Cyberscope

Audit Report

zoozToken

March 2023

Type	BEP20
Network	BSC
Address	0x132306a39d6fC1E49C3Cb6D8FE8d07d4D44B462a
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Review 2
Audit Updates 2
Source Files 2
Roles 3
Analysis 4
BC - Blacklists Addresses 5
Description 5
Recommendation 5
Diagnostics 6
MVN - Misleading Variables Naming 7
Description 7
Recommendation 7
L04 - Conformance to Solidity Naming Conventions 8
Description 8
Recommendation 8
Functions Analysis 9
Inheritance Graph 11
Flow Graph 12
Summary 13
Disclaimer 14
About Cyberscope 15

Review

Contract Name	ZOOZToken
Compiler Version	v0.8.7+commit.e28d00a7
Optimization	200 runs
Explorer	https://bscscan.com/address/0x132306a39d6fc1e49c3cb6d8fe8d07d4d44b462a
Address	0x132306a39d6fc1e49c3cb6d8fe8d07d4d44b462a
Network	BSC
Symbol	ZOOZ
Decimals	9
Total Supply	770,000,000

Audit Updates

Initial Audit	16 Feb 2023 https://github.com/cyberscope-io/audits/blob/main/zooz/v1/audit.pdf
Corrected Phase 2	01 Mar 2023 https://github.com/cyberscope-io/audits/blob/main/zooz/v2/audit.pdf
Corrected Phase 3	13 Mar 2023

Source Files

Filename	SHA256
ZOOZToken.sol	ecdbda204d2c247ad0ef19a55e984d73b8dbd05658a353049a35ad946bd1d1b7

Roles

The contract consist of three roles. The `owner`, `manager,` and `governor` roles.

The `Owner` has the authority to

- Grant or revokes the `governor` and the `manager` role.
- Renounce or Transfer ownership.
- Set blacklisted addresses.
- Set excluded from fees addresses.

The `Manager` has the authority to

- Transfer the `manager` role.
- Configure reward address.

The `Governor` has the authority to

- Set blacklisted addresses.
- Set excluded from fees addresses.

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	Unresolved

BC - Blacklists Addresses

Criticality	Medium
Location	contracts/ZOOZToken.sol#L467
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `setBlockedAddress` function.

```
function setBlockedAddress(address holderAddress, bool blocked) public
onlyGovernance()  {
        require(holderAddress != address(0), "HolderAddress can't be the zero
address");

    blockedAddresses[holderAddress][_msgSender()] = blocked;

        if(blocked) {
            emit AddressBlocked(holderAddress);
            return;
        }

        emit AddressUnblocked(holderAddress);
}
```

Recommendation

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
●	MVN	Misleading Variables Naming	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved

MVN - Misleading Variables Naming

Criticality	Minor / Informative
Location	contracts/ZOOZToken.sol#L210,384
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.

The contract utilizes the variable `botAddresses` and the function `_isItBotAddress`. These variables and the function implement an exclude from the fees mechanism.

```
mapping (address => mapping (address => bool)) internal botAddresses;

function _isItBotAddress(address addr) internal view returns(bool) {
    return botAddresses[addr][governance1Address]
        && botAddresses[addr][governance2Address]
        && botAddresses[addr][governance3Address];
}
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	ZOOZToken.sol#L216
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.

```
uint256 public constant totalsupply
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

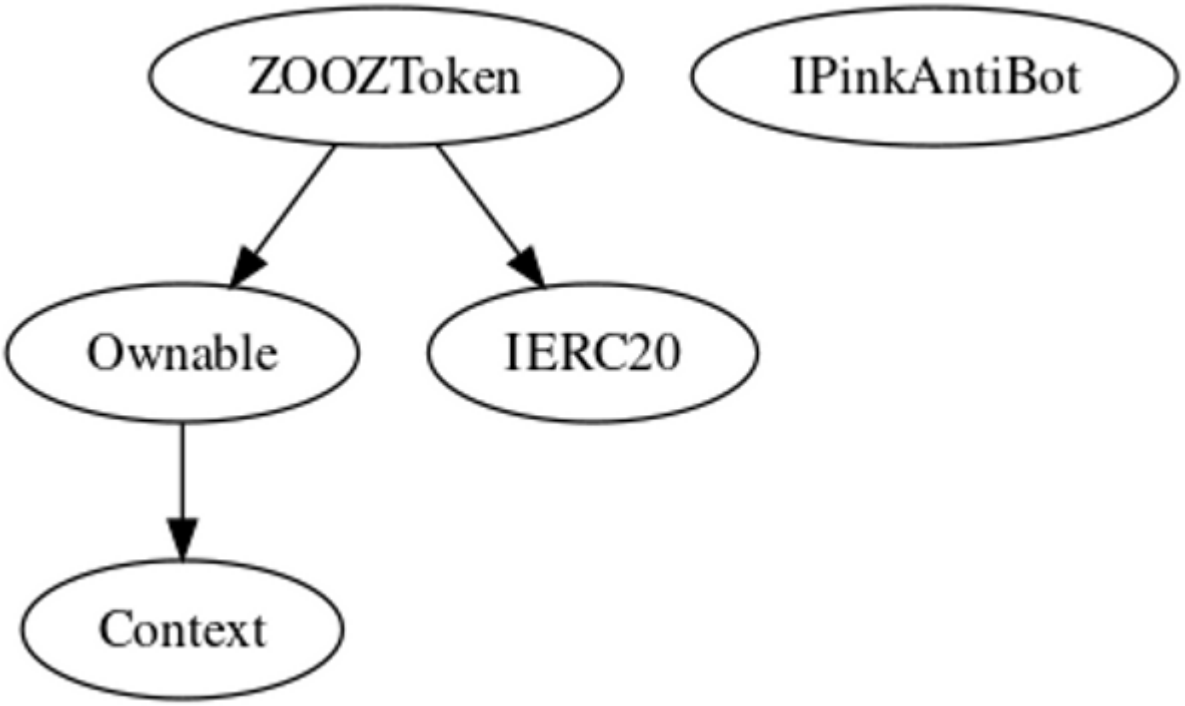
Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IPinkAntiBot	Interface			
	setTokenOwner	External	✓	-
	onPreTransferCheck	External	✓	-

ZOOZToken	Implementation	Ownable, IERC20		
		Public	✓	-
	totalSupply	Public		-
	balanceOf	Public		-
	timestampOf	Public		-
	balancesOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_approve	Private	✓	
	_transfer	Private	✓	
	_isBlockedAddress	Internal		
	_isItBotAddress	Internal		
	_getFees	Internal		
	_holdDateHook	Internal	✓	
	_stdTransfer	Private	✓	
	setManagerAddress	Public	✓	onlyManager
	setRewardsTeamAddress	Public	✓	onlyManager
	setBlockedAddress	Public	✓	onlyGovernance
	setBotAddress	Public	✓	onlyGovernance
	setPair	Public	✓	onlyManager
	setEnableAntiBot	External	✓	onlyManager
	setGovernance	Public	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
renounceOwnership
transferOwnership
<Constructor>
onlyOwner
_transferOwnership
_msgSender
owner
IPinkAntiBot (iface)
setTokenOwner
onPreTransferCheck
ZOOZToken
setBlockedAddress
setBotAddress
<Constructor>
totalSupply
balanceOf
timestampOf
balancesOf
onlyGovernance
allowance
onlyManager
transfer
setManagerAddress
setRewardsTeamAddress
transferFrom
approve
setPair
setEnableAntiBot
setGovernance
_transfer
_approve
_holdDateHook
_getFees
_isBotAddress
_isBlockedAddress
_stdTransfer
Context
_msgSender
_msgData

Summary

ZOOZ Token contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like blacklist addresses. 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 fee percentage decreases over time, as the time elapsed since the last transaction of the holder. If the time elapsed is less than or equal to 1 week, the fee percentage is 14%. After 1 week, the fee percentage decreases to 10% for the next 3 weeks (1 month total). After 3 months, the fee percentage decreases again to 5%, and after 6 months, the fee percentage decreases to 2%. Eventually, after more than 6 months the fee percentage reaches 0.

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

Cyberscope

Audit Report

zoozToken

February 2023

SHA256 e70a4a1de2336825b93622c836a7317ae371ea7f70fc86d071e761c7c6114c3d

Audited by © cyberscope

Table of Contents

Table of Contents 1
Review 3
Audit Updates 3
Source Files 3
Roles 4
Analysis 5
BC - Blacklists Addresses 6
Description 6
Recommendation 6
Diagnostics 7
RSML - Redundant SafeMath Library 8
Description 8
Recommendation 8
US - Untrusted Source 9
Description 9
Recommendation 9
TUU - Time Units Usage 10
Description 10
Recommendation 11
MVN - Misleading Variables Naming 12
Description 12
Recommendation 12
L02 - State Variables could be Declared Constant 13
Description 13
Recommendation 13
L04 - Conformance to Solidity Naming Conventions 14
Description 14
Recommendation 14
L09 - Dead Code Elimination 16
Description 16
Recommendation 16
L16 - Validate Variable Setters 18

Description 18
Recommendation 18
L17 - Usage of Solidity Assembly 19
Description 19
Recommendation 19
L19 - Stable Compiler Version 20
Description 20
Recommendation 20
Functions Analysis 21
Inheritance Graph 24
Flow Graph 25
Summary 26
Disclaimer 27
About Cyberscope 28

Review

Contract Name	ZOOZToken
Repository	https://github.com/coalichain/ZOOZToken/blob/main/contracts
Commit	26a3c8eb77920bcb95e8f27fce25e83bf14b7abc
Testing Deploy	https://testnet.bscscan.com/address/0xd75ae0d91a2db2c8f5363c87999e75e7dad7cf28
Symbol	ZOOZ
Decimals	9
Total Supply	770.000.000

Audit Updates

Initial Audit	16 Feb 2023

Source Files

Filename	SHA256
contracts/ZOOZToken.sol	e70a4a1de2336825b93622c836a7317ae371ea7f70fc86d071e761c7c6114c3d

Roles

The contract consist of three roles. The `owner`, `manager,` and `governor` roles.

The `Owner` has the authority to

- Grant or revokes the `governor` and the `manager` role.
- Renounce or Transfer ownership.
- Set blacklisted addresses.
- Set excluded from fees addresses.

The `Manager` has the authority to

- Transfer the `manager` role.
- Configure reward address.

The `Governor` has the authority to

- Set blacklisted addresses.
- Set excluded from fees addresses.

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	Unresolved

BC - Blacklists Addresses

Criticality	Medium
Location	contracts/ZOOZToken.sol#L886
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `setBlocked` function.

```
function setBlocked(address holderAddress, bool blocked)
    public
  onlyGovernance
{
  require(
    holderAddress != address(0),
    "HolderAddress can't be the zero address"
  );

  _blocked[holderAddress][_msgSender()] = blocked;

  if (blocked) {
    emit AddressBlocked(holderAddress);
    return;
  }

  emit AddressUnblocked(holderAddress);
}
```

Recommendation

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
●	RSML	Redundant SafeMath Library	Unresolved
●	US	Untrusted Source	Unresolved
●	TUU	Time Units Usage	Unresolved
●	MVN	Misleading Variables Naming	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved
●	L19	Stable Compiler Version	Unresolved

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	contracts/ZOOZToken.sol#L321
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 on 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 unnecessarily the gas consumption.

```
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.

US - Untrusted Source

Criticality	Critical
Location	contracts/ZOOZToken.sol#L941
Status	Unresolved

Description

The contract uses an external contract in order to determine the transaction's flow. The external contract is untrusted. As a result, it may produce security issues and harm the transactions.

```
function setAntiBotAddr(address pinkbotAddr) external onlyManager() {
    pinkAntiBot = IPinkAntiBot(pinkbotAddr);
    pinkAntiBot.setTokenOwner(msg.sender);
}
```

Recommendation

The contract should use a trusted external source. A trusted source could be either a commonly recognized or an audited contract. The pointing addresses should not be able to change after the initialization.

TUU - Time Units Usage

Criticality	Minor / Informative
Location	contracts/ZOOZToken.sol#L816
Status	Unresolved

Description

The contract is using arbitrary numbers to form time-related values. As a result, it decreases the readability of the codebase and prevents the compiler to optimize the source code.

```
function _getFees(uint timestamp) internal view returns(uint256) {
    if(timestamp == 0 || timestamp >= block.timestamp)
        return 14;

    uint diff = block.timestamp - timestamp;

    // 1 Week: 3600 * 24 * 7
    if(diff <= 604800)
        return 14;

    // 1 Month: 3600 * 24 * 30
    if(diff <= 2592000)
        return 10;

    // 3 Months: 3600 * 24 * 30 * 3
    if(diff <= 7776000)
        return 5;

    // 6 Months: 3600 * 24 * 30 * 6
    if(diff <= 15552000)
        return 2;

    // > 6 Months
    return 0;
}
```

Recommendation

It is a good practice to use the time units reserved keywords like `seconds`, `minutes`, `hours`, `days`, `weeks` and `years` to process time-related calculations.

It's important to note that these time units are simply a shorthand notation for representing time in seconds, and do not have any effect on the actual passage of time or the execution of the contract. The time units are simply a convenience for expressing time in a more human-readable form.

MVN - Misleading Variables Naming

Criticality	Minor / Informative
Location	contracts/ZOOZToken.sol#L639,807
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.

The contract utilizes the variable `bots` and the function `_isBot`. This variable and function implement a mechanism to exclude address from fees.

```
mapping (address => mapping (address => bool)) internal _bots;

function _isBot(address addr) internal view returns(bool) {
    return _bots[addr][governance1Address]
            && _bots[addr][governance2Address]
            && _bots[addr][governance3Address];
}
```

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.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	ZOOZToken.sol#L664,665,666,667
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.

```
string public name = "ZOOZ Token"
string public symbol = "ZOOZ"
uint8 public decimals = 9
uint256 public _totalSupply = 770 * 10**6 * 10**9
```

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	ZOOZToken.sol#L658,660,661,662,667
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.

```
mapping(address => Holder) internal _balances
mapping(address => bool) internal _pairs
mapping(address => mapping(address => bool)) internal _bots
mapping(address => mapping(address => bool)) internal _blocked
uint256 public _totalSupply = 770 * 10**6 * 10**9
```

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	ZOOZToken.sol#L120,148,176,189,208,228,250,265,282,300,311
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 isContract(address account) internal view returns (bool) {
    // This method relies on extcodesize, which returns 0 for contracts
in
    // construction, since the code is only stored at the end of the
    // constructor execution.

    uint256 size;
    assembly {
      size := extcodesize(account)
    }
    return size > 0;
  }

...
```

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

Criticality	Minor / Informative
Location	ZOOZToken.sol#L920,929
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.

```
managerAddress = newAddress
rewardsAddress = newAddress
```

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

Criticality	Minor / Informative
Location	ZOOZToken.sol#L126,323
Status	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.

```
assembly {
      size := extcodesize(account)
    }

assembly {
          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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	ZOOZToken.sol#L3
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.

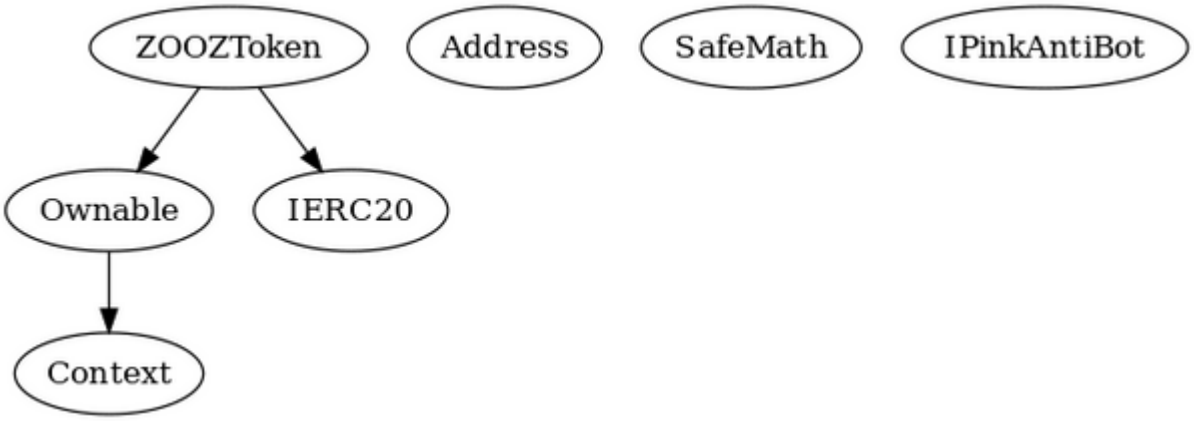
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	_verifyCallResult	Private		

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		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IPinkAntiBot	Interface			
	setTokenOwner	External	✓	-
	onPreTransferCheck	External	✓	-
ZOOZToken	Implementation	Ownable, IERC20		
		Public	✓	-

	totalSupply	Public		-
	balanceOf	Public		-
	timestampOf	Public		-
	balancesOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_approve	Private	✓	
	_transfer	Private	✓	
	_isBlocked	Internal		
	_isBot	Internal		
	_getFees	Internal		
	_holdDateHook	Internal	✓	
	_stdTransfer	Private	✓	
	setManagerAddress	Public	✓	onlyManager
	setRewardsTeamAddress	Public	✓	onlyManager
	setBlocked	Public	✓	onlyGovernance
	setBot	Public	✓	onlyGovernance
	setPair	Public	✓	onlyManager
	setEnableAntiBot	External	✓	onlyManager
	setAntiBotAddr	External	✓	onlyManager
	setGovernance	Public	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
sendValue
functionCallWithValue
functionCall
functionStaticCall
functionDelegateCall
verifyCallResult
isContract
Address (lib)
target
staticcall
delegatecall
renounceOwnership
transferOwnership
onlyOwner
owner
_msgSender
Ownable
ZOOZToken
setBlocked
setBot
totalSupply
balanceOf
timestampOf
onlyGovernance
allowance
onlyManager
setManagerAddress
setRewardsTeamAddress
_transfer
_approve
setPair
setAntiBotAddr
setGovernance
holdDateHook
_getFees
isBot
isBlocked
stdTransfer
amount
sub
mul
onPreTransferCheck
setTokenOwner
Context
_msgData

Summary

There are some functions that can be abused by the owner like stopping transactions. 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 fee percentage decreases over time, as the time elapsed since the last transaction of the holder. If the time elapsed is less than or equal to 1 week, the fee percentage is 14%. After 1 week, the fee percentage decreases to 10% for the next 3 weeks (1 month total). After 3 months, the fee percentage decreases again to 5%, and after 6 months, the fee percentage decreases to 2%. Eventually, after more than 6 months the fee percentage reaches 0.

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

Cyberscope

Audit Report

zoozToken

March 2023

Github	https://github.com/coalichain/ZOOZToken/blob/main/contracts
Commit	3f1cacf62a5d372e80cc1e24978a79ec04b76ff4
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Review 1
Audit Updates 1
Source Files 1
Roles 1
Analysis 3
BC - Blacklists Addresses 3
Description 4
Recommendation 5
Diagnostics 6
US - Untrusted Source 6
Description 6
Recommendation 6
MVN - Misleading Variables Naming 6
Description 6
Recommendation 9
Functions Analysis 9
Inheritance Graph 10
Flow Graph 10
Summary 10
Disclaimer 11
About Cyberscope 15

Review

Contract Name	ZOOZToken
Repository	https://github.com/coalichain/ZOOZToken/blob/main/contracts
Commit	3f1cacf62a5d372e80cc1e24978a79ec04b76ff4
Testing Deploy	https://testnet.bscscan.com/address/0x6a15792f6c25efdc549330219a733e8ed460a6ab

Audit Updates

Initial Audit	16 Feb 2023 https://github.com/cyberscope-io/audits/blob/main/zooz/v1/audit.pdf
Corrected Phase 2	01 Mar 2023

Source Files

Filename	SHA256
contracts/ZOOZToken.sol	1f9681eb46946e1261e02d5073e28ababdead2c259176f39daf36460ee9a8ebd

Roles

The contract consist of three roles. The `owner`, `manager,` and `governor` roles.

The `Owner` has the authority to

- Grant or revokes the `governor` and the `manager` role.
- Renounce or Transfer ownership.
- Set blacklisted addresses.
- Set excluded from fees addresses.

The `Manager` has the authority to

- Transfer the `manager` role.
- Configure reward address.

The `Governor` has the authority to

- Set blacklisted addresses.
- Set excluded from fees addresses.

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	Unresolved

BC - Blacklists Addresses

Criticality	Medium
Location	contracts/ZOOZToken.sol#L466
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `setBlockedAddress` function.

```
function setBlockedAddress(address holderAddress, bool blocked) public
onlyGovernance()  {
  require(holderAddress != address(0), "HolderAddress can't be the zero
address");

      blockedAddresses[holderAddress][_msgSender()] = blocked;

  if(blocked) {
    emit AddressBlocked(holderAddress);
    return;
  }

  emit AddressUnblocked(holderAddress);
}
```

Recommendation

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
●	US	Untrusted Source	Unresolved
●	MVN	Misleading Variables Naming	Unresolved

US - Untrusted Source

Criticality	Critical
Location	contracts/ZOOZToken.sol#L203
Status	Unresolved

Description

The contract uses an external contract in order to determine the transaction's flow. The external contract is untrusted. As a result it may produce security issues and harm the transactions.

```
IPinkAntiBot public pinkAntiBot;
```

Recommendation

The contract should use a trusted external source. A trusted source could be either a commonly recognized or an audited contract. The pointing addresses should not be able to change after the initialization.

MVN - Misleading Variables Naming

Criticality	Minor / Informative
Location	contracts/ZOOZToken.sol#L210,383
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.

The contract utilizes the variable `botAddresses` and the function `_isItBotAddress`. These variables and the function implement an exclude from the fees mechanism.

```
mapping (address => mapping (address => bool)) internal botAddresses;

function _isItBotAddress(address addr) internal view returns(bool) {
    return botAddresses[addr][governance1Address]
        && botAddresses[addr][governance2Address]
        && botAddresses[addr][governance3Address];
}
```

Recommendation

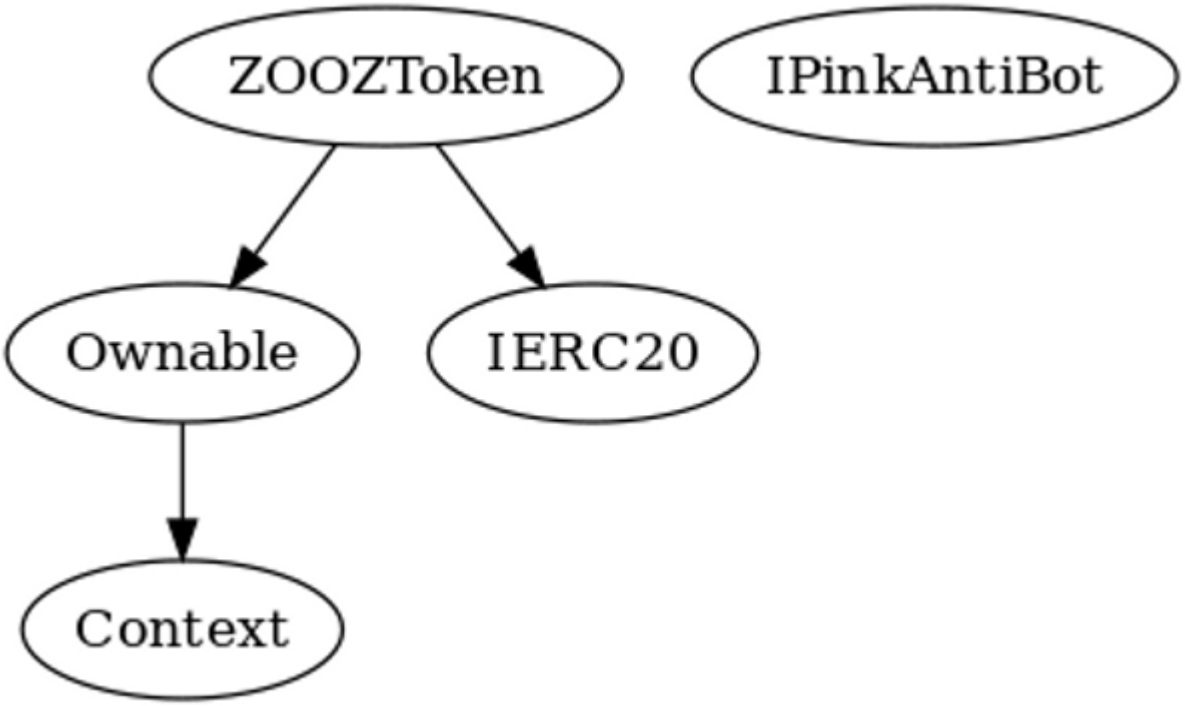
It's always a good practice for the contract to contain variable names that are specific and descriptive. The team is advised to keep in mind the readability of the code.

Functions Analysis

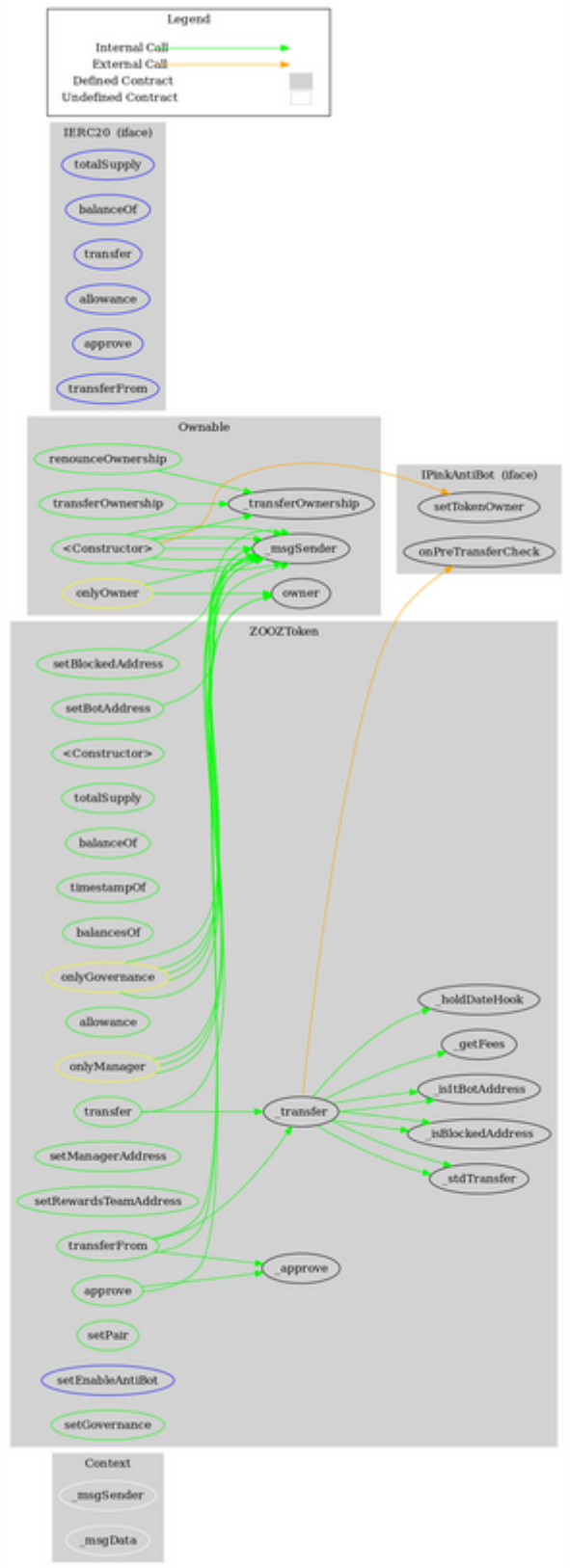
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IPinkAntiBot	Interface			
	setTokenOwner	External	✓	-
	onPreTransferCheck	External	✓	-

ZOOZToken	Implementation	Ownable, IERC20		
		Public	✓	-
	totalSupply	Public		-
	balanceOf	Public		-
	timestampOf	Public		-
	balancesOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_approve	Private	✓	
	_transfer	Private	✓	
	_isBlockedAddress	Internal		
	_isItBotAddress	Internal		
	_getFees	Internal		
	_holdDateHook	Internal	✓	
	_stdTransfer	Private	✓	
	setManagerAddress	Public	✓	onlyManager
	setRewardsTeamAddress	Public	✓	onlyManager
	setBlockedAddress	Public	✓	onlyGovernance
	setBotAddress	Public	✓	onlyGovernance
	setPair	Public	✓	onlyManager
	setEnableAntiBot	External	✓	onlyManager
	setGovernance	Public	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
renounceOwnership
transferOwnership
<Constructor>
onlyOwner
_transferOwnership
_msgSender
owner
IPinkAntiBot (iface)
setTokenOwner
onPreTransferCheck
ZOOZToken
setBlockedAddress
setBotAddress
<Constructor>
totalSupply
balanceOf
timestampOf
balancesOf
onlyGovernance
allowance
onlyManager
transfer
setManagerAddress
setRewardsTeamAddress
transferFrom
approve
setPair
setEnableAntiBot
setGovernance
_transfer
_approve
holdDateHook
_getFees
isItBotAddress
_isBlockedAddress
_stdTransfer
Context
_msgSender
_msgData

Summary

There are some functions that can be abused by the owner like stopping transactions. 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 fee percentage decreases over time, as the time elapsed since the last transaction of the holder. If the time elapsed is less than or equal to 1 week, the fee percentage is 14%. After 1 week, the fee percentage decreases to 10% for the next 3 weeks (1 month total). After 3 months, the fee percentage decreases again to 5%, and after 6 months, the fee percentage decreases to 2%. Eventually, after more than 6 months the fee percentage reaches 0.

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