Cyberscope

Audit Report

DirtiCoin

September 2022

SHA256 1d79c13b531a60d627fc5efe47bb2f113dfe285ef6363d234b83cd8bd4513de3

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	DIDToken
Compiler Version	v0.8.6+commit.11564f7e
Optimization	200 runs
Testing Deploy	https://testnet.bscscan.com/address/0x2080dcbaa63743c83ef6e7d1a10ceab3c995736a
Testing Upgradeable Proxy Deploy	https://testnet.bscscan.com/address/0x7665f8ED94667d3C59C57deA83aBd2aD46e344CF
Decimals	18

Audit Updates

Initial Audit	2nd September 2022
Corrected	

Source Files

Filename	SHA256
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	da66c17044345dc892d85bd7ddc9745d25df0b3dacfba8f84eb87c60d6e40fe3
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	cd823c76cbf5f5b6ef1bda565d58be66c843c37707cd93eb8fb5425deebd6756
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	36a6477c6263d9441dab59861e0ca97a201caf2843598af2a8e04e897a738c2f
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-IERC20PermitUpgradeable.sol	b97515a88e75c313eacf0a27c9439ef371d86d4c2730d3b13076640942f813df
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol	68bcca423fc72ec9625e219c9e36306c726a347e43f3711467c579bd3f6500c8
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	4e09a7479aa3e7c313f8fc141c4c8fc04e0abfeb8754615ef7d78ec94c298b07

@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	b7410d275fc7d26e36b0851541d6ff290593ba72d64b5c906978124b123915c1
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	35fb271561f3dc72e91b3a42c6e40c2bb2e788cd8ca58014ac43f6198b8d32ca
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	5fb301961e45cb482fe4e05646d2f529aa449fe0e90c6671475d6a32356fa2d4
@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol	4039686a509394aed475619c4e0b3a2df1df34fe59e90b9add8669de371eb731
contracts/DIDToken.sol	1d79c13b531a60d627fc5efe47bb2f113dfe285ef6363d234b83cd8bd4513de3
contracts/libraries/ERC20TaxTokenU.sol	1ca5f7f64a461773836714302845a9dae020d41d5bb0ec01978c03b1e51a01b9

Contract 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	Unresolved
●	BT	Burns Tokens	Unresolved
●	BC	Blacklists Addresses	Passed

ELFM - Exceeds Fees Limit

Criticality	critical
Location	contract.sol/libraries/ERC20TaxTokenU.sol#L46,54
Status	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 updateTaxFees, updateTaxFeeById function with a high percentage value.

```
function updateTaxFees(TaxFee[] memory _taxFees) public onlyOwner {
        delete taxFees;
        for (uint i=0; i<_taxFees.length; i++) {
            taxFees.push(_taxFees[i]);
        }
        calcTotalFeeRate();
    }

function updateTaxFeeById(uint8 index, string memory name, address wallet, uint16 rate)
public onlyOwner {
        if (index == 255) {
            taxFees.push(TaxFee(name, wallet, rate));
        } else {
            taxFees[index] = TaxFee(name, wallet, rate);
        }
        calcTotalFeeRate();
    }
```

Recommendation

The contract could embody a check for the maximum acceptable value. The calcTransFee is using the basisFeePoint dominator that is fixed to 10000, so the updateTaxFees and updateTaxFeeById should check that the sum of the taxes are less than 2500.

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.

MT - Mints Tokens

Criticality	critical
Location	contract.sol#L39
Status	Unresolved

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `mint` function. As a result the contract tokens will be highly inflated.

```
function mint(address _to, uint256 _amount) external onlyOwner {
        require(_to != address(0x0), "zero address");
        _mint(_to, _amount);
    }
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

BT - Burns Tokens

Criticality	critical
Location	contract.sol#L44
Status	Unresolved

Description

The contract owner has the authority to burn tokens from a specific address. The owner may take advantage of it by calling the `burn` function. As a result the targeted contract address will lose the corresponding tokens.

```
function burn(address _from, uint256 _amount) external onlyOwner {
        require(_from != address(0x0), "zero address");
        _burn(_from, _amount);
    }
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

Contract Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	BLC	Business Logic Concern	Unresolved
●	L01	Public Function could be Declared External	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L05	Unused State Variable	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved

BLC - Business Logic Concern

Criticality	minor / informative
Location	contracts/libraries/ERC20TaxTokenU.sol#L54
Status	Unresolved

Description

The business logic seems peculiar. The implementation may not follow the expected behaviour.

Since the arguments index is uint8 (2^8) and the value 255 is calculated as an upper bound, then the `_taxFees` array should have an array length requirement to be less than 255. In the `_taxFees` argument of `updateTaxFees` and `taxFees.push` of `updateTaxFeeById`.

```
function updateTaxFeeById(uint8 index, string memory name, address wallet, uint16 rate) public
onlyOwner {
        if (index == 255) {
            taxFees.push(TaxFee(name, wallet, rate));
        } else {
            taxFees[index] = TaxFee(name, wallet, rate);
        }
        calcTotalFeeRate();
    }
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

L01 - Public Function could be Declared External

Criticality	minor / informative
Location	contracts/DIDToken.sol#L24 contracts/libraries/ERC20TaxTokenU.sol#L54
Status	Unresolved

Description

Public functions that are never called by the contract should be declared external to save gas.

```
initialize
updateTaxFeeById
```

Recommendation

Use the external attribute for functions never called from the contract.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contracts/libraries/ERC20TaxTokenU.sol#L63,34,67,46 contracts/DIDToken.sol#L28,39,44
Status	Unresolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_status
_taxFees
__TaxToken_init
_addr
_to
_amount
_from
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions.

L05 - Unused State Variable

Criticality	minor / informative
Location	contracts/DIDToken.sol#L8
Status	Unresolved

Description

There are segments that contain unused state variables.

```
DIDToken
```

Recommendation

Remove unused state variables.

L15 - Local Scope Variable Shadowing

Criticality	minor / informative
Location	contracts/libraries/ERC20TaxTokenU.sol#L54 contracts/DIDToken.sol#L25,26
Status	Unresolved

Description

The are variables that are defined in the local scope containing the same name from an upper scope.

```
name
symbol
```

Recommendation

The local variables should have different names from the upper scoped variables.

Contract Functions

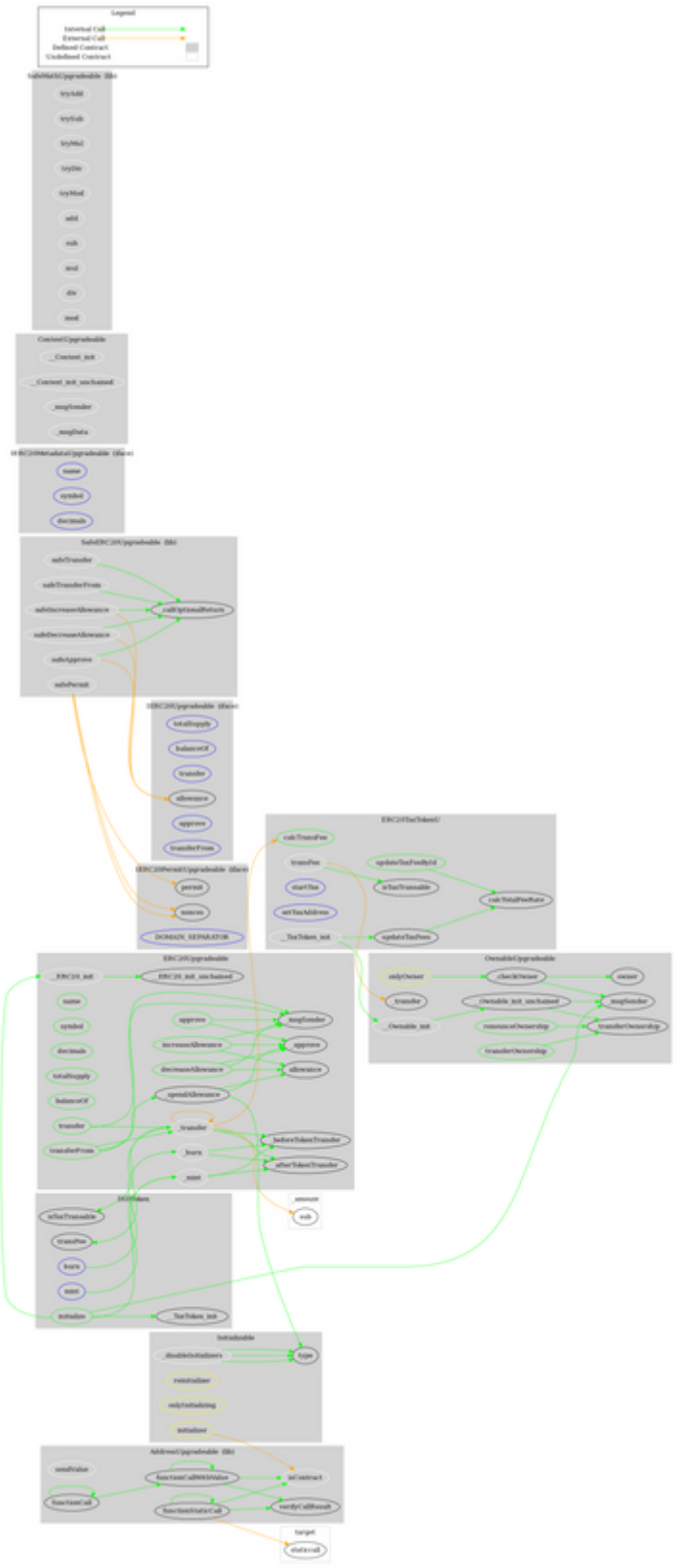
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
OwnableUpgradeable	Implementation	Initializable, ContextUpgradeable		
	__Ownable_init	Internal	✓	onlyInitializing
	__Ownable_init_unchained	Internal	✓	onlyInitializing
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Initializable	Implementation			
	_disableInitializers	Internal	✓	
ERC20Upgradeable	Implementation	Initializable, ContextUpgradeable, IERC20Upgradeable, IERC20MetadataUpgradeable		
	__ERC20_init	Internal	✓	onlyInitializing
	__ERC20_init_unchained	Internal	✓	onlyInitializing
	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	✓	
IERC20Permit Upgradeable	Interface			
	permit	External	✓	-
	nonces	External		-
	DOMAIN_SEPARATOR	External		-
IERC20Metad ataUpgradeabl e	Interface	IERC20Upg radeable		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20Upgrad eable	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeERC20Up gradeable	Library			

	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	safePermit	Internal	✓	
	_callOptionalReturn	Private	✓	
AddressUpgradeable	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	verifyCallResult	Internal		
ContextUpgradeable	Implementation	Initializable		
	__Context_init	Internal	✓	onlyInitializing
	__Context_init_unchained	Internal	✓	onlyInitializing
	_msgSender	Internal		
	_msgData	Internal		
SafeMathUpgradeable	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		
DIDToken	Implementation	ERC20TaxTokenU		
	initialize	Public	✓	initializer
	mint	External	✓	onlyOwner
	burn	External	✓	onlyOwner
	_transfer	Internal	✓	
ERC20TaxTokenU	Implementation	ERC20Upgradeable, OwnableUpgradeable		
	__TaxToken_init	Internal	✓	initializer
	updateTaxFees	Public	✓	onlyOwner
	updateTaxFeeById	Public	✓	onlyOwner
	startTax	External	✓	onlyOwner
	setTaxAddress	External	✓	onlyOwner
	calcTransFee	Public		-
	isTaxTransable	Public		-
	transFee	Internal	✓	
	calcTotalFeeRate	Private	✓	

Contract Flow



Summary

There are some functions that can be abused by the owner like manipulating fees, minting tokens and burning tokens. if the contract owner abuses the mint functionality, then the contract will be highly inflated. if the contract owner abuses the burn functionality, then the users could lost their tokens. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io