Cyberscope

Audit Report

# WKDLPPool

September 2022

| | |
|---|---|
| Type | BEP20 |
| Network | BSC |
| Address | 0x23459CA18cA4323356a2aC9C4d8297417798757A |
| Audited by | © cyberscope |

# Table of Contents

## Contract Review

| | |
|---|---|
| **Contract Name** | WKDLPPool |
| **Compiler Version** | v0.6.12+commit.27d51765 |
| **Optimization** | 999 runs |
| **Licence** | MIT |
| **Explorer** | https://bscscan.com/token/0x23459CA18cA4323356a2aC9C4d8297417798757A |
| **Domain** | - |

## Source Files

| **Filename** | **SHA256** |
|---|---|
| **contract.sol** | 51a74723cc2b0628d5078c83255b9dc7a3778598eb67ac561bb2db4f97d85aae |

## Audit Updates

| | |
|---|---|
| **Initial Audit** | 21st September 2022 |
| **Corrected** | |

# Introduction

The WKDLPPool contract implements a Liquidity Provider pool. Where users can deposit and withdraw liquidity provider tokens. Users can withdraw tokens without taking into consideration the rewards at any moment.

The contract has the authority to add and update pools. In addition, the contract owner has the authority to transfer all the contract tokens to him.

## Contract Diagnostics

🔴 Critical 🟠 Medium ● Minor / Informative

| Severity | Code | Description | Status |
|---|---|---|---|
| ● | OCTD | Transfers Contract's Tokens | Unresolved |
| ● | STC | Succeeded Transfer Check | Unresolved |
| ● | CO | Code Optimization | Unresolved |
| ● | L01 | Public Function could be Declared External | Unresolved |
| ● | L02 | State Variables could be Declared Constant | Unresolved |
| ● | L03 | Redundant Statements | Unresolved |
| ● | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| ● | L09 | Dead Code Elimination | Unresolved |
| ● | L13 | Divide before Multiply Operation | Unresolved |

## OCTD - Transfers Contract's Tokens

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L1343 |
| **Status** | Unresolved |

### 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 `emergencyRescue` function.

```
function emergencyRescue(uint256 _amount, address _token) public onlyOwner {
        IBEP20(_token).transfer(msg.sender, _amount);
}
```

### 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.

## STC - Succeeded Transfer Check

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L1343 |
| **Status** | Unresolved |

### 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 emergencyRescue(uint256 _amount, address _token) public onlyOwner {
        IBEP20(_token).transfer(msg.sender, _amount);
}
```

### Recommendation

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

## CO - Code Optimization

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L1350,999 |
| **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.

Since the first condition is true, then the `balance < _amount` will never fulfill.

```
function _safeTransfer(address _to, uint256 _amount) internal {
        if (_amount > 0) {
            if (WKD.balanceOf(address(this)) < _amount) {
                revert("Not enough WKD for rewards");
            }
            uint256 balance = WKD.balanceOf(address(this));
            if (balance < _amount) {
                _amount = balance;
            }
            WKD.safeTransfer(_to, _amount);
        }
    }
```

The update of the pool id should only happen if the entire pool is not updated.

```
function set(
        uint256 _pid,
        uint256 _allocPoint,
        bool _withUpdate
    ) external onlyOwner {
        // No matter _withUpdate is true or false, we need to execute updatePool once before set
the pool parameters.
        updatePool(_pid);

        if (_withUpdate) {
            massUpdatePools();
```

## Recommendation

Rewrite some code segments so the runtime will be more performant.

The update of the pool id could be executed only `if (!_withUpdate) updatePool(_pid);`

## L01 - Public Function could be Declared External

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L75,946,1339,84 |
| **Status** | Unresolved |

### Description

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

```
renounceOwnership
poolLength
emergencyRescue
transferOwnership
```

### Recommendation

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

## L02 - State Variables could be Declared Constant

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L887,859 |
| **Status** | Unresolved |

### Description

Constant state variables should be declared constant to save gas.

```
WKD_PER_BLOCK
burnAdmin
```

### Recommendation

Add the constant attribute to state variables that never change.

## L03 - Redundant Statements

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L16 |
| **Status** | Unresolved |

### Description

The contract contains statements that are not used and have no effect. As a result, those segments increase the code size of the contract unnecessarily.

```
Context
```

### Recommendation

Remove the redundant statements in order to decrease the code size.

## L04 - Conformance to Solidity Naming Conventions

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L1237,1339,960,961,997,998,1023,1308,1324,1216,1218,856,1074,1121,1323,1162,887,1089,1266,1249,1267,1217,959,958,1265,996,1322,1192 |
| **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.

```
_isValid
_token
_isRegular
_withUpdate
_allocPoint
_amount
_user
_pid
_boostMultiplier
...
```

### Recommendation

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

## L09 - Dead Code Elimination

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L763,545,596,655,645,621,745,726,631,571 |
| **Status** | Unresolved |

### Description

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

```
safeDecreaseAllowance
sendValue
functionCallWithValue
functionDelegateCall
functionStaticCall
safeIncreaseAllowance
safeApprove
functionCall
...
```

### Recommendation

Remove unused functions.

## L13 - Divide before Multiply Operation

| | |
|---|---|
| **Criticality** | minor / informative |
| **Location** | contract.sol#L1321,1023,1089,1121,1162,1264 |
| **Status** | Unresolved |

### Description

Performing divisions before multiplications may cause lose of prediction.

```
boostedAmount = user.amount.mul(_boostMultiplier).div(BOOST_PRECISION)
boostedAmount = user.amount.mul(getBoostMultiplier(_user,_pid)).div(BOOST_PRECISION)
wkdReward = multiplier.mul(wkdPerBlock(pool.isRegular)).mul(pool.allocPoint).div(totalAllocPoint)
user.rewardDebt =
user.amount.mul(multiplier).div(BOOST_PRECISION).mul(pool.accWkdPerShare).div(ACC_WKD_
PRECISION)
accWkdPerShare =
accWkdPerShare.add(wkdReward.mul(ACC_WKD_PRECISION).div(lpSupply))
user.rewardDebt =
user.amount.mul(_newMultiplier).div(BOOST_PRECISION).mul(pool.accWkdPerShare).div(ACC_
WKD_PRECISION)
```

### Recommendation

The multiplications should be prior to the divisions.

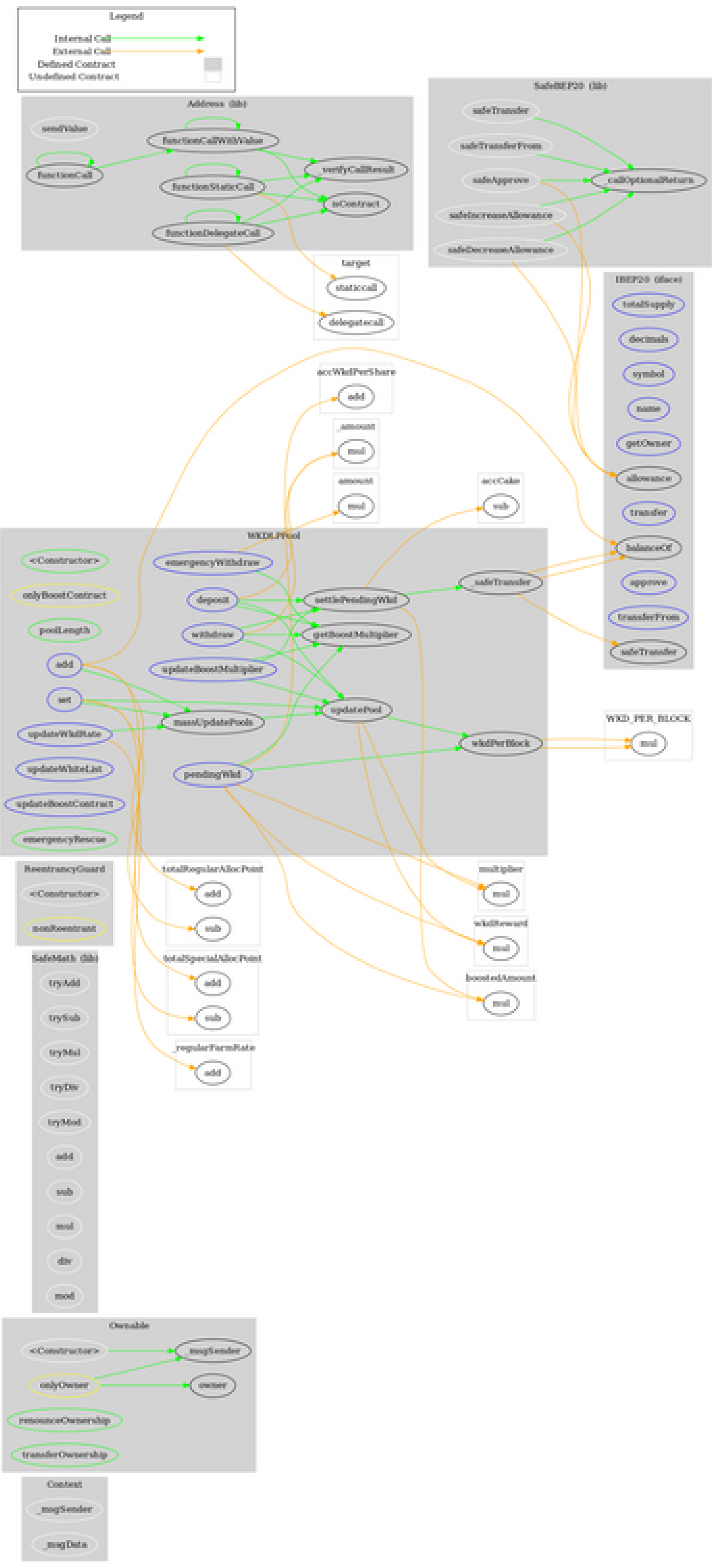
## Contract Functions

| 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 | | |
| | | | | |
| **ReentrancyGuard** | Implementation | | | |
| | <Constructor> | Internal | ✓ | |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **IBEP20** | Interface | | | |
| | totalSupply | External | | - |
| | decimals | External | | - |
| | symbol | External | | - |
| | name | External | | - |
| | getOwner | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| **Address** | Library | | | |
| | isContract | Internal | | |
| | sendValue | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | functionStaticCall | Internal | | |
| | functionStaticCall | Internal | | |
| | functionDelegateCall | Internal | ✓ | |
| | functionDelegateCall | Internal | ✓ | |
| | _verifyCallResult | Private | | |
| | | | | |
| **SafeBEP20** | Library | | | |
| | safeTransfer | Internal | ✓ | |
| | safeTransferFrom | Internal | ✓ | |
| | safeApprove | Internal | ✓ | |
| | safeIncreaseAllowance | Internal | ✓ | |
| | safeDecreaseAllowance | Internal | ✓ | |
| | _callOptionalReturn | Private | ✓ | |
| | | | | |
| **WKDLPPool** | Implementation | Ownable, Reentrancy | | |

| | | | | |
|---|---|---|---|---|
| | | Guard | | |
| | <Constructor> | Public | ✓ | - |
| | poolLength | Public | | - |
| | add | External | ✓ | onlyOwner |
| | set | External | ✓ | onlyOwner |
| | pendingWkd | External | | - |
| | massUpdatePools | Public | ✓ | - |
| | wkdPerBlock | Public | | - |
| | updatePool | Public | ✓ | - |
| | deposit | External | ✓ | nonReentrant |
| | withdraw | External | ✓ | nonReentrant |
| | emergencyWithdraw | External | ✓ | nonReentrant |
| | updateWkdRate | External | ✓ | onlyOwner |
| | updateWhiteList | External | ✓ | onlyOwner |
| | updateBoostContract | External | ✓ | onlyOwner |
| | updateBoostMultiplier | External | ✓ | onlyBoostContract nonReentrant |
| | getBoostMultiplier | Public | | - |
| | settlePendingWkd | Internal | ✓ | |
| | emergencyRescue | Public | ✓ | onlyOwner |
| | _safeTransfer | Internal | ✓ | |

## Contract Flow



## Summary

The WKDLPPool contract operates as a liquidity provider pool. There are some functions that can be abused by the owner like transferring tokens to the team’s wallet. We state that the owner privileges are necessary and required for proper protocol operations. Thus, we emphasize the contract owner be extra careful with the credentials.

## 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