



Cyberscope

## Audit Report

# Crypto Bitaxy

January 2023

Type      BEP20

Network      BSC

Address      0x848B2113F00577C96Cb15eaD79dd0fe87dE6379F

Audited by      © cyberscope

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Review</b>	<b>3</b>
<b>Audit Updates</b>	<b>3</b>
<b>Source Files</b>	<b>3</b>
<b>Analysis</b>	<b>4</b>
<b>ELFM - Exceeds Fees Limit</b>	<b>5</b>
<b>Description</b>	<b>5</b>
<b>Recommendation</b>	<b>6</b>
<b>Diagnostics</b>	<b>7</b>
<b>CO - Code Optimization</b>	<b>8</b>
<b>Description</b>	<b>8</b>
<b>Recommendation</b>	<b>8</b>
<b>L02 - State Variables could be Declared Constant</b>	<b>9</b>
<b>Description</b>	<b>9</b>
<b>Recommendation</b>	<b>9</b>
<b>L04 - Conformance to Solidity Naming Conventions</b>	<b>10</b>
<b>Description</b>	<b>10</b>
<b>Recommendation</b>	<b>11</b>
<b>L07 - Missing Events Arithmetic</b>	<b>12</b>
<b>Description</b>	<b>12</b>
<b>Recommendation</b>	<b>12</b>
<b>L09 - Dead Code Elimination</b>	<b>13</b>
<b>Description</b>	<b>13</b>
<b>Recommendation</b>	<b>14</b>
<b>L16 - Validate Variable Setters</b>	<b>15</b>
<b>Description</b>	<b>15</b>
<b>Recommendation</b>	<b>15</b>
<b>L17 - Usage of Solidity Assembly</b>	<b>16</b>
<b>Description</b>	<b>16</b>
<b>Recommendation</b>	<b>16</b>
<b>Contract Functions</b>	<b>17</b>

<b>Inheritance Graph</b>	<b>23</b>
<b>Flow Graph</b>	<b>24</b>
<b>Summary</b>	<b>25</b>
<b>Disclaimer</b>	<b>26</b>
<b>About Cyberscope</b>	<b>27</b>

## Review

<b>Contract Name</b>	CryptoBitaxy
<b>Compiler Version</b>	v0.6.12+commit.27d51765
<b>Optimization</b>	200 runs
<b>Explorer</b>	<a href="https://bscscan.com/address/0x848b2113f00577c96cb15ead79dd0fe87de6379f">https://bscscan.com/address/0x848b2113f00577c96cb15ead79dd0fe87de6379f</a>
<b>Address</b>	0x848b2113f00577c96cb15ead79dd0fe87de6379f
<b>Network</b>	BSC
<b>Symbol</b>	C-BITAXY
<b>Decimals</b>	9
<b>Total Supply</b>	250,000,000

## Audit Updates

<b>Initial Audit</b>	09 Jan 2023
----------------------	-------------

## Source Files

<b>Filename</b>	SHA256
<b>CryptoBitaxy.sol</b>	df5bd33d9d74010ef44ff5f36adec6f86df475475f1bd6b6e2257228eed9619e

# 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	Unresolved
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

## ELFM - Exceeds Fees Limit

<b>Criticality</b>	Critical
<b>Location</b>	CryptoBitaxy.sol#L901,905,909,913
<b>Status</b>	Unresolved

### Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the following functions with a high percentage value:

- `setTaxFeePercent`
- `setLiquidityFeePercent`
- `setCharityFeePercent`
- `setBurnFeePercent`

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}
...
function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
    _liquidityFee = liquidityFee;
}
...
function setCharityFeePercent(uint256 charityFee) external onlyOwner() {
    _charityFee = charityFee;
}
...
function setBurnFeePercent(uint256 burnFee) external onlyOwner() {
    _burnFee = burnFee;
}
```

## Recommendation

The contract could embody a check for the maximum acceptable value. 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
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved



## CO - Code Optimization

<b>Criticality</b>	Minor / Informative
<b>Location</b>	CryptoBitaxy.sol#L832,834,836,838,840
<b>Status</b>	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 contract is calculating the same value multiple times. More specifically, the amount remaining after subtracting the fees. As a result, the operation is performed multiple times instead of one.

```
if (!_isExcluded[sender] && !_isExcluded[recipient]) {
    _transferFromExcluded(sender, recipient, (amount.sub(burnAmt).sub(charityAmt)));
} else if (!_isExcluded[sender] && _isExcluded[recipient]) {
    _transferToExcluded(sender, recipient, (amount.sub(burnAmt).sub(charityAmt)));
} else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
    _transferStandard(sender, recipient, (amount.sub(burnAmt).sub(charityAmt)));
} else if (_isExcluded[sender] && _isExcluded[recipient]) {
    _transferBothExcluded(sender, recipient, (amount.sub(burnAmt).sub(charityAmt)));
} else {
    _transferStandard(sender, recipient, (amount.sub(burnAmt).sub(charityAmt)));
}
```

### 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. A recommendation would be to extract the `amount.sub(burnAmt).sub(charityAmt)` into a new variable or overwrite the existing one, and reuse the variable instead.

## L02 - State Variables could be Declared Constant

<b>Criticality</b>	Minor / Informative
<b>Location</b>	CryptoBitaxy.sol#L457,461,462,463,488
<b>Status</b>	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.

```
6 private _tTotal = 250 * 10**6 * 10**9;  
  
private _name = "Crypto Bitaxy";  
  
private _symbol = "C-BITAXY";  
  
private _decimals = 9;  
  
6 private numTokensSellToAddToLiquidity = 50 * 10**18;
```

### 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	CryptoBitaxy.sol#L270,271,287,308,466,470,474,478,691,697,923
<b>Status</b>	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.

```
on DOMAIN_SEPARATOR() external view returns (bytes32);

on PERMIT_TYPEHASH() external pure returns (bytes32);

...

6 public _taxFee = 1;

6 public _liquidityFee = 1;

...
```

## 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	CryptoBitaxy.sol#L902,906,910,914
<b>Status</b>	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.

```
e = taxFee;  
  
dityFee = liquidityFee;  
  
tyFee = charityFee;  
  
ee = burnFee;
```

### 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	CryptoBitaxy.sol#L107,118,126,130,134,138,143
<b>Status</b>	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.

```
on isContract(address account) internal view returns (bool) {
    // According to EIP-1052, 0x0 is the value returned for not-yet created
    accounts
    // and 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470 is
    returned
    // for accounts without code, i.e. `keccak256('')`
    bytes32 codehash;
    bytes32 accountHash =
0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
    // solhint-disable-next-line no-inline-assembly
    assembly { codehash := extcodehash(account) }
    return (codehash != accountHash && codehash != 0x0);
}

...
```

## 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.

## L16 - Validate Variable Setters

<b>Criticality</b>	Minor / Informative
<b>Location</b>	CryptoBitaxy.sol#L898
<b>Status</b>	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.

```
yWallet = newWallet;
```

### 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.



## L17 - Usage of Solidity Assembly

<b>Criticality</b>	Minor / Informative
<b>Location</b>	CryptoBitaxy.sol#L114,156
<b>Status</b>	Unresolved

### Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
ly { codehash := extcodehash(account) }  
  
ly {  
    let returndata_size := mload(returndata)  
    revert(add(32, returndata), returndata_size)  
}
```

### Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

# Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>IERC20</b>	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
<b>SafeMath</b>	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
<b>Context</b>	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
<b>Address</b>	Library			
	isContract	Internal		

	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	_functionCallWithValue	Private	✓	
<b>Ownable</b>	Implementation	Context		
		Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	geUnlockTime	Public		-
	lock	Public	✓	onlyOwner
	unlock	Public	✓	-
<b>IUniswapV2Factory</b>	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
<b>IUniswapV2Pair</b>	Interface			
	name	External		-
	symbol	External		-

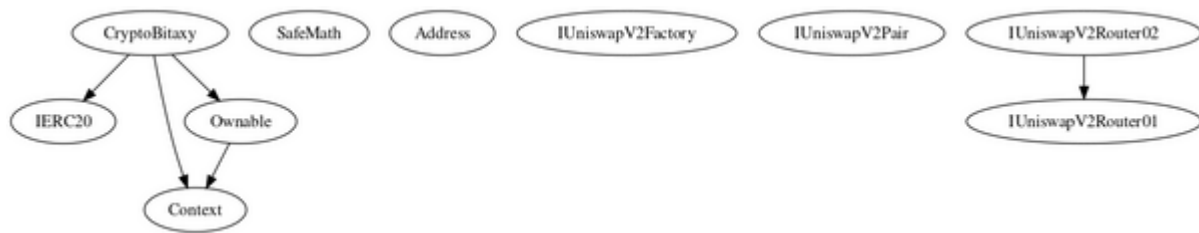
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
<b>IUniswapV2Router01</b>	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		-
<b>IUniswapV2Router02</b>	Interface	IUniswapV2 Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
<b>CryptoBitaxy</b>	Implementation	Context, IERC20, Ownable		
		Public	✓	-

	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	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
	_transferBothExcluded	Private	✓	
		External	Payable	-
	_reflectFee	Private	✓	
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidity	Private	✓	
	calculateTaxFee	Private		
	calculateLiquidityFee	Private		

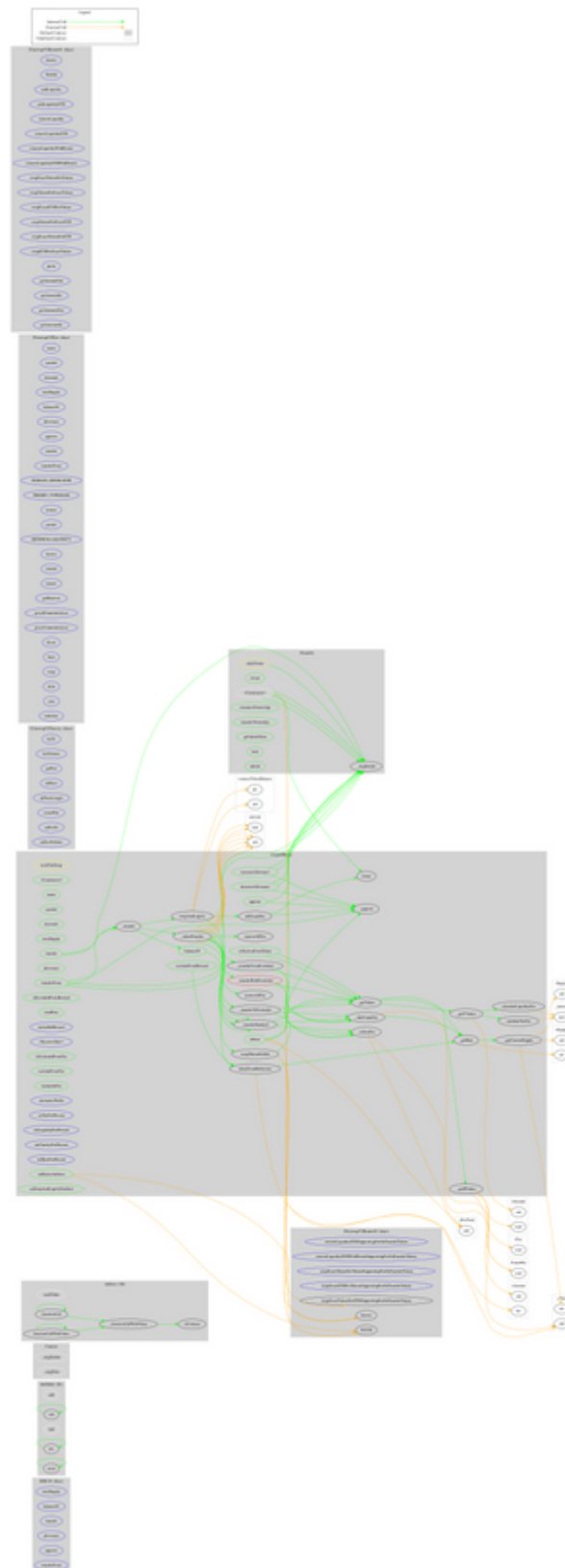
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	
	isExcludedFromFee	Public		-
	_approve	Private	✓	
	_transfer	Private	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	
	excludeFromFee	Public	✓	onlyOwner
	includeInFee	Public	✓	onlyOwner
	setcharityWallet	External	✓	onlyOwner
	setTaxFeePercent	External	✓	onlyOwner
	setLiquidityFeePercent	External	✓	onlyOwner
	setCharityFeePercent	External	✓	onlyOwner
	setBurnFeePercent	External	✓	onlyOwner
	setRouterAddress	Public	✓	onlyOwner
	setSwapAndLiquifyEnabled	Public	✓	onlyOwner

# Inheritance Graph





# Flow Graph



## Summary

There are some functions that can be abused by the owner like manipulate the fees. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

The contract contains a tax inconsistency. The Burn/Charity fees are the primary tax factors and the Tax/Liquidity fees are the secondary factors. For instance, we assume the following example:

- The Burn fee is 10%.
- The Charity fee is 10%.
- The Tax fee is 10%.
- The Liquidity fee is 10%.
- The transferred amount is 100.

The burn and charity amount will be 10% of the 100 tokens, so 10 tokens will be taxed. The tax and liquidity fee will be 10% of the remaining 80 tokens, so 8 tokens will be taxed.

## 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>