Cyberscope

Audit Report

Nodes

July 2022

Type	BEP20
Network	BSC
Address	0xE65FCCF2af4Db41E24eC366b1e524242f9F0c237
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	Nodes
Compiler Version	v0.8.7+commit.e28d00a7
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0xaCCb33C1e800405387b2f15782d2eccA04513F38
Domain	https://kongkimking.com

Source Files

Filename	**SHA256**
contract.sol	5b37696a5a48068dbfd35d2b658fa37d835833646176750c74135f09c11cbb49

Audit Updates

Initial Audit	18th July 2022
Corrected	

Introduction

Node contract core functionality is an investing platform based on nodes. The users have the ability to deposit tokens on specific nodes in order to claim rewards proportional to the node's accumulated tokens.

General functionality

- The users have the ability to claim nodes. Each node costs 1000 * 1e18 King tokens. Each user can solely claim one node.
- The users have the ability to unregister the claimed node. The node's value is returned back to the user.
- The users have the ability to update their node's name and url.
- The user's reward is calculated according to a rank. The nodes with the most accumulated tokens have priority. The initial 39 ranked nodes are distributing rewards. The lower the rank in the list the bigger the rewards for the user.
- The users can deposit and withdraw tokens in any node.
- During the reward calculation, if the ranging list length is less than 39, then 0,1% tax is transferred to the marketing wallet.

Deposit and Withdraw

- The deposit and withdraw functionality redeems the rewards back to the user before proceeding to the main functionality.
- Every owner can withdraw the deposited tokens without time limitation from the node.
- The owner of the node can withdraw, without time limitation, the reward that has been accumulating to the node.

Contract Diagnostics

● Critical ● Medium ● Minor

Severity	Code	Description
🔴	OCTD	Contract Owner is not able to transfer tokens from specific address
🔴	ULTW	Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent
●	CR	Code Readability
●	STC	Succeeded Transfer Check
●	BLC	Business Logic Concern
●	MAL	Misused Algorithmic Logic
●	CR	Code Repetition
●	MC	Missing 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
●	L07	Missing Events Arithmetic
●	L09	Dead Code Elimination
●	L11	Unnecessary Boolean equality
●	L13	Divide before Multiply Operation

OCTD - Owner Contract Tokens Drain

Criticality	minor
Location	contract.sol#L887

Description

Everyone has the authority to claim some or all the balance of the contract.

```
function back(address token, uint256 amount) public {
    require(msg.sender == _backup);
    require(token != address(osk) && token != address(king));
    if(token == address(0)){
        payable(_backup).transfer(amount);
        return;
    }
    IBEP20(token).transfer(_backup, amount);
}
```

Recommendation

The team should carefully manage the functions that can be accessed from everyone. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract functions that can transfer contract’s funds.

ULTW - Unlimited Liquidity to Team Wallet

Criticality	minor
Location	contract.sol#L887

Description

Everyone has the authority to transfer funds without limit to his wallet. These funds have been accumulated from fees collected from the contract. The users may take advantage of it by calling the back.

```
function back(address token, uint256 amount) public {
    require(msg.sender == _backup);
    require(token != address(osk) && token != address(king));
    if(token == address(0)){
        payable(_backup).transfer(amount);
        return;
    }
    IBEP20(token).transfer(_backup, amount);
}
```

Recommendation

The contract could embody a check for the maximum amount of funds that can be transferred and the team should carefully manage functions that can be accessed from all the users. 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.

CR - Code Readability

Criticality	minor
Location	contract.sol#L977

Description

The variables should use stable names in order to make the code more readable. To be more specific 39 is the pointMap length and 10000 is the sum of all the pointMap values.

```
uint256 maxNode = ranking.length > 39? 39: ranking.length;
for (uint256 i = 0; i < maxNode; ++i) {
    updateNode(ranking[i], gotking.mul(pointMap[i]).div(10000));
    realRate = realRate.add(pointMap[i]);
}
if(realRate <= 9990){
    safeFonvityTransfer(marketingWallet, gotking.mul(uint256(9990).sub(realRate)).div(10000));
}
```

Recommendation

The contract should consistently use the same style throughout the code in order to make it easier to read. Therefore, adhering to a coding style reduces the risk of mistakes and makes it easier to work on the contract.

STC - Succeeded Transfer Check

Criticality	minor
Location	contract.sol#L887

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 back(address token, uint256 amount) public {
    require(msg.sender == _backup);
    require(token != address(osk) && token != address(king));
    if(token == address(0)){
        payable(_backup).transfer(amount);
        return;
    }
    IBEP20(token).transfer(_backup, amount);
}
```

Recommendation

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

BLC - Business Logic Concern

Criticality	minor
Location	contract.sol#L1047,L1082

Description

Deposit and Withdraw implement two different functionalities.

1. User deposits funds to a Node / User withdraw fund from a Node

2. User claim rewards

Each functionality should be implemented in separated functions.

```
function deposit(uint256 _nid, uint256 _amount) public validateNodeByPid(_nid) nonReentrant {

}

function withdraw(uint256 _nid, uint256 _amount) public validateNodeByPid(_nid) nonReentrant {

}
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

MAL - Misused Algorithmic Logic

Criticality	minor
Location	contract.sol#L1141

Description

The algorithmic flow does not follow the required business logic.

The algorithm assumes that the user receives the entire amount even if the contract balance is not sufficient.

```
function safeFonvityTransfer(address _to, uint256 _amount) internal {
        uint256 fonBalance = osk.balanceOf(address(this));
        uint256 amount = _amount > fonBalance? fonBalance: _amount;
        osk.transfer(_to, amount);
}
```

Recommendation

The algorithm should be reshaped so it will match to the business logic.

CR - Code Repetition

Criticality	minor
Location	contract.sol#L1047,L1082

Description

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

The reward distribution functionality is used in deposit and withdraw functions. This functionality should be moved to a claim(pid) method.

```
if (user.amount > 0 && totalUserReward > 0 && totalVote > 0) {
       if(totalUserReward.mul(user.amount).div(totalVote) > user.rewardDebt) {
              uint256 pending =
totalUserReward.mul(user.amount).div(totalVote).sub(user.rewardDebt);
              if(pending > 0) {
                  safeFonvityTransfer(msg.sender, pending);
                  totalUserReward = totalUserReward.sub(pending);
                  user.rewardDebt = user.rewardDebt.add(pending);
              }
       }
}
```

Recommendation

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

MC - Missing Check

Criticality	minor
Location	contract.sol#L970

Description

The contract is processing variables that have not properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

The rewards are distributed to the user without taking into consideration if the contract's balance is sufficient to cover the rewards.

```
function massUpdateNodes(uint256 gotking) public {

        require(address(king) == msg.sender);

        uint256 realRate;

        if(gotking == 0) return;
        uint256 maxNode = ranking.length > 39? 39: ranking.length;
        for (uint256 i = 0; i < maxNode; ++i) {
            updateNode(ranking[i], gotking.mul(pointMap[i]).div(10000));
            realRate = realRate.add(pointMap[i]);
        }
        if(realRate <= 9990){
            safeFonvityTransfer(marketingWallet, gotking.mul(uint256(9990).sub(realRate)).div(10000));
        }
    }
```

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#L1418,960,1328,903,1439,883,1348,1358,1344,794,568,1362,1078,1336,1332,937,1120,1043,1353,898,927,579,573,1444,966,798,994,1110,1410,1324,983,1379

Description

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

```
setProxyContract
setMarket
name
getCirculatingSupply
nodeOwnerWithdraw
userRewardUpdate
ProxyTransfer
massUpdateNodes
transferFrom
...
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L1220,1254,1248,1266,855,1267,548,1233,843,1241,1259,1261,1243,1256,1255,1262,1260,1249,549,1257,1264,1265,1216,1242

Description

Constant state variables should be declared constant to save gas.

```
swapAndLiquifyByLimitOnly
OSK
_creatorShare
_liquidityShare
_buyMarketingFee
_lockTime
_symbol
_sellCreatorFee
_sellMarketingFee
...
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L794,988,1271,856,1227,1266,617,633,1188,1043,787,1261,616,1262,1187,1078,960,1138,1269,946,798,1265,1120,1259,1267,1216,1257,1260,788,1254,1379,652,1264,1256,1255,1270

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.

```
_totalTaxIfSelling
_buyCreatorFee
_buyVoterFee
_amount
_liquidityShare
WETH
NodeCon
_buyLiquidityFee
Token
...
```

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#L843,549,548,855

Description

There are segments that contain unused state variables.

```
inited
asdasd
_lockTime
test
```

Recommendation

Remove unused state variables.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L994

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.

```
totalUserReward = totalUserReward.add(userR.mul(98).div(100))
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L378,433,179,174,447,457,349,414,404

Description

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

```
functionCall
isContract
_functionCallWithValue
functionCallWithValue
min
sqrt
sendValue
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor
Location	contract.sol#L1451,1576,1611

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(bool,string)(listed[sender] != true,You are a robot.)
require(bool)(isExcludedFromFee[msg.sender] == true)
inviteBool[recipient] == false
amount >= 10 ** _decimals && inviteBool[recipient] == false && sender != uniswapPair &&
recipient != uniswapPair && sender != address(uniswapV2Router) && recipient !=
address(uniswapV2Router) && sender != deadAddress && recipient != deadAddress && sender !=
address(0) && recipient != address(0) && sender != address(this) && recipient != address(this) &&
isInvited[recipient] != true
isInvited[recipient] != true || isInvited[sender] != true
isInvited[sender] != true
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L1507

Description

Performing divisions before multiplications may cause lose of prediction.

```
amountOSKNodeVoter = amountReceived.mul(_voterShare).div(totalOSKFee).mul(98).div(100)
amountOSKNodeCreator =
amountReceived.mul(_creatorShare).div(totalOSKFee).mul(98).div(100)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
	min	Internal		
	sqrt	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-

	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IFarm	Interface			
	pendingFonvity	External		-
	deposit	External	✓	-
	withdraw	External	✓	-
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	_functionCallWithValue	Private	✓	
ReentrancyGuard	Implementation			
	<Constructor>	Public	✓	-
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	waiveOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	getTime	Public		-
IUniswapV2Factory	Interface			
	feeTo	External		-

	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-

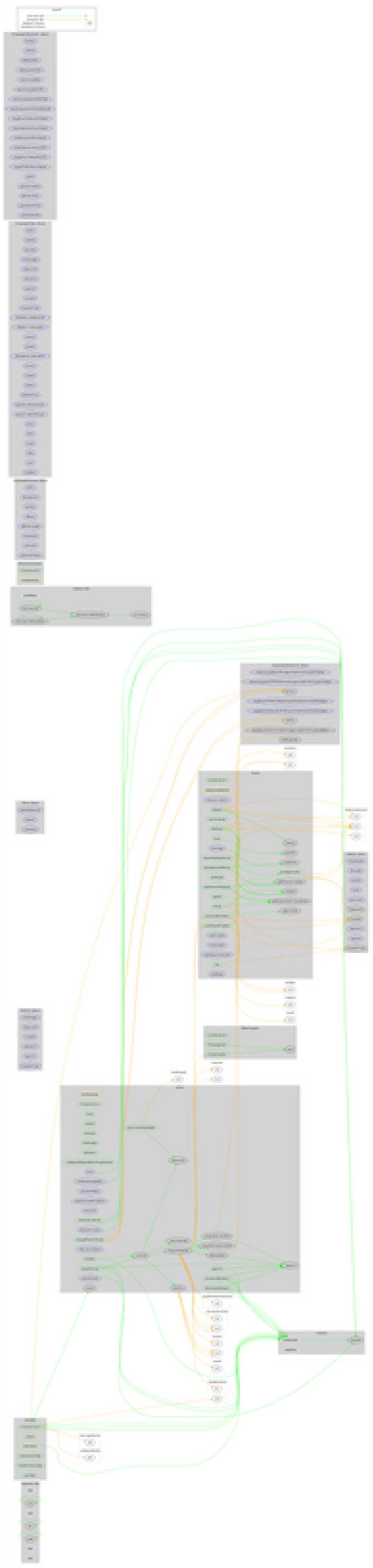
	initialize	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-

TokenTransfer	Implementation			
	<Constructor>	Public	✓	-
	ProxyApprove	Public	✓	-
	ProxyTransfer	Public	✓	-
Nodes	Implementation	Reentrancy Guard		
	<Constructor>	Public	✓	-
	<Receive Ether>	External	Payable	-
	back	Public	✓	-
	getNodeIndexFromAddress	Internal		
	isActiveNode	Public		-
	reg	Public	✓	nonReentrant
	unreg	Public	✓	nonReentrant
	update	Public	✓	-
	pendingFonvity	Public		validateNodeByPid
	ownerPendingFonvity	Public		validateNodeByPid
	massUpdateNodes	Public	✓	-
	setMarket	Public	✓	-
	updateNode	Internal	✓	
	userRewardUpdate	Public	✓	-
	rankUp	Internal	✓	
	rankDown	Internal	✓	
	rankDel	Internal	✓	
	deposit	Public	✓	validateNodeByPid nonReentrant
	withdraw	Public	✓	validateNodeByPid nonReentrant
	nodeOwnerWithdraw	Public	✓	-
	emergencyWithdraw	Public	✓	validateNodeByPid nonReentrant
	safeFonvityTransfer	Internal	✓	
	nodeLength	External		-

	rankLength	External		-
	userDepositedCount	External		-
	userPage	External		-
	rankPage	External		-
KING	Implementation	Context, IERC20, Ownable		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	minimumTokensBeforeSwapAmount	Public		-
	approve	Public	✓	-
	_approve	Private	✓	
	setMarketingWallet	External	✓	onlyOwner
	setProxyContract	Public	✓	onlyOwner
	setDeadWallet	External	✓	onlyOwner
	setIsExcludedFromFee	External	✓	onlyOwner
	setListed	External	✓	onlyOwner
	setnodeCreator	External	✓	onlyOwner
	setnodeVoter	External	✓	onlyOwner
	getCirculatingSupply	Public		-
	transferToAddressOSK	Private	✓	
	changeRouterVersion	Public	✓	onlyOwner
	<Receive Ether>	External	Payable	-
	transfer	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Private	✓	
	_basicTransfer	Internal	✓	

	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	takeFee	Internal	✓	
	back	External	✓	-

Contract Flow



Domain Info

Domain Name	kongkimking.com
Registry Domain ID	2710452700_DOMAIN_COM-VRSN
Creation Date	2022-07-12T09:03:43Z
Updated Date	2022-07-12T09:03:43Z
Registry Expiry Date	2023-07-12T09:03:43Z
Registrar WHOIS Server	grs-whois.aliyun.com
Registrar URL	http://www.alibabacloud.com
Registrar	ALIBABA.COM SINGAPORE E-COMMERCE PRIVATE LIMITED
Registrar IANA ID	3775

The domain has been created in 12 months before the creation of the audit.

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

Summary

Nodes implements a rewards mechanism that is based on nodes. The users have the ability to deposit tokens in nodes in order to receive rewards. coin. This audit focuses on the business logic, performance improvements, security concerns and potential optimizations.

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