Cyberscope

# Audit Report

# Dexioprotocol

August 2022

| | |
|---|---|
| Type | ERC20 |
| Network | CRONOS |
| Address | 0x3e1bEF464F2600eF94dE4d965c8991e06570d2e8 |
| Audited by | © cyberscope |

# Table of Contents

Table of Contents 1
Contract Review 3
Source Files 3
Audit Updates 3
Contract Analysis 4
ST - Stop Transactions 5
Description 5
Recommendation 5
OCTD - Owner Contract Tokens Drain 6
Description 6
Recommendation 6
ULTW - Unlimited Liquidity to Team Wallet 7
Description 7
Recommendation 7
MT - Mint Tokens 8
Description 8
Recommendation 8
BT - Burn Tokens 9
Description 9
Recommendation 9
BC - Blacklisted Contracts 10
Description 10
Recommendation 10
Contract Diagnostics 11
CBD - Contract Balance Diversion 12
Description 12

Recommendation 12
STC - Succeeded Transfer Check 13
Description 13
Recommendation 13
L01 - Public Function could be Declared External 14
Description 14
Recommendation 14
L02 - State Variables could be Declared Constant 15
Description 15
Recommendation 15
L04 - Conformance to Solidity Naming Conventions 16
Description 16
Recommendation 16
L05 - Unused State Variable 17
Description 17
Recommendation 17
L09 - Dead Code Elimination 18
Description 18
Recommendation 18
L14 - Uninitialized Variables in Local Scope 19
Description 19
Recommendation 19
Contract Functions 20
Contract Flow 24
Domain Info 25
Summary 26
Disclaimer 27
About Cyberscope 28

# Contract Review

| | |
|---|---|
| **Contract Name** | DEXI |
| **Compiler Version** | v0.6.12+commit.27d51765 |
| **Deploy** | https://cronoscan.com/address/0x3e1bef464f2600ef94de4d965c8991e06570d2e8 |
| **Symbol** | DEXI |
| **Decimals** | 9 |
| **Total Supply** | 50,000,000 |
| **Domain** | https://dexioprotocol.com |

# Source Files

| **Filename** | **SHA256** |
|---|---|
| **contract.sol** | 2e54c1010c15d9e55c17754f6bf9262e91b3b6f5f6ba74a78356d7c084a6d8d7 |

# Audit Updates

| | |
|---|---|
| **Initial Audit** | 18th August 2022 |
| **Corrected** | |

# Contract Analysis

Critical Medium Minor Pass

| Severity | Code | Description |
|---|---|---|
| ● | ST | Contract Owner is not able to stop or pause transactions |
| ● | OCTD | Contract Owner is not able to transfer tokens from specific address |
| ● | OTUT | Owner Transfer User's Tokens |
| ● | ELFM | Contract Owner is not able to increase fees more than a reasonable percent (25%) |
| ● | ULTW | Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent |
| ● | MT | Contract Owner is not able to mint new tokens |
| ● | BT | Contract Owner is not able to burn tokens from specific wallet |
| ● | BC | Contract Owner is not able to blacklist wallets from selling |

# ST - Stop Transactions

| | |
|---|---|
| **Criticality** | medium |
| **Location** | contract.sol#L576 |

## 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.

```
function _transfer(
        address from,
        address to,
        uint256 amount
    ) private {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");
        require(amount > 0, "Transfer amount must be greater than zero");

        require(!_isBlacklisted[from] && !_isBlacklisted[to], "This address is blacklisted");

        if(from != owner() && to != owner())
            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#L709,470,696 |

## 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#L696 |

## 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#L490 |

## Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the mint function. The contract owner can mint 10% of the totalSupply yearly. As a result the contract tokens will be highly inflated.

```
function mint(address _account, uint256 _amount) external onlyOwner {
    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#L503 |

## 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#L544 |

## 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 |
|---|---|---|
| ● | CBD | Contract Balance Diversion |
| ● | STC | Succeeded Transfer Check |
| ● | L01 | Public Function could be Declared External |
| ● | L02 | State Variables could be Declared Constant |
| ● | L04 | Conformance to Solidity Naming Conventions |
| ● | L05 | Unused State Variable |
| ● | L09 | Dead Code Elimination |
| ● | L14 | Uninitialized Variables in Local Scope |

# CBD - Contract Balance Diversion

| | |
|---|---|
| **Criticality** | critical |
| **Location** | contract.sol#L505 |

## Description

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

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 team is advised to carefully check if the implementation follows the expected behaviour.

# STC - Succeeded Transfer Check

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L709 |

## 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.

# L01 - Public Function could be Declared External

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L375,474,39,482,478,713,486,44,510,360 |

## Description

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

```
getDaysInMonth
balanceOf
transferOwnership
totalSupply
isPoolPair
symbol
decimals
renounceOwnership
name
...
```

## Recommendation

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

# L02 - State Variables could be Declared Constant

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L439 |

## 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 |
| **Location** | contract.sol#L443,442,627,418,703,420,650,490,438,503,686,419,655,660,410,691,680,323,713,423,444,439,696 |

## 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.

```
_receiver
_burnpoolWalletAddress
_isBlacklisted
_amount
initialSupply
_pair
_DateTime
_pairAddress
_enableTaxOnlyForTransfer
...
```

## 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 |
| **Location** | contract.sol#L338,337,333 |

## Description

There are segments that contain unused state variables.

```
DAY_IN_SECONDS
HOUR_IN_SECONDS
MINUTE_IN_SECONDS
```

## Recommendation

Remove unused state variables.

# L09 - Dead Code Elimination

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L190,194,256,272,169,232,186,300,296,236,208,276,220,285,306,240,244,250,268,289,140,182,264 |

## Description

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

```
functionCall
remove
_remove
functionStaticCall
isContract
at
length
_verifyCallResult
functionCallWithValue
...
```

## Recommendation

Remove unused functions.

# L14 - Uninitialized Variables in Local Scope

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L545 |

## 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

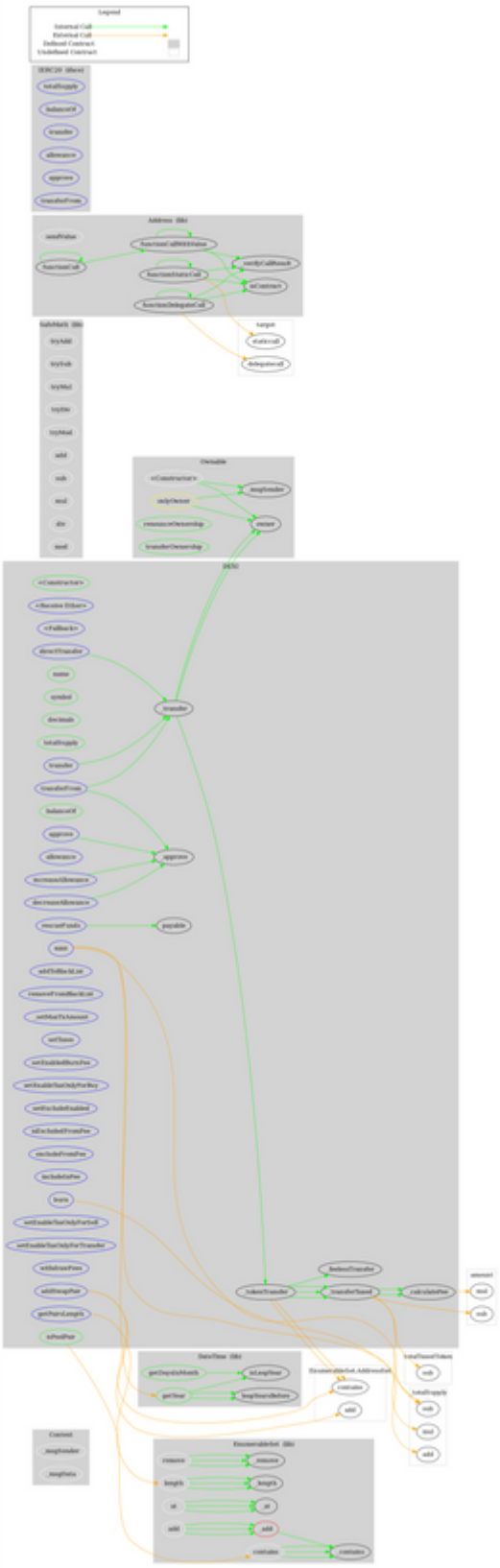
| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **Context** | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| **Ownable** | Implementation | Context | | |
| | <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 has been created in 8 months before the creation of the audit.

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. The users could lose 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