Cyberscope

Audit Report

Tea-Fi

October 2024

Files ProxyTrade, SynthToken, SynthTokenFactory, TeaFiRelayer, TeaFiTrustedForwarder, Authorizable, Permitable, PermitPayable, DecimalsCorrectionLib

Audited by © cyberscope

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Audit Scope

The current contract heavily relies on the `trustedForwarder_` external contract, to perform crucial functionalities. While this dependency enables important functionality, any interactions with this external contract should be carefully reviewed and handled, as it is beyond the scope of this audit. The behavior and security of this external contract have not been assessed as part of this audit, and any interactions with it should be treated with caution to mitigate potential risks.

Audit Updates

Initial Audit	30 Sep 2024 https://github.com/cyberscope-io/audits/blob/main/1-tea/v1/audit.pdf
Corrected Phase 2	25 Oct 2024

Source Files

TeaFiTrustedForwarder.sol	58a620464dce035c70b7dbda0c05c4939ccc6d33c06625a1f36d1f2990044d28
TeaFiRelayer.sol	b49f0b59405958e3e86fbc9fdbad86d598b84cd9e3ff5ed69bc083bd36056e43
SynthTokenFactory.sol	9b7c0b2aec72bcbf7e182f91d17ed41ef4a94474afc1f444d7103916bd79cf7a
SynthToken.sol	6411648bb5994c1aa4eb18dd8397532004102b79d553af5ac4cf0ef8e6c0c3ec
ProxyTrade.sol	c0739b3a8001e2767d70c075c3fdfedcbfc04a67e87e1ebd6f2121aa9c9cdfd5

components/Permitable.sol	65ce3ec1b3f1b95ea846fedf7c21eaf18225dd890ef00a04b3bbc2565940a1ea
components/PermitPayable.sol	074e2cba7c9c75bbe058f8b4342a55585881c8339ca37a7c7e981bc708c6bef4
components/DecimalsCorrectionLib.sol	366fddecaad707407227e2528418ecf3615bc4c503c8d6236fca66b6f7031010
components/Authorizable.sol	191beaadf6c7cf4cbad63590072285ec4aa7ca7b7591947730a9de1ec555339e

Overview

ProxyTrade Contract

The `ProxyTrade` contract enables token swaps via the 1inch DEX aggregator, facilitating secure token approvals and transfers. It serves as a proxy for executing swaps, managing token authorizations, and supporting meta-transactions with the Permit2 system to enhance security.

`makePublicSwap` Functionality

Enables users to swap tokens by interacting with 1inch, handling token reception, allowance checks, and execution of the swap, with resulting tokens sent to the user's wallet.

`makePublicSwapWithPermit` & `makePublicSwapWithTwoPermits` Functionalities

These functions extend swaps by using Permit2 and token-specific permits for off-chain authorization. They verify permit signatures, handle token reception and allowance updates, and execute swaps through 1inch.

`relayToDop` Functionalities

Provides token relays with optional Permit2 and token-specific permits for secure approval and transfer, supporting actions with trusted DOP-related addresses (`dopRelayer` and `dopSmartWallet`).

Meta-Transactions and Synth Tokens

Integrates ERC2771Context to enable meta-transactions, allowing gasless interactions. Additionally, it supports synth tokens, handling underlying assets and conversions when required during swaps.

TeaFiRelayer Contract

The `TeaFiRelayer` contract manages gasless meta-transactions by forwarding calls through a trusted forwarder. It leverages Permit2 for handling token allowances and allows operators to execute calls on behalf of users, with strict access control. Role-based permissions for operators and token limit managers ensure secure relay of calls and payments.

`relayCall` Functionality

Allows operators to relay a meta-transaction on behalf of a user by first processing the payment and then forwarding the request to the trusted forwarder for execution.

`relayCallWithPermit` Functionality

Extends `relayCall` with Permit2 support, enabling token payments through a permit signature, eliminating the need for on-chain approvals. The payment is processed before relaying the call via the trusted forwarder.

`relayCallWithTwoPermits` Functionality

Handles complex payments requiring two permits: one for the token and another for Permit2. Both permits are verified, payment is received, and the transaction is relayed through the trusted forwarder.

General Functionalities

The constructor sets up essential components, including the trusted forwarder, treasury, and token limits. Role-based access allows operators to relay transactions, while token limit managers can set token payment limits. ERC2771Context integration enables gasless transactions on behalf of users.

Authorities Functionalities

Inherits from `Authorizable`, managing operator roles for executing relays and token limit manager roles for setting token limits, ensuring secure and controlled access to the contract's functions.

TeaFiTrustedForwarder Contract

The `TeaFiTrustedForwarder` contract serves as a secure meta-transaction forwarder with role-based access control, allowing only authorized relayers to execute transactions. By whitelisting specific relayer contracts, it adds a layer of security to the forwarding process, ensuring that only trusted entities can interact with the contract.

`execute` Functionality

Allows authorized relayers with `PROXY_ROLE` to execute a single meta-transaction on behalf of a user. The function securely forwards the transaction request, maintaining strict access control.

`executeBatch` Functionality

Enables relayers to process multiple meta-transactions in a batch, reducing gas costs and improving transaction efficiency. Only whitelisted relayers can call this function, further securing the batch execution.

General Functionalities

The constructor assigns initial admin rights, and the `setupRoles` function is used to designate multisig wallets as admins and assign the `PROXY_ROLE` to trusted relayer addresses. Once initialized, the contract prevents further role changes, ensuring security and stability.

Authorities Functionalities

Implements `AccessControl` to manage roles like `DEFAULT_ADMIN_ROLE` and `PROXY_ROLE`. These roles are set during deployment and are locked post-initialization, allowing only trusted relayers to interact with the forwarder and ensuring robust security.

SynthToken Contract

The `SynthToken` contract is an ERC20-compatible token that represents synthetic assets tied to an underlying asset. It supports meta-transactions through a trusted forwarder and uses Permit2 for off-chain token approvals. The contract allows for wrapping and unwrapping of assets, minting synthetic tokens upon wrapping and burning them during unwrapping, with pausing capabilities for added control.

`wrap` Functionality

The `wrap` function enables users to convert underlying assets into synthetic tokens. It transfers the underlying asset to the treasury and mints an equivalent amount of synthetic tokens, supporting both standard ERC20 approvals and Permit2 for token transfers.

`unwrap` Functionality

This function allows users to convert synthetic tokens back into the underlying asset by burning the synthetic tokens and transferring the corresponding amount from the treasury to the user.

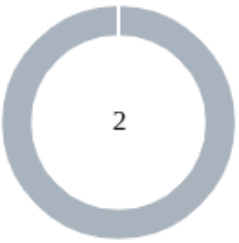
General Functionalities

The constructor sets up the synthetic token's name, symbol, underlying asset, treasury, and trusted forwarder, assigning the factory as the contract's owner. It integrates `ERC2771Context` for meta-transaction support and includes `pause` and `unpause` methods to restrict all token activities when needed.

Authorities Functionalities

The factory address manages pausing, with role-based restrictions ensuring that only the factory can invoke `pause` and `unpause`. This provides controlled access over token transfers, minting, and burning, enhancing the contract's security and flexibility.

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	2

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	0	2	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	CCR	Contract Centralization Risk	Acknowledged
●	PTAI	Potential Transfer Amount Inconsistency	Acknowledged

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	SynthTokenFactory.sol#L115,152,163,174 TeaFiRelayer.sol#L57,75,86 TeaFiTrustedForwarder.sol#L35,57,66
Status	Unresolved

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

Specifically, the following roles have significant authority over key contract functions:

- **OPERATOR_ROLE**: Has the authority to create new synthetic tokens with the provided settings and parameters, and execute relay calls.
- **DEFAULT_ADMIN_ROLE**: Has the authority to set the global treasury and the trusted forwarder address, and set up roles.
- **TOKEN_MANAGER**: Can pause and unpause transactions.
- **TOKEN_LIMIT_MANAGER_ROLE**: Can change token limits.
- **PROXY_ROLE**: Can execute single and multiple transactions via the relayer.

This concentration of power could lead to potential abuse or mismanagement if these roles are not properly decentralized or adequately secured.

```
  function createSynthTokens(
        TokenSettings[] memory args
    ) external onlyRole(OPERATOR_ROLE) returns (address[]
memory) {
        address[] memory tokens = new address[](args.length);
        ...
        return tokens;
    }

    function setGlobalTreasury(address newTreasury) external
virtual onlyRole(DEFAULT_ADMIN_ROLE) {
        ...
        factorySettings.globalTreasury = newTreasury;

        emit GlobalTreasurySet(newTreasury);
    }

    function setTrustedForwarder(address newForwarder) external
virtual onlyRole(DEFAULT_ADMIN_ROLE) {
        ...
        factorySettings.trustedForwarder = newForwarder;

        emit TrustedForwarderSet(newForwarder);
    }


    function pauseTokens(address[] calldata tokens) external
onlyRole(TOKEN_MANAGER) {
        ...
            SynthToken(tokens[i]).pause();
        }
    }

    function unpauseTokens(address[] calldata tokens) external
onlyRole(TOKEN_MANAGER) {
       ...
            SynthToken(tokens[i]).unpause();
        }
    }
```

```
    function changeTokenLimit(
        address[] calldata tokens,
        uint256[] calldata limits
    ) external override onlyRole(TOKEN_LIMIT_MANAGER_ROLE) {
        ...
            paymentTokenLimit[token] = limit;

            emit TokenLimitChanged(token, limit);
        }
    }

    function relayCall(
        ERC2771Forwarder.ForwardRequestData calldata request,
        PermitPayable.PaymentData calldata paymentData
    ) external override onlyRole(OPERATOR_ROLE)
checkSupplierAndSigner(request.from, paymentData.payer) {
        // receive payment
        _receivePayment(paymentData);
        // execute the call
        trustedForwarder.execute(request);
    }

    function relayCallWithPermit(
        ERC2771Forwarder.ForwardRequestData calldata request,
        PermitPayable.PaymentData calldata paymentData,
        IAllowanceTransfer.PermitSingle calldata
permitSignatureDetails,
        bytes calldata permitSingleSignature
    ) external override onlyRole(OPERATOR_ROLE)
checkSupplierAndSigner(request.from, paymentData.payer) {
        ...
        trustedForwarder.execute(request);
    }
```

```
function setupRoles(address multisigWallet, address[] memory
relayers) external onlyRole(DEFAULT_ADMIN_ROLE) {
        ...
        initialized = true;
    }

    function execute(ForwardRequestData calldata request)
public payable override onlyRole(PROXY_ROLE) {
        super.execute(request);
    }

    function executeBatch(
        ForwardRequestData[] calldata requests,
        address payable refundReceiver
    ) public payable override onlyRole(PROXY_ROLE) {
        super.executeBatch(requests, refundReceiver);
    }
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

Team Update

The team has acknowledged that this is not a security issue and states:

The DEFAULT_ADMIN_ROLE is a multisig contract where the administrators are 3 people and to implement a transaction, at least 2 signatures are needed, which minimizes centralization and all other operators/managers will be reliably protected. If we suddenly lose access to one of OPERATOR_ROLE or TOKEN_MANAGER or TOKEN_LIMIT_MANAGER_ROLE, the DEFAULT_ADMIN_ROLE can revoke the role for them. For PROXY_ROLE it is impossible to do something critical because you need a user signature.

PTAI - Potential Transfer Amount Inconsistency

Criticality	Minor / Informative
Location	SynthToken.sol#L95
Status	Unresolved

Description

The `safeTransferFrom` function IS used to transfer a specified amount of tokens to an address. The fee or tax is an amount that is charged to the sender of an ERC20 token when tokens are transferred to another address. According to the specification, the transferred amount could potentially be less than the expected amount. This may produce inconsistency between the expected and the actual behavior.

The following example depicts the diversion between the expected and actual amount.

Tax	Amount	Expected	Actual
No Tax	100	100	100
10% Tax	100	100	90

```
SafeERC20.safeTransferFrom(IERC20(underlyingAsset),
_msgSender(), treasury, amount);
```

Recommendation

The team is advised to take into consideration the actual amount that has been transferred instead of the expected.

It is important to note that an ERC20 transfer tax is not a standard feature of the ERC20 specification, and it is not universally implemented by all ERC20 contracts. Therefore, the contract could produce the actual amount by calculating the difference between the transfer call.

`Actual Transferred Amount = Balance After Transfer - Balance Before Transfer`

Team Update

The team has acknowledged that this is not a security issue and states:

SynthToken contracts can only be created by the Tea-Fi team, which designed the underlying assets to be free of transfer fees. The addition of some additional checks complicates the execution of the contract and increases the consumption of gas.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
TeaFiTrustedForwarder	Implementation	ERC2771Forwarder, AccessControl, ZeroAddressError		
		Public	✓	ERC2771Forwarder
	setupRoles	External	✓	onlyRole
	execute	Public	Payable	onlyRole
	executeBatch	Public	Payable	onlyRole
	hashTypedDataV4	External		-
TeaFiRelayer	Implementation	PermitPayable, Authorizable, ITeaFiRelayer		
		Public	✓	PermitPayable Authorizable
	changeTokenLimit	External	✓	onlyRole
	relayCall	External	✓	onlyRole checkSupplierAndSigner
	relayCallWithPermit	External	✓	onlyRole checkSupplierAndSigner
	relayCallWithTwoPermits	External	✓	onlyRole checkSupplierAndSigner

SynthTokenFactory	Implementation	AccessControl		
		Public	✓	-
	createSynthTokens	External	✓	onlyRole
	setGlobalTreasury	External	✓	onlyRole
	setTrustedForwarder	External	✓	onlyRole
	pauseTokens	External	✓	onlyRole
	unpauseTokens	External	✓	onlyRole
SynthToken	Implementation	ERC20, ISynthToken, ERC2771Context, ERC20Permit, Permitable, ERC20Pausable		
		Public	✓	ERC20 ERC2771Context Permitable ERC20Permit
	wrap	External	✓	-
	wrap	Public	✓	-
	wrap	Public	✓	checkZeroAmount
	unwrap	External	✓	checkZeroAmount
	pause	External	✓	onlyFactory
	unpause	External	✓	onlyFactory
	_update	Internal	✓	
	_msgSender	Internal		
	_msgData	Internal		

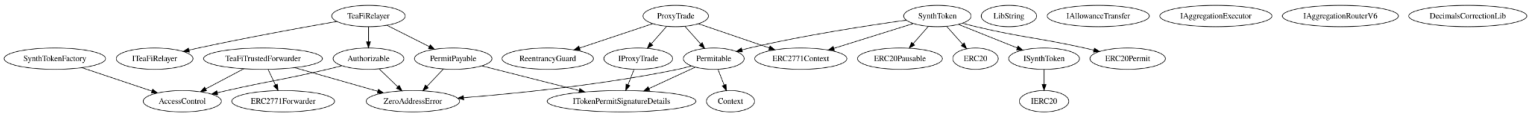
	_contextSuffixLength	Internal		
	hashTypedDataV4	External		-
ProxyTrade	Implementation	ERC2771Context, Permitable, IProxyTrade, ReentrancyGuard		
		Public	✓	ERC2771Context Permitable
	makePublicSwapWithTwoPermits	External	✓	-
	makePublicSwapWithPermit	Public	✓	-
	makePublicSwap	Public	✓	-
	relayToDopWithApprovalAndTwoPermits	External	✓	-
	relayToDopWithApprovalAndPermit	Public	✓	-
	relayToDopWithApproval	Public	✓	-
	relayToDop	Public	✓	nonReentrant
	_afterOneInchSwap	Internal	✓	
	_checkAllowance	Internal	✓	
	_msgSender	Internal		
	_msgData	Internal		
	_contextSuffixLength	Internal		
LibString	Library			
	toString	Internal		
	toString	Internal		

	toHexString	Internal		
	toHexStringNoPrefix	Internal		
	toHexString	Internal		
	toMinimalHexString	Internal		
	toMinimalHexStringNoPrefix	Internal		
	toHexStringNoPrefix	Internal		
	toHexStringChecksummed	Internal		
	toHexString	Internal		
	toHexStringNoPrefix	Internal		
	toHexString	Internal		
	toHexStringNoPrefix	Internal		
	runeCount	Internal		
	is7BitASCII	Internal		
	replace	Internal		
	indexOf	Internal		
	indexOf	Internal		
	lastIndexOf	Internal		
	lastIndexOf	Internal		
	contains	Internal		
	startsWith	Internal		
	endsWith	Internal		
	repeat	Internal		
	slice	Internal		

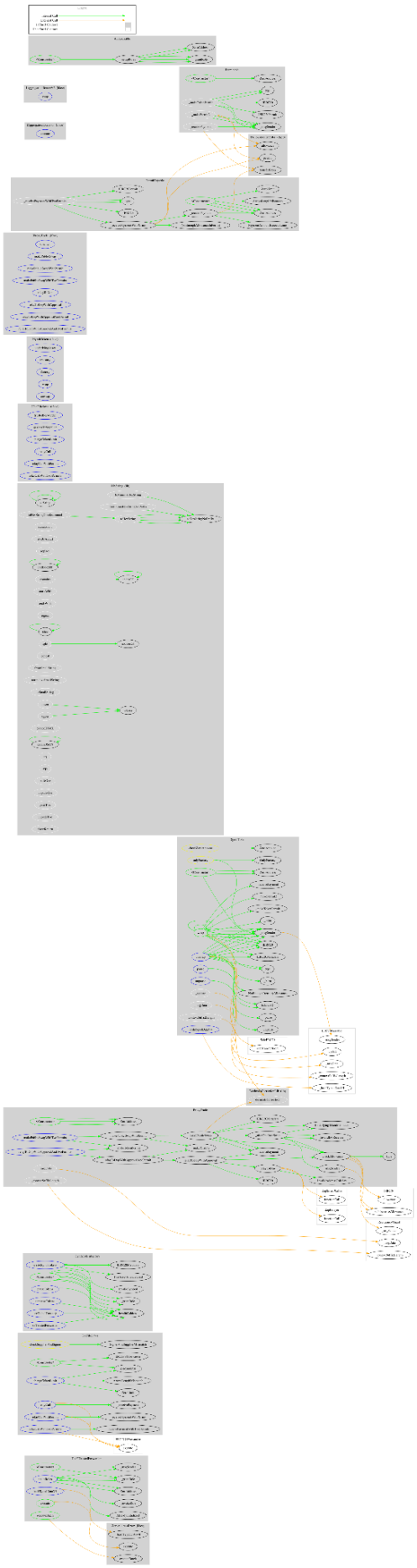
	slice	Internal		
	indicesOf	Internal		
	split	Internal		
	concat	Internal		
	toCase	Internal		
	fromSmallString	Internal		
	normalizeSmallString	Internal		
	toSmallString	Internal		
	lower	Internal		
	upper	Internal		
	escapeHTML	Internal		
	escapeJSON	Internal		
	escapeJSON	Internal		
	eq	Internal		
	eqs	Internal		
	packOne	Internal		
	unpackOne	Internal		
	packTwo	Internal		
	unpackTwo	Internal		
	directReturn	Internal		
ZeroAddressError	Interface			

Permitable	Implementation	ZeroAddressError, ITokenPermitSignatureDetails, Context		
		Public	✓	-
	_makeTokenPermit	Internal	✓	
	_makePermit2	Internal	✓	
	_receivePayment	Internal	✓	
PermitPayable	Implementation	ZeroAddressError, ITokenPermitSignatureDetails		
		Public	✓	-
	_receivePaymentWithTwoPermits	Internal	✓	
	_receivePaymentWithPermit	Internal	✓	
	_receivePayment	Internal	✓	
DecimalsCorrectionLib	Library			
	decimalsCorrection	Internal		
Authorizable	Implementation	AccessControl, ZeroAddressError		
		Public	✓	-
	_setupRoles	Internal	✓	

Inheritance Graph



Flow Graph



Summary

The Tea-Fi suite of contracts implements a comprehensive system for facilitating token swaps, synthetic asset creation, meta-transactions, and role-based access control. This audit investigates security vulnerabilities, business logic concerns, and potential improvements in the use of trusted forwarders, the Permit2 system, and role-based governance across the contracts. The team has acknowledged the findings.

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

cyberscope.io