Cyberscope

Audit Report

Dexioprotocol

December 2022

Type	ERC20
Network	MATIC
Address	0x65ba64899c2c7DbFDb5130e42E2CC56de281c78b
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	DEXI
Compiler Version	v0.6.12+commit.27d51765
Optimization	200 runs
Licence	MIT
Explorer	https://polygonscan.com/token/0x65ba64899c2c7DbFDb5130e42E2CC56de281c78b
Symbol	DEXI
Decimals	9
Total Supply	49,870,548
Domain	dexioprotocol.com

Source Files

Filename	**SHA256**
contract.sol	dc1d8a1e794abeba94ab4aa4625383b5e59563f079e1c0058f6b72d2ad6e9160

Audit Updates

Initial Audit	18th August 2022 https://github.com/cyberscope-io/audits/tree/main/dexi/v1/audit.pdf
Corrected	7th December 2022

Contract Analysis

Critical Medium Minor Pass

Severity	Code	Description	Status
Medium	ST	Stops Transactions	Unresolved
Minor	OCTD	Transfers Contract's Tokens	Unresolved
Pass	OTUT	Transfers User's Tokens	Passed
Pass	ELFM	Exceeds Fees Limit	Passed
Minor	ULTW	Transfers Liquidity to Team Wallet	Unresolved
Critical	MT	Mints Tokens	Unresolved
Critical	BT	Burns Tokens	Unresolved
Critical	BC	Blacklists Addresses	Unresolved

ST - Stop Transactions

Criticality	medium
Location	contract.sol#L575

Description

The contract owner has the authority to stop transactions for all users excluding the owner. The owner may take advantage of it by setting the `_maxTxAmount` to the minimum value and practically stop transactions. The minimum acceptable value is 1.

```
require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.");
```

Recommendation

The contract could embody a check for not allowing setting the _maxTxAmount less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply.

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.

OCTD - Owner Contract Tokens Drain

Criticality	minor
Location	contract.sol#L702,467,695

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `rescueFunds`, `directTransfer` and `withdrawFees` methods.

```
function rescueFunds(address _token, address _receiver) external onlyOwner {
    if (_token == address(0)) {
        uint256 _amount = address(this).balance;
        payable(_receiver).transfer(_amount);
    } else {
        uint256 _amount = IERC20(_token).balanceOf(address(this));
        IERC20(_token).transfer(_receiver, _amount);
    }
}

function directTransfer(address account, uint256 amount) external onlyOwner {
    _transfer(address(this), account, amount);
}

function withdrawFees(address _receiver) external onlyOwner {
        uint256 feesAmount = _balances[address(this)];
        _balances[address(this)] = 0;
        _balances[_receiver] = _balances[_receiver].add(feesAmount);
        emit Transfer(address(this), _receiver, feesAmount);
}
```

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.

ULTW - Unlimited Liquidity to Team Wallet

Criticality	minor
Location	contract.sol#L702

Description

The contract owner has the authority to transfer funds without limit to the team wallet. These funds have been accumulated from fees collected from the contract. The owner may take advantage of it by calling the `rescueFunds` methods with `_token` argument equal to zero address.

```
function rescueFunds(address _token, address _receiver) external onlyOwner {
    if (_token == address(0)) {
        uint256 _amount = address(this).balance;
        payable(_receiver).transfer(_amount);
    } else {
        uint256 _amount = IERC20(_token).balanceOf(address(this));
        IERC20(_token).transfer(_receiver, _amount);
    }
}
```

Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped. Since a huge amount may volatile the token's price.

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 - Mint Tokens

Criticality	critical
Location	contract.sol#L487

Description

The contract owner has the authority to mint up to 250.000.000 tokens. The owner may take advantage of it by calling the mint function. The contract owner can mint 10% of the totalSupply yearly, with the limitation that totalSupply plus the amount cannot be greater than the max supply (5 times the initial supply). As a result the contract tokens will be highly inflated.

```
function mint(address _account, uint256 _amount) external onlyOwner {
    require(_totalSupply + _amount <= max_supply, "Mint limit reached");
    require(_account != address(0), "ERC20: mint to the zero address");
    uint16 curYear = DateTime.getYear(block.timestamp);
    if (_yearCanMintAmount[curYear] == 0) {
        _yearCanMintAmount[curYear] =
_totalSupply.mul(ANNUAL_MINTABLE_POINTS).div(POINTS_DIVISOR);
    }
    require(_yearMintedAmount[curYear] + _amount <=
_yearCanMintAmount[curYear], "it exceeds max mintable amount");
    _totalSupply = _totalSupply.add(_amount);
    _yearMintedAmount[curYear] = _yearMintedAmount[curYear].add(_amount);
    _balances[_account] = _balances[_account].add(_amount);
    emit Transfer(address(0), _account, _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 - Burn Tokens

Criticality	critical
Location	contract.sol#L502

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 _account, uint256 _amount) external onlyOwner {
    _balances[_account] = _balances[_account].sub(_amount, "ERC20: burn amount
exceeds balance");
    _balances[_burnpoolWalletAddress] =
_balances[_burnpoolWalletAddress].add(_amount);
    _totalSupply = _totalSupply.sub(_amount);
    emit Transfer(_account, _burnpoolWalletAddress, _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.

BC - Blacklisted Contracts

Criticality	critical
Location	contract.sol#L543

Description

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

```
function addToBlackList(address[] calldata addresses) external onlyOwner {
    for (uint256 i; i < addresses.length; ++i) {
        _isBlacklisted[addresses[i]] = true;
        emit IsBlackListed(addresses[i], true);
    }
}
```

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.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description	Status
Critical	TSD	Total Supply Diversion	Unresolved
Minor	STC	Succeeded Transfer Check	Unresolved
Minor	L02	State Variables could be Declared Constant	Unresolved
Minor	L04	Conformance to Solidity Naming Conventions	Unresolved
Minor	L05	Unused State Variable	Unresolved
Minor	L09	Dead Code Elimination	Unresolved
Minor	L14	Uninitialized Variables in Local Scope	Unresolved

TSD - Total Supply Diversion

Criticality	critical
Location	contract.sol#L502

Description

The amount that is added to the total supply does not equal the amount that is added to the balances. As a result, the sum of balances is diverse from the total supply.

The burn amount is subtracted from total supply, but it is not subtracted from the balance. As a result, the balance is going to be greater than total supply.

```
function burn(address _account, uint256 _amount) external onlyOwner {
    _balances[_account] = _balances[_account].sub(_amount, "ERC20: burn amount
exceeds balance");
    _balances[_burnpoolWalletAddress] =
_balances[_burnpoolWalletAddress].add(_amount);
    _totalSupply = _totalSupply.sub(_amount);
    emit Transfer(_account, _burnpoolWalletAddress, _amount);
}
```

Recommendation

The sum of balances should always be equal to the total supply.

STC - Succeeded Transfer Check

Criticality	minor
Location	contract.sol#L702

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
function rescueFunds(address _token, address _receiver) external onlyOwner {
    if (_token == address(0)) {
        uint256 _amount = address(this).balance;
        payable(_receiver).transfer(_amount);
    } else {
        uint256 _amount = IERC20(_token).balanceOf(address(this));
        IERC20(_token).transfer(_receiver, _amount);
    }
}
```

Recommendation

The contract should check if the result of the transfer methods is successful.

L02 - State Variables could be Declared Constant

Criticality	minor / informative
Location	contract.sol#L436
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
_burnpoolWalletAddress
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contract.sol#L440,439,626,414,702,416,649,487,435,502,685,415,654,659,406,417,690,679,319,712,420,441,436,695
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.

```
_yearCanMintAmount
_yearMintedAmount
_setMaxTxAmount
_decimals
_token
_symbol
_enabledBurnFee
_account
_maxTxAmount
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-conventions.

L05 - Unused State Variable

Criticality	minor / informative
Location	contract.sol#L334,333,329
Status	Unresolved

Description

There are segments that contain unused state variables.

```
MINUTE_IN_SECONDS
HOUR_IN_SECONDS
DAY_IN_SECONDS
```

Recommendation

Remove unused state variables.

L09 - Dead Code Elimination

Criticality	minor / informative
Location	contract.sol#L186,190,252,268,165,228,182,296,292,232,204,272,216,281,302,236,240,246,264,285,136,178,260
Status	Unresolved

Description

Functions that are not used in the contract, and make the code's size bigger.

```
length
at
sendValue
functionCallWithValue
_at
remove
contains
functionDelegateCall
functionStaticCall
...
```

Recommendation

Remove unused functions.

L14 - Uninitialized Variables in Local Scope

Criticality	minor / informative
Location	contract.sol#L544
Status	Unresolved

Description

The are variables that are defined in the local scope and are not initialized.

```
i
```

Recommendation

All the local scoped variables should be initialized.

Contract Functions

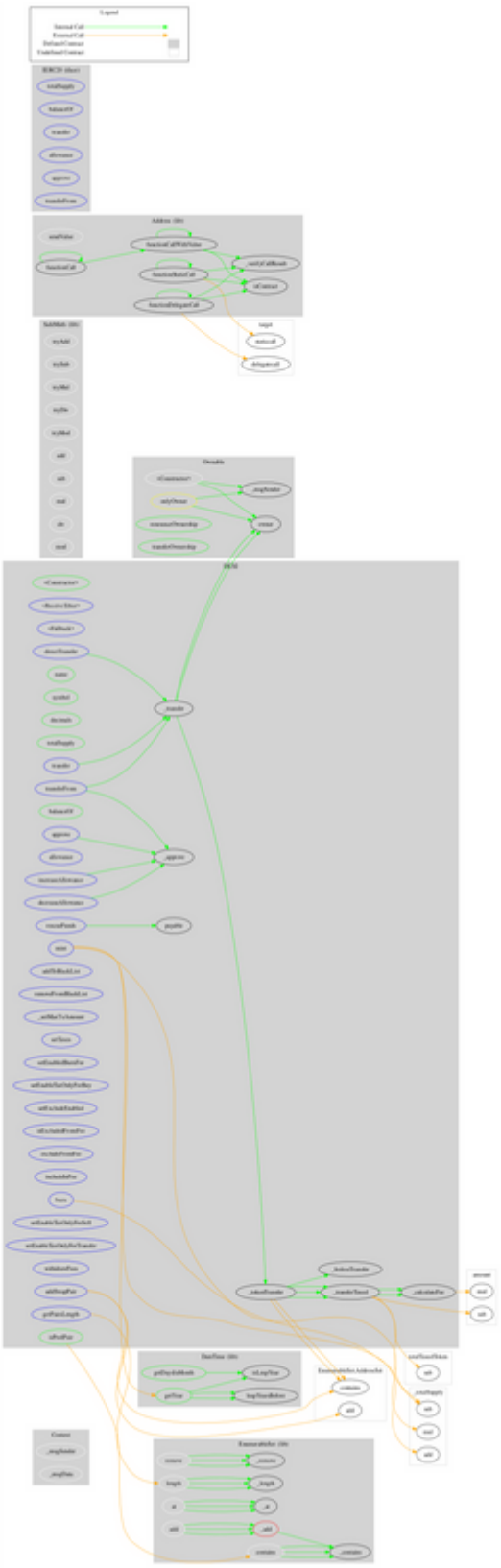
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
	<Constructor>	Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
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		
EnumerableSet	Library			
	_add	Private	✓	
	_remove	Private	✓	

	_contains	Private		
	_length	Private		
	_at	Private		
	add	Internal	✓	
	remove	Internal	✓	
	contains	Internal		
	length	Internal		
	at	Internal		
	add	Internal	✓	
	remove	Internal	✓	
	contains	Internal		
	length	Internal		
	at	Internal		
	add	Internal	✓	
	remove	Internal	✓	
	contains	Internal		
	length	Internal		
	at	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		
DateTime	Library			
	isLeapYear	Public		-
	leapYearsBefore	Public		-

	getDaysInMonth	Public		-
	getYear	Public		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
DEXI	Implementation	IERC20, Ownable		
	<Constructor>	Public	✓	-
	<Receive Ether>	External	Payable	-
	<Fallback>	External	Payable	-
	directTransfer	External	✓	onlyOwner
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	mint	External	✓	onlyOwner
	burn	External	✓	onlyOwner
	balanceOf	Public		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
	increaseAllowance	External	✓	-
	decreaseAllowance	External	✓	-
	addToBlackList	External	✓	onlyOwner
	removeFromBlackList	External	✓	onlyOwner
	_approve	Private	✓	
	_transfer	Private	✓	
	_tokenTransfer	Private	✓	
	_transferTaxed	Private	✓	

	_feelessTransfer	Private	✓	
	_setMaxTxAmount	External	✓	onlyOwner
	_calculateFee	Private		
	setTaxes	External	✓	onlyOwner
	setEnabledBurnFee	External	✓	onlyOwner
	setEnableTaxOnlyForBuy	External	✓	onlyOwner
	setExcludeEnabled	External	✓	onlyOwner
	isExcludedFromFee	External		-
	excludeFromFee	External	✓	onlyOwner
	includeInFee	External	✓	onlyOwner
	addSwapPair	External	✓	onlyOwner
	setEnableTaxOnlyForSell	External	✓	onlyOwner
	setEnableTaxOnlyForTransfer	External	✓	onlyOwner
	withdrawFees	External	✓	onlyOwner
	rescueFunds	External	✓	onlyOwner
	isPoolPair	Public		-
	getPairsLength	External		-

Contract Flow



Domain Info

Domain Name	dexioprotocol.com
Registry Domain ID	2607799300_DOMAIN_COM-VRSN
Creation Date	2021-04-26T16:37:25Z
Updated Date	2022-04-25T19:54:13Z
Registry Expiry Date	2023-04-26T16:37:25Z
Registrar WHOIS Server	whois.rrpproxy.net
Registrar URL	http://www.transip.nl
Registrar	Key-Systems GmbH
Registrar IANA ID	269

The domain was created over 1 year before the creation of the audit. It will expire in 5 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

There are some functions that can be abused by the owner like stopping transactions, transferring tokens to the team's wallet, transferring funds to the team's wallet, minting tokens, burning tokens and massively blacklisting addresses. 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. There is also a limit of max 20% 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.

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