Cyberscope

Audit Report

WKDLPPool

September 2022

Github	https://github.com/Bloceducare/WakandaInu-Contracts/tree/main/src
Commit	627de2245bb43b0de4ce357b33b38be7fc3a130e
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	WKDLPPool
Compiler Version	v0.6.12+commit.27d51765
Testing Deploy	https://testnet.bscscan.com/token/0xb5D0297EF84a0a9ceba4132f31FdBbB6521CaDf2
Domain	https://wakandainu.com

Audit Updates

Initial Audit	21st September 2022 https://github.com/cyberscope-io/audits/blob/main/wkd/wkdLpPool.pdf
Corrected	26th September 2022

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	b9f957b42bdcf3d3499be4c94558152e91658e34a1fe5a5e8f0972ce20e15ed7
@openzeppelin/contracts/math/SafeMath.sol	4a04d0a20a19e3ef1dcabae9cad9ba006430a4e7eec4d9b519db87999722c98a
@openzeppelin/contracts/utils/Address.sol	11ad5e3e21434e00c4ceba1f5a977b7a68bdd7d16b849276ce4ff4495129eec7
@openzeppelin/contracts/utils/Context.sol	9a3d1e5be0f0ace13e2d9aa1d0a1c3a6574983983ad5de94fc412f878bf7fe89
@openzeppelin/contracts/utils/ReentrancyGuard.sol	3fc7968f4a1937caf3c96dffbac350398f86faad96288502e02c3a2b9f245e39
contracts/farm/WKLDLPPool.sol	cf078e3627c628987573f5c32b42ac7e422ee20c30ae3819d14c82e6b7fa8aff
contracts/helpers/IBEP20.sol	5f8366fc3b9a5a8e25a639f2cf8534b5e017ffdce91c597dd7668e557c2fe272
contracts/helpers/SafeBEP20.sol	fa16115d3837e0e87ec528b29a4fbc0ee0bb3078ac075d06dd7cbfa4864acdf0

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, update pools and update WDK reward per block.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	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
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L13	Divide before Multiply Operation	Unresolved

STC - Succeeded Transfer Check

Criticality	minor / informative
Location	contract.sol#L553
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#L206
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 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	contracts/farm/WKLDLPPool.sol#L151,550,293
Status	Unresolved

Description

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

```
poolLength
emergencyRescue
updateWKDPERBLOCK
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor / informative
Location	contracts/farm/WKLDLPPool.sol#L64
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
burnAdmin
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contracts/farm/WKLDLPPool.sol#L448,550,165,166,202,203,228,293,519,535,427,429,61,279,332,534,373,92,300,477,460,478,428,164,163,476,201,533,403
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
_newRate
_pid
...
```

Recommendation

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

L07 - Missing Events Arithmetic

Criticality	minor / informative
Location	contracts/farm/WKLDLPPool.sol#L293
Status	Unresolved

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
WKD_PER_BLOCK = _newRate
```

Recommendation

Emit an event for critical parameter changes.

L13 - Divide before Multiply Operation

Criticality	minor / informative
Location	contracts/farm/WKLDLPPool.sol#L373,475,332,228,532,300
Status	Unresolved

Description

Performing divisions before multiplications may cause lose of prediction.

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

Recommendation

The multiplications should be prior to the divisions.

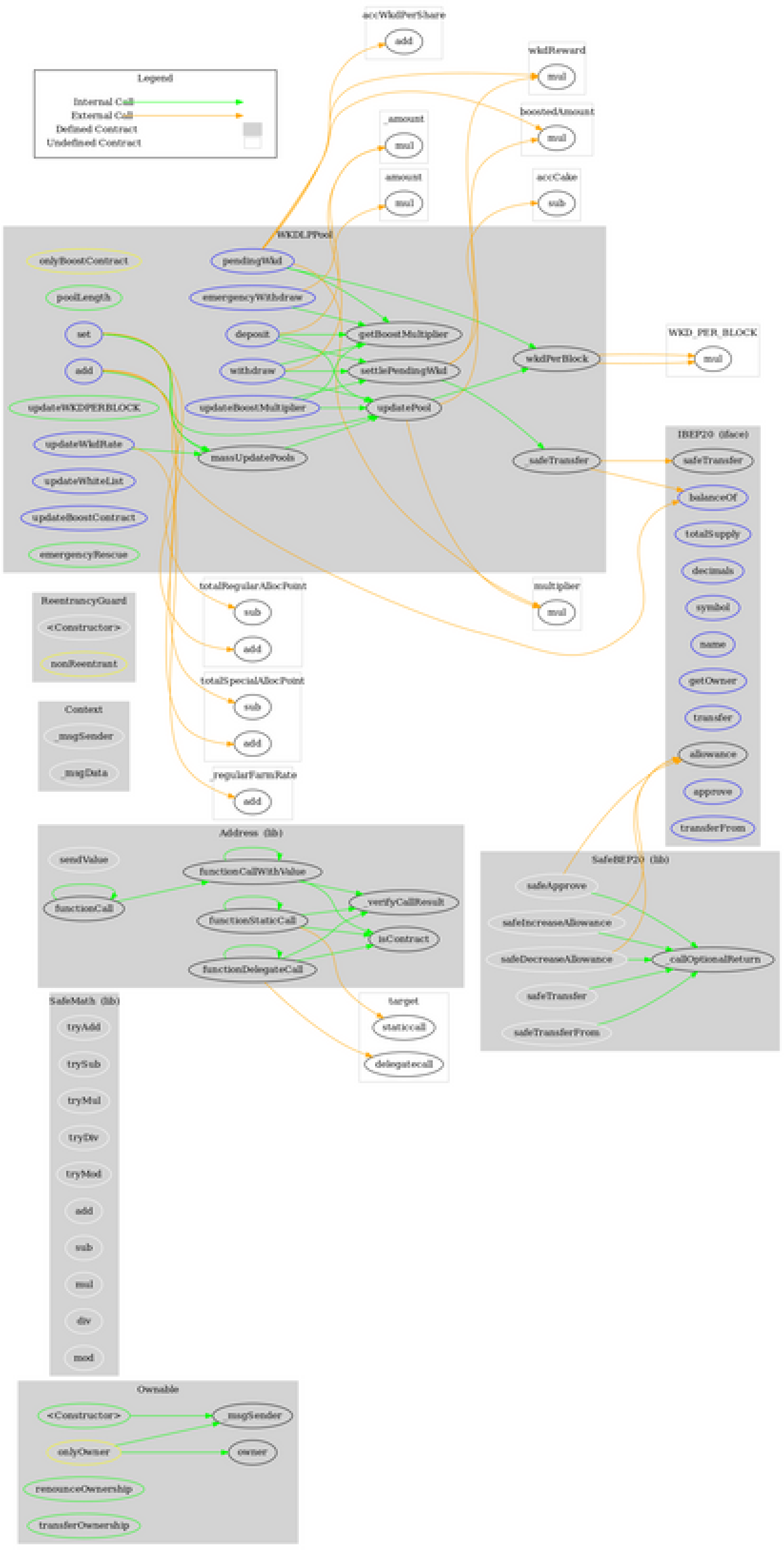
Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
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		
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	

	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	_verifyCallResult	Private		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
ReentrancyGuard	Implementation			
	<Constructor>	Internal	✓	
WKDLPPool	Implementation	Ownable, ReentrancyGuard		
	<Constructor>	Public	✓	-
	poolLength	Public		-
	add	External	✓	onlyOwner
	set	External	✓	onlyOwner
	pendingWkd	External		-
	massUpdatePools	Public	✓	-
	wkdPerBlock	Public		-
	updateWKDPERBLOCK	Public	✓	onlyOwner
	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	✓	
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeBEP20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	_callOptionalReturn	Private	✓	

Contract Flow



Domain Info

Domain Name	wakandainu.com
Registry Domain ID	2650366346_DOMAIN_COM-VRSN
Creation Date	2021-10-26T11:48:53.00Z
Updated Date	2021-11-11T12:32:24.22Z
Registry Expiry Date	2026-10-26T11:48:53.00Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	http://www.namecheap.com
Registrar	NAMECHEAP INC
Registrar IANA ID	1068

The domain was created 11 months before the creation of the audit. It will expire in about 4 years.

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

Summary

The WKDLPPool contract operates as a liquidity provider pool. 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