Cyberscope

Audit Report

RBX Staking

May 2022

Type	BEP20
Network	BSC
Address	0x864D434308997e9648838D23f3eedf5D0Fd17BEA
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	RocketDropV1point5
Compiler Version	v0.6.12+commit.27d51765
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0x864D434308997e9648838D23f3eedf5D0Fd17BEA

Source Files

Filename	**SHA256**
contract.sol	62031292daf63b70e4bdd3050e931f2c2615f9c61f880c0cfc41621b4acddc47

Audit Updates

Initial Audit	20th May 2022
Corrected	

Contract Diagnostics

🔴 Critical 🟠 Medium ● Minor

Severity	Code	Description
●	DSI	Data Structure Improvement
●	UUW	Stakers Counting Mismatch
●	EUP	Execution on Uninitialized Pools
●	STC	Succeeded Transfer Check
●	CO	Code Optimization
●	CR	Code Repetition
🟠	MC	Missing Check
●	L01	Public Function could be Declared External
●	L04	Conformance to Solidity Naming Conventions
●	L09	Dead Code Elimination

DSI - Data Structure Improvement

Criticality	minor
Location	contract.sol

Description

The poolInfo contains information regarding each staking pool indexed by the pid. The userInfo contains the user stake information indexed by the users address that is indexed by the pid. All the methods that access the userInfo, are accessing the poolInfo as well. Hence, the contract is keeping up to date two data structures with the same indexes.

```
// Info of each pool.
PoolInfo[] public poolInfo;
PoolExtras[] public poolExtras;

// Info of each user that stakes LP tokens.
mapping (uint256 => mapping (address => UserInfo)) public userInfo;
```

Recommendation

The contract could embed the userInfo mapping inside the poolInfo structure so there is no need for keeping up to date two indexes for data structures.

```
struct PoolInfo {
    IERC20 lpToken;
    uint256 lastRewardBlock;
    uint256 accERC20PerShare;
    IERC20 rewardToken;
    uint256 startBlock;
    uint256 endBlock;
    uint256 rewardPerBlock;
    uint256 paidOut;
    uint256 tokensStaked;
    uint256 gasAmount;
    uint256 minStake;
    uint256 maxStake;
    address payable partnerTreasury;
    uint256 partnerPercent;
    mapping (address => UserInfo) userInfo;
}
```

UUW - Stakers Counting Mismatch

Criticality	minor
Location	contract.sol#L1157

Description

The totalStakers is increased every time that a deposit is taking place. In contrast, the totalStakers counter is decreased if the remaining user's amount is zero. As a result, the totalStakers does not indicate any helpful information and conditions like `poolEx.totalStakers < poolEx.maxStakers` are useless.

```
poolEx.totalStakers = poolEx.totalStakers.add(1);
```

```
if(user.amount == 0){
    poolEx.totalStakers = poolEx.totalStakers.sub(1);
}
```

Recommendation

The contract should increase the *totalStakers* counter the first time that a user deposits an amount. The totalStakers should reflect the unique depositors.

EUP - Execution on Uninitialized Pools

Criticality	minor
Location	contract.sol

Description

The users have the authority to call methods with pid that contain indexes that have not been initialized yet.

```
function emergencyWithdraw(uint256 _pid) public {
    PoolInfo storage pool = poolInfo[_pid];
    PoolExtras storage poolEx = poolExtras[_pid];
    UserInfo storage user = userInfo[_pid][msg.sender];
    pool.lpToken.safeTransfer(address(msg.sender), user.amount);
    pool.tokensStaked = pool.tokensStaked.sub(user.amount);
    emit EmergencyWithdraw(msg.sender, _pid, user.amount);
    user.amount = 0;
    user.rewardDebt = 0;
    poolEx.totalStakers = poolEx.totalStakers.sub(1);
}
```

Recommendation

All the methods that accept the pid as parameter should initially check if the pid is less than the active pool's length.

STC - Succeeded Transfer Check

Criticality	minor
Location	contract.sol#L1219

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 erc20Transfer(address _to, uint256 _pid, uint256 _amount) internal {
    IERC20 erc20;
    erc20 = poolInfo[_pid].rewardToken;
    erc20.transfer(_to, _amount);
    poolInfo[_pid].paidOut += _amount;
}
```

Recommendation

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

CO - Code Optimization

Criticality	minor
Location	contract.sol#L1047

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.

There are methods that are solely accessing the data but are declaring variables as storage.

```
function deposited(uint256 _pid, address _user) external view returns (uint256)
{
    UserInfo storage user = userInfo[_pid][_user];
    return user.amount;
}
```

Recommendation

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

CR - Code Repetition

Criticality	minor
Location	contract.sol#L940,956,1147,1189

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily.

```
IERC20 erc20;
erc20 = poolInfo[_pid].rewardToken;

uint256 startTokenBalance = erc20.balanceOf(address(this));
erc20.safeTransferFrom(address(msg.sender), address(this), _amount);
uint256 endTokenBalance = erc20.balanceOf(address(this));
uint256 trueDepositedTokens = endTokenBalance.sub(startTokenBalance);
```

```
if (pool.partnerPercent == 0) {
    treasury.transfer(msg.value);
} else {
    uint256 totalAmount = msg.value;
    uint256 partnerAmount = totalAmount.mul(pool.partnerPercent).div(10000);
    uint256 treasuryAmount = totalAmount.sub(partnerAmount);
    treasury.transfer(treasuryAmount);
    pool.partnerTreasury.transfer(partnerAmount);
}
```

Recommendation

Create an internal function that contains the code segment and remove it from all the sections.

MC - Missing Check

Criticality	medium
Location	contract.sol

Description

Percentage Value Checks

The values that are used as percentages should be checked to not exceed the 100% value. Otherwise, the contract calculations will revert.

```
function changePartnerPercent(uint256 _pid, uint256 newPercent) public onlyOwner
{
    poolInfo[_pid].partnerPercent = newPercent;
}
```

For instance, the following segment will underflow.

```
uint256 partnerAmount = totalAmount.mul(pool.partnerPercent).div(10000);
uint256 treasuryAmount = totalAmount.sub(partnerAmount);
```

Normal Value Checks

The contract should check if the configured values may exploit the calculation results. For instance, if the result of `poolEx.lpTokenFee.mul(endTokenBalance).div(1000)` is greater than the `minStake` value, the expressions `endTokenBalance.sub(startTokenBalance).sub(depositFee);` will underflow.

Maximum Value Exceed

If variables like the `accessTokenMin`, `poolGasAmount` set to a high value, the users will not be able to withdraw their rewards.

If the `poolEx.accessToken` is set to the dead address, then the user's balance calculation will exploit.

```
if(poolEx.accessTokenRequired){
    require(poolEx.accessToken.balanceOf(msg.sender) >= poolEx.accessTokenMin,
'Must have minimum amount of access token!');
}
```

The first condition `_amount >= pool.minStake` checks if the amount is more than a threshold. The second condition checks if the user's amount is going to be more than a threshold. The contract should either check that the deposit value is between the thresholds or the user's amount is between the thresholds, but not both of them.

```
require(_amount >= pool.minStake && (_amount.add(user.amount)) <= pool.maxStake,
'Min/Max stake required!');
```

Recommendation

The contract should properly check the variables according to the required specifications

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L817,836,845,924,938,954,969,1004,1013,1017,1021,1025,1029,1033,1037,1041,1110,1162,1206,1227,1232,1235,1238,1243,1247,1251,1255,1259,1265,1269,1273,1277

Description

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

```
transfer
changePartnerPercent
changePartnerTreasury
changeTreasury
withdrawAnyToken
adjustLastBlock
adjustEndBlock
adjustBlockReward
adjustPoolGas
...
```

Recommendation

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

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L938,954,969,1004,1013,1017,1021,1025,1029,1033,1037,1041,1047,1053,1071,1089,1110,1162,1206,1219,1243,1247,1251,1255,1259,1269,1273

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.

```
_pid
_amount
_ERC20address
_recipient
_to
_user
_accessTokenMin
_accessToken
_accessTokenRequired
...
```

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
Location	contract.sol#L368,393,403,342,564,640,619,626,582,695,750,674,729,681,736,667,722,469,485,480

Description

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

```
safeIncreaseAllowance
safeDecreaseAllowance
safeApprove
remove
length
contains
at
_remove
_length
...
```

Recommendation

Remove unused functions.

Contract Functions

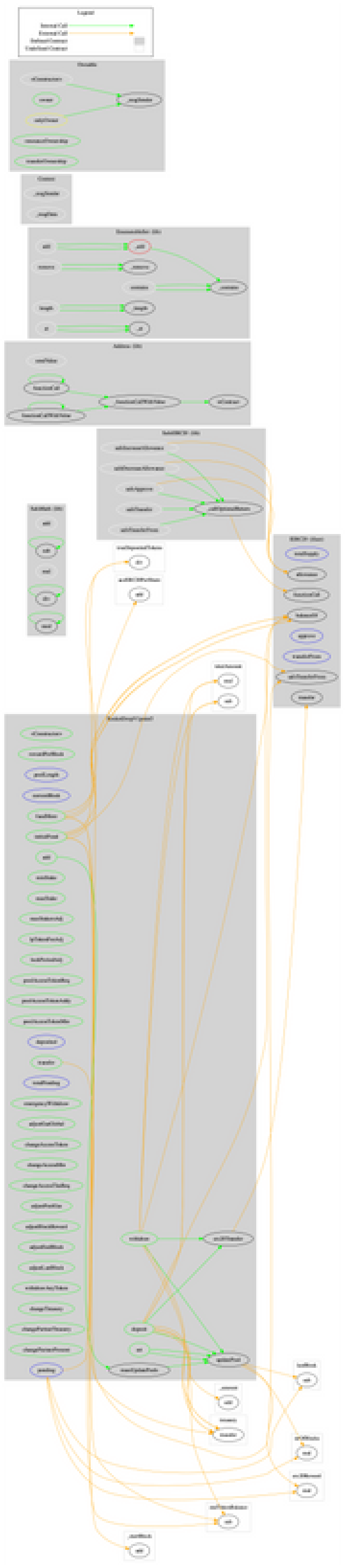
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	_functionCallWithValue	Private	✓	
SafeERC20	Library			
	safeTransfer	Internal	✓	

	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	_callOptionalReturn	Private	✓	
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		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
	<Constructor>	Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
RocketDropV1point5	Implementation	Ownable		

	<Constructor>	Public	✓	-
	rewardPerBlock	Public		-
	poolLength	External		-
	currentBlock	External		-
	initialFund	Public	✓	-
	fundMore	Public	✓	-
	add	Public	✓	onlyOwner
	set	Public	✓	onlyOwner
	minStake	Public	✓	onlyOwner
	maxStake	Public	✓	onlyOwner
	maxStakersAdj	Public	✓	onlyOwner
	lpTokenFeeAdj	Public	✓	onlyOwner
	lockPeriodAdj	Public	✓	onlyOwner
	poolAccessTokenReq	Public	✓	onlyOwner
	poolAccessTokenAddy	Public	✓	onlyOwner
	poolAccessTokenMin	Public	✓	onlyOwner
	deposited	External		-
	pending	External		-
	totalPending	External		-
	massUpdatePools	Public	✓	-
	updatePool	Public	✓	-
	deposit	Public	Payable	-
	withdraw	Public	Payable	-
	emergencyWithdraw	Public	✓	-
	erc20Transfer	Internal	✓	
	adjustGasGlobal	Public	✓	onlyOwner
	changeAccessToken	Public	✓	onlyOwner
	changeAccessMin	Public	✓	onlyOwner
	changeAccessTknReq	Public	✓	onlyOwner
	adjustPoolGas	Public	✓	onlyOwner
	adjustBlockReward	Public	✓	onlyOwner
	adjustEndBlock	Public	✓	onlyOwner
	adjustLastBlock	Public	✓	onlyOwner
	withdrawAnyToken	Public	✓	onlyOwner
	changeTreasury	Public	✓	onlyOwner

	changePartnerTreasury	Public	✓	onlyOwner
	changePartnerPercent	Public	✓	onlyOwner
	transfer	Public	✓	onlyOwner

Contract Flow



Summary

The contract implements a staking functionality. Users have the ability to deposit an amount and receive rewards proportional to the time that has elapsed. This audit focuses on the business logic implementation, the security concerns and some potential performance improvements.

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 provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io