapV2Router);
        address _UniswapV2Pair =
IUniswapV2Factory(_UniswapV2Router.factory())
                .createPair(address(this),
_UniswapV2Router.WETH());
        UniswapV2Router = _UniswapV2Router;
        UniswapV2Pair = _UniswapV2Pair;
        _isLimitExcluded[_UniswapV2Pair] = true;
```

Recommendation

To mitigate the risks associated with attempting to create an already existing pair, it is recommended to implement a check to determine whether the pair already exists before proceeding to create a new pair. This can be achieved by utilizing the getPair function of the Factory contract to retrieve the address of the pair contract for the specified tokens. If the address returned by the getPair function is the zero address, it indicates that the pair does not exist, and the contract can proceed with the createPair function. Conversely, if a non-zero address is returned, it indicates that the pair already exists, and the createPair function will revert.

MVN - Misleading Variables Naming

Criticality	Minor / Informative
Location	AXGT.sol#L681
Status	Unresolved

Description

Variables can have misleading names if their names do not accurately reflect the value they contain or the purpose they serve. The contract uses some variable names that are too generic or do not clearly convey the information stored in the variable. Misleading variable names can lead to confusion, making the code more difficult to read and understand.

Specifiacally, the contract is using the variable `ONE` to represent the standard number of decimals of the token, which is `10**18`. Using the name `ONE` can indeed be misleading, as it doesn't convey the purpose of the variable.

```
uint256 private constant ONE = 10**18;
```

Recommendation

It's always a good practice for the contract to contain variable names that are specific and descriptive. The team is advised to keep in mind the readability of the code. It is recommended to rename the variable to a more descriptive name. A common convention is to use DECIMALS or TOKEN_DECIMALS to represent this value.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	contracts/AXGT.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/AXGT.sol#L118,119,136,172,678,679,682,697,698,729,738,846,853,861,868,873,878,884
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);
IUniswapV2Router02 public UniswapV2Router;
address public UniswapV2Pair;
uint256 private constant PercentageOfMaximumTokensToAccumate =
10;
mapping (address => bool)     public  _isExcluded;
mapping (address => bool)     public  _isLimitExcluded;
...
```

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	contracts/AXGT.sol#L843,881,886
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.

```
swapTokensAtAmount = amount;
...
balanceLimit = _limit;
...
liquidityFee = _newLPFee;
adminFee = _newadminFee;
fundFee = _newfundFee;
```

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.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	contracts/AXGT.sol#L788
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.

```
uint256 half = tokens / 2;
```

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.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	contracts/AXGT.sol#L767
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
bool takeFee;
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/AXGT.sol#L734
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.

```
UniswapV2Pair = _UniswapV2Pair;
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/AXGT.sol#L18
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 ^0.8.7;
```

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**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Context	Implementation			
	_msgSender	Internal		
ReentrancyGuard	Implementation			
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-

	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
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		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-

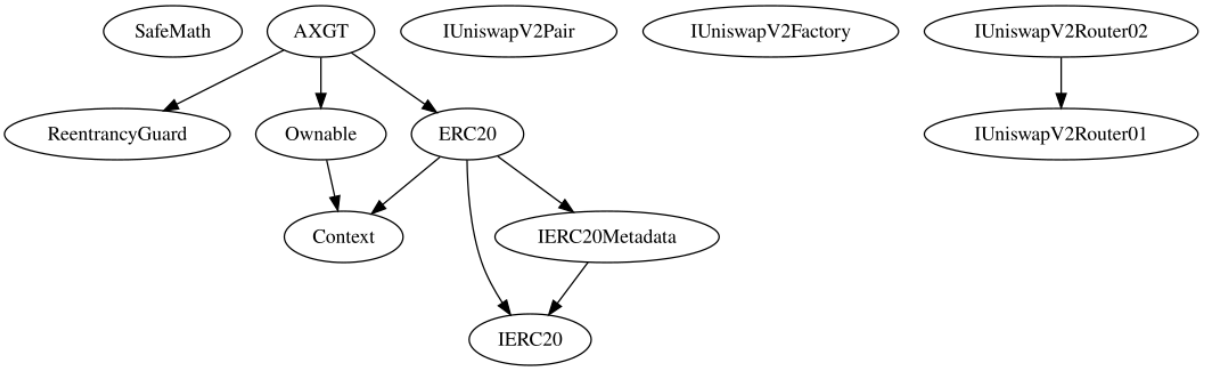
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-

	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
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		-

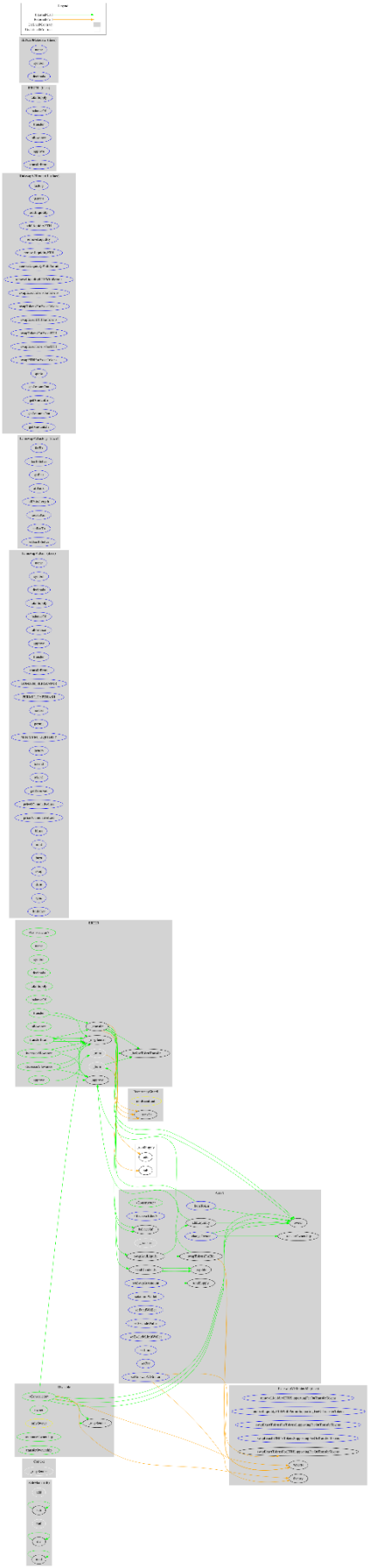
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
AXGT	Implementation	ERC20, Ownable, ReentrancyGuard		
		Public	✓	ERC20
		External	Payable	-
	setUniswapV2Router	External	✓	onlyOwner
	_burnToken	External	✓	onlyOwner
	_transfer	Internal	✓	
	swapAndLiquify	Private	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	sendDividends	Private	✓	nonReentrant

	setSwapAtAmount	External	✓	onlyOwner
	setadminWallet	External	✓	onlyOwner
	setFundWAllet	External	✓	onlyOwner
	changeOwner	External	✓	onlyOwner
	setExcludeWallet	External	✓	onlyOwner
	setExcludeLimitWallet	External	✓	onlyOwner
	setLimit	External	✓	onlyOwner
	setFee	External	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

Axondao contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Axondao 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. There is also a limit of max 25% 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