Cyberscope

Audit Report

CICO DRAGON

February 2024

Network	BSC
Address	0x29395d312681fcccda6aa7a9ee79bc577b3700da
Audited by	© cyberscope

Analysis

🔴 Critical 🟠 Medium ⚪ Minor / Informative 🔵 Pass

Severity	Code	Description	Status
🔴	ST	Stops Transactions	Unresolved
🔵	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
●	CR	Code Repetition	Unresolved
●	DDP	Decimal Division Precision	Unresolved
●	MEM	Missing Error Messages	Unresolved
●	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
●	RFO	Redundant Function Overrides	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	RSD	Redundant Swap Duplication	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	CICO_DRAGON
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://bscscan.com/address/0x29395d312681fcccda6aa7a9ee79bc577b3700da
Address	0x29395d312681fcccda6aa7a9ee79bc577b3700da
Network	BSC
Symbol	CICO
Decimals	18
Total Supply	21,000,000,000
Badge Eligibility	Yes

Audit Updates

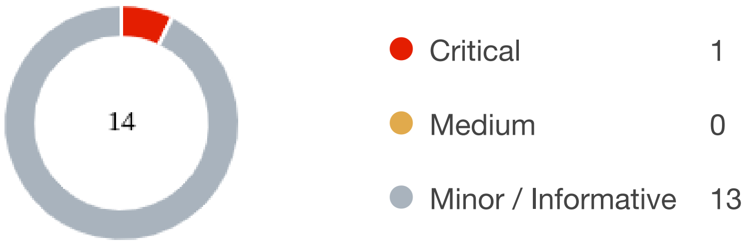
Initial Audit	06 Feb 2024

Source Files

Filename	SHA256
Token.sol	f8a4cab3de4c09b24713ef29bcf185358f9e9167dc90d85683352caaa9497071
Ownable2Step.sol	3e3bdb084bc14ade54e8259e710287956a7dbf2b2b4ad1e4cd8899d2293c7241

Ownable.sol	33422e7771fefe5fbfe89348375150971 19d82a50eda0e49b38e4d6a64a1c25d
Initializable.sol	b05c26d897c4178cbdb35ad113527e463e1bdeae5764869318a54f93c8b98a94
IUniswapV2Router02.sol	a2900701961cb0b6152fc073856b972564f7c798797a4a044e83d2ab8f0e8d38
IUniswapV2Router01.sol	0439ffe0fd4a5e1f4e22d71ddbda76d63d61679947d158cba4ee0a1da60cf663
IUniswapV2Pair.sol	29c75e69ce173ff8b498584700fef76bc81498c1d98120e2877a1439f0c31b5a
IUniswapV2Factory.sol	51d056199e3f5e41cb1a9f11ce581aa3e190cc982db5771ffeef8d8d1f962a0d
IERC20Metadata.sol	b10e2f8bcc3ed53a5d9a82a29b1ad3209225331bb4de4a0459862a762cf83a1a
IERC20.sol	7ebde70853ccafcf1876900dad458f46eb9444d591d39bfc58e952e2582f5587
ERC20Burnable.sol	480b22ce348050fdb85a693e38ed6b4767a94e4776fc6806d6808a0ec171177e
ERC20.sol	f70c6ae5f2dda91a37e17cfcbec390cc59515ed0d34e316f036f5431b5c0a3f2
Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
CoinDividendTracker.sol	6c5ffeff24461bad9a3bd144e494d44bc7321d600ec6318842ef2f2962cc9f15

Findings Breakdown



	Severity	Unresolved	Acknowledged	Resolved	Other
●	Critical	1	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	13	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	Token.sol#L426,447
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enable the owner will not be able to disable them again. Additionally, the contract incorporates an anti-bot mechanism that restricts users not excluded from limits from transacting more frequently than a specified cooldown period, up to 12 hours. While this feature aims to mitigate abusive trading practices, the lack of flexibility in re-disabling transactions or adjusting the anti-bot mechanism parameters post-enablement could limit the contract's adaptability to evolving security threats or operational requirements.

```
   function _beforeTokenTransfer(address from, address to, uint256
amount)
        internal
        override
    {
        if(!isExcludedFromLimits[from])
            require(lastTrade[from] + tradeCooldownTime <=
block.timestamp, "Antibot: Transaction sender is in anti-bot cooldown");
        if(!isExcludedFromLimits[to])
            require(lastTrade[to] + tradeCooldownTime <=
block.timestamp, "Antibot: Transaction recipient is in anti-bot
cooldown");

        // Interactions with DEX are disallowed prior to enabling
trading by owner
        if ((AMMPairs[from] && !isExcludedFromTradingRestriction[to]) ||
(AMMPairs[to] && !isExcludedFromTradingRestriction[from])) {
            require(tradingEnabled, "EnableTrading: Trading was not
enabled yet");
        }

        super._beforeTokenTransfer(from, to, amount);
    }
```

```
   function updateTradeCooldownTime(uint256 _tradeCooldownTime) public
onlyOwner {
        require(_tradeCooldownTime <= 12 hours, "Antibot: Trade cooldown
too long");

        tradeCooldownTime = _tradeCooldownTime;

        emit TradeCooldownTimeUpdated(_tradeCooldownTime);
    }
        function enableTrading() external onlyOwner {
        require(!tradingEnabled, "EnableTrading: Trading was enabled
already");
        tradingEnabled = true;

        emit TradingEnabled();
  }
```

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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

CR - Code Repetition

Criticality	Minor / Informative
Location	Token.sol#L180,236,266
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
   function marketingFeesSetup(uint16 _buyFee, uint16 _sellFee,
uint16 _transferFee) public onlyOwner {
        totalFees[0] = totalFees[0] - marketingFees[0] +
_buyFee;
        totalFees[1] = totalFees[1] - marketingFees[1] +
_sellFee;
        totalFees[2] = totalFees[2] - marketingFees[2] +
_transferFee;
        require(totalFees[0] <= 2500 && totalFees[1] <= 2500 &&
totalFees[2] <= 2500, "TaxesDefaultRouter: Cannot exceed max
total fee of 25%");

        marketingFees = [_buyFee, _sellFee, _transferFee];

        emit marketingFeesUpdated(_buyFee, _sellFee,
_transferFee);
    }

    function autoBurnFeesSetup(uint16 _buyFee, uint16 _sellFee,
uint16 _transferFee) public onlyOwner {
        ...
    }

    function liquidityFeesSetup(uint16 _buyFee, uint16
_sellFee, uint16 _transferFee) public onlyOwner {
        ...
    }

    function rewardsFeesSetup(uint16 _buyFee, uint16 _sellFee,
uint16 _transferFee) public onlyOwner {
        ...
    }
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	Token.sol#L329
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
_marketingPending += fees * marketingFees[txType] /
totalFees[txType];

if (autoBurnFees[txType] > 0) {
    autoBurnPortion = fees * autoBurnFees[txType] /
totalFees[txType];
    _burn(from, autoBurnPortion);
        emit autoBurned(autoBurnPortion);
    }

    _liquidityPending += fees * liquidityFees[txType] /
totalFees[txType];

    _rewardsPending += fees * rewardsFees[txType] /
totalFees[txType];
```

Recommendation

The team is advised to take into consideration the rounding results that are produced from the solidity calculations. The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

MEM - Missing Error Messages

Criticality	Minor / Informative
Location	CoinDividendTracker.sol#L191
Status	Unresolved

Description

The contract is using missing error messages. There is no error message do accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(totalSupply() > 0)
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	Token.sol#L145
Status	Unresolved

Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
    function _swapTokensForCoin(uint256 tokenAmount) private {
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = routerV2.WETH();

        _approve(address(this), address(routerV2), tokenAmount);


routerV2.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount, 0, path, address(this), block.timestamp);
    }
```

Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

RFO - Redundant Function Overrides

Criticality	Minor / Informative
Location	Token.sol#L141
Status	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 overrides the `decimals` function without changing the default function definition. Hence, the override is redundant.

```
    function decimals() public pure override returns (uint8) {
        return 18;
    }
```

Recommendation

The team is advised to take the above segments into consideration 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.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	CoinDividendTracker.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.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	Token.sol#L289,416
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
   function excludeFromFees(address account, bool isExcluded)
public onlyOwner {
        isExcludedFromFees[account] = isExcluded;

        emit ExcludeFromFees(account, isExcluded);
    }

    function excludeFromLimits(address account, bool
isExcluded) external onlyOwner {
        _excludeFromLimits(account, isExcluded);
    }
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

RSD - Redundant Swap Duplication

Criticality	Minor / Informative
Location	Token.sol#L202,248,348
Status	Unresolved

Description

The contract contains multiple swap methods that individually perform token swaps and transfer promotional amounts to specific addresses and features. This redundant duplication of code introduces unnecessary complexity and increases dramatically the gas consumption. By consolidating these operations into a single swap method, the contract can achieve better code readability, reduce gas costs, and improve overall efficiency.

```
_swapTokensForCoin(token2Swap);
```

Recommendation

A more optimized approach could be adopted to perform the token swap operation once for the total amount of tokens and distribute the proportional amounts to the corresponding addresses, eliminating the need for separate swaps.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Token.sol#L31,55,67,68,69,71,72,74,75,78,79,135,155,170,180,191,236,266,426 IUniswapV2Router01.sol#L5 IUniswapV2Pair.sol#L18,19,36 CoinDividendTracker.sol#L175,366,499
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.

```
contract CICO_DRAGON is ERC20, ERC20Burnable, Ownable2Step,
DividendTrackerFunctions, Initializable {

    uint16 public swapThresholdRatio;

    uint256 private _marketingPending;
    uint256 private _liquidityPending;
...
    {
        if (AMMPairs[from] && !isExcludedFromLimits[to])
lastTrade[to] = block.timestamp;
        else if (AMMPairs[to] && !isExcludedFromLimits[from])
lastTrade[from] = block.timestamp;

        super._afterTokenTransfer(from, to, amount);
    }
}

...
```

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.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	CoinDividendTracker.sol#L184
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
string memory _name
string memory _symbol
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	Ownable2Step.sol#L36
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.

```
_pendingOwner = newOwner
```

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.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	Token.sol#L18 Ownable2Step.sol#L4 Ownable.sol#L4 IUniswapV2Router02.sol#L1 IUniswapV2Router01.sol#L1 IUniswapV2Pair.sol#L1 IUniswapV2Factory.sol#L1 Initializable.sol#L3 IERC20Metadata.sol#L4 IERC20.sol#L4 ERC20Burnable.sol#L4 ERC20.sol#L4 Context.sol#L4 CoinDividendTracker.sol#L6
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity 0.8.19;
pragma solidity >=0.5.0;
pragma solidity >=0.6.2;
pragma solidity ^0.8.0;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Ownable2Step.sol#L4 Ownable.sol#L4 Initializable.sol#L3 IERC20Metadata.sol#L4 IERC20.sol#L4 ERC20Burnable.sol#L4 ERC20.sol#L4 Context.sol#L4 CoinDividendTracker.sol#L6
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 ^0.8.0;
```

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
CICO_DRAGON	Implementation	ERC20, ERC20Burnable, Ownable2Step, DividendTrackerFunctions, Initializable		
		Public	✓	ERC20
	initialize	External	✓	initializer
		External	Payable	-
	decimals	Public		-
	_swapTokensForCoin	Private	✓	
	updateSwapThreshold	Public	✓	onlyOwner
	getSwapThresholdAmount	Public		-
	getAllPending	Public		-
	marketingAddressSetup	Public	✓	onlyOwner
	marketingFeesSetup	Public	✓	onlyOwner
	autoBurnFeesSetup	Public	✓	onlyOwner
	_swapAndLiquify	Private	✓	
	_addLiquidity	Private	✓	
	addLiquidityFromLeftoverTokens	External	✓	-
	liquidityFeesSetup	Public	✓	onlyOwner

	_sendDividends	Private	✓	
	excludeFromDividends	External	✓	onlyOwner
	_excludeFromDividends	Internal	✓	
	rewardsFeesSetup	Public	✓	onlyOwner
	_burn	Internal	✓	
	_mint	Internal	✓	
	excludeFromFees	Public	✓	onlyOwner
	_transfer	Internal	✓	
	_updateRouterV2	Private	✓	
	setAMMPair	External	✓	onlyOwner
	_setAMMPair	Private	✓	
	excludeFromLimits	External	✓	onlyOwner
	_excludeFromLimits	Internal	✓	
	updateTradeCooldownTime	Public	✓	onlyOwner
	enableTrading	External	✓	onlyOwner
	excludeFromTradingRestriction	Public	✓	onlyOwner
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
Ownable2Step	Implementation	Ownable		
	pendingOwner	Public		-
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	

	acceptOwnership	Public	✓	-
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Initializable	Implementation			
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	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		-
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	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-

ERC20Burnable	Implementation	Context, ERC20		
	burn	Public	✓	-
	burnFrom	Public	✓	-
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	✓	

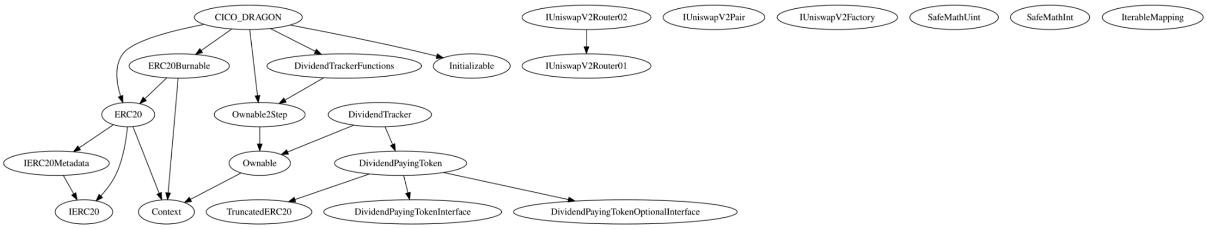
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
SafeMathUint	Library			
	toInt256Safe	Internal		
SafeMathInt	Library			
	toUint256Safe	Internal		
TruncatedERC20	Implementation			
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	_mint	Internal	✓	
	_burn	Internal	✓	

DividendPayingTokenInterface	Interface			
	dividendOf	External		-
DividendPayingTokenOptionalInterface	Interface			
	withdrawableDividendOf	External		-
	withdrawnDividendOf	External		-
	accumulativeDividendOf	External		-
DividendPayingToken	Implementation	TruncatedERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
		Public	✓	TruncatedERC20
		External	Payable	-
	distributeDividends	Public	Payable	-
	_withdrawDividend	Internal	✓	
	dividendOf	Public		-
	withdrawableDividendOf	Public		-
	withdrawnDividendOf	Public		-
	accumulativeDividendOf	Public		-
	_mint	Internal	✓	
	_burn	Internal	✓	

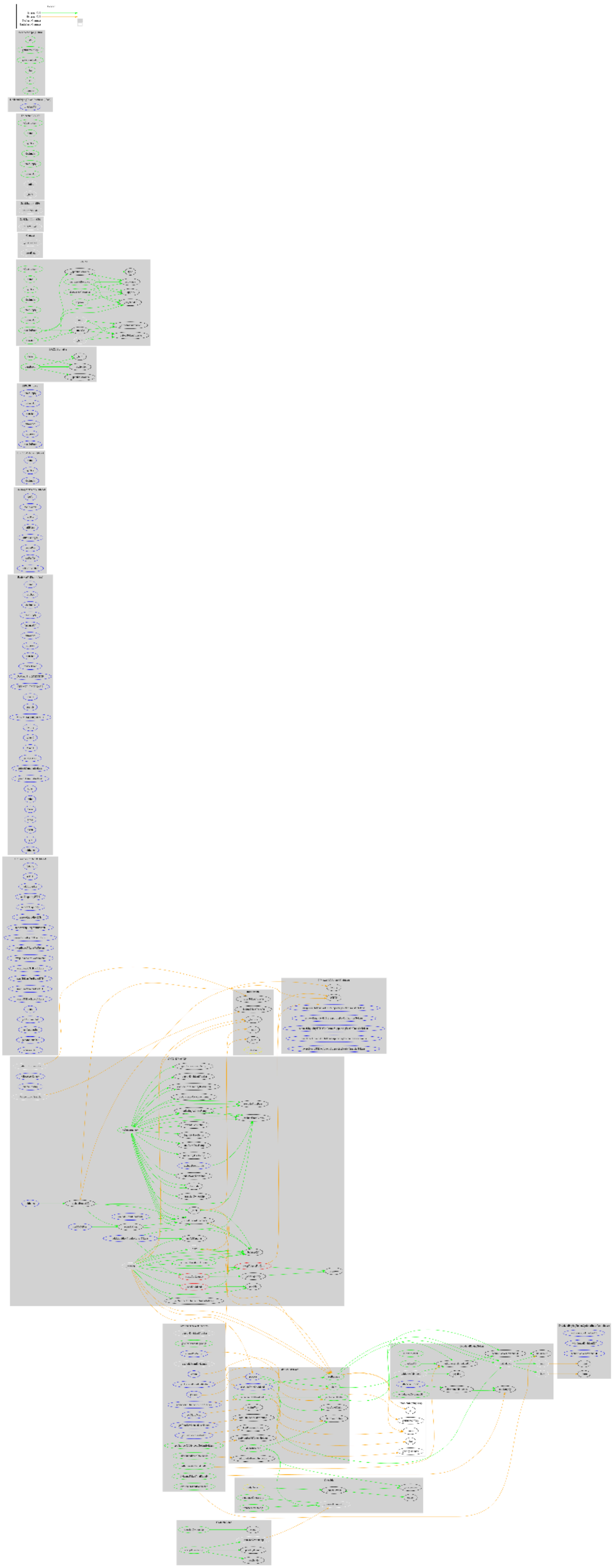
	_setBalance	Internal	✓	
IterableMapping	Library			
	get	Public		-
	getIndexOfKey	Public		-
	getKeyAtIndex	Public		-
	size	Public		-
	set	Public	✓	-
	remove	Public	✓	-
DividendTracker	Implementation	Ownable, DividendPayingToken		
		Public	✓	DividendPayingToken
	excludeFromDividends	External	✓	onlyOwner
	claimWaitSetup	Public	✓	onlyOwner
	getNumberOfTokenHolders	External		-
	getAccountData	Public		-
	getAccountDataAtIndex	Public		-
	claim	Public	✓	onlyOwner
	_canAutoClaim	Private		
	setBalance	Public	✓	onlyOwner
	process	External	✓	onlyOwner

DividendTrackerFunctions	Implementation	Ownable2Step		
	_deployDividendTracker	Internal	✓	
	gasForProcessingSetup	Public	✓	onlyOwner
	claimWaitSetup	External	✓	onlyOwner
	_excludeFromDividends	Internal	✓	
	isExcludedFromDividends	Public		-
	claim	External	✓	-
	getClaimWait	External		-
	getTotalDividendsDistributed	External		-
	withdrawableDividendOf	Public		-
	dividendTokenBalanceOf	Public		-
	dividendTokenTotalSupply	Public		-
	getAccountDividendsInfo	External		-
	getAccountDividendsInfoAtIndex	External		-
	getLastProcessedIndex	External		-
	getNumberOfDividendTokenHolders	Public		-
	process	External	✓	-

Inheritance Graph



Flow Graph



Summary

CICO DRAGON 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 stop 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. 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