Cyberscope

Audit Report

Harmony Nodes Manager

April 2022

Github	https://github.com/harmonynodes/harmonynodes
Commit	71007f66ea6f560be6f9533aaeb0bbb4b0b84bfa
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Contract Review 3
Audit Updates 3
Source Files 4
Harmony Nodes Workflow 6
Contract Diagnostics 7
MNO - Mint Node Overcharge 8
Description 8
Recommendation 8
STC - Succeeded Transfer Check 9
Description 9
Recommendation 9
MC - Missing Check 10
Description 10
Recommendation 10
L02 - State Variables could be Declared Constant 11
Description 11
Recommendation 11
L04 - Conformance to Solidity Naming Conventions 12
Description 12
Recommendation 12
L05 - Unused State Variable 13
Description 13
Recommendation 13
L09 - Dead Code Elimination 14
Description 14

Recommendation 14
L11 - Unnecessary Boolean equality 15
Description 15
Recommendation 15
L13 - Divide before Multiply Operation 16
Description 16
Recommendation 16
Contract Functions 17
Contract Flow 24
Summary 25
Disclaimer 26
About Cyberscope 27

Contract Review

Github	https://github.com/harmonynodes/harmonynodes
Commit	71007f66ea6f560be6f9533aaeb0bbb4b0b84bfa
Contract Name	HarmonyNodeManage

Audit Updates

Initial Audit	26th April 2022
Corrected	

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	75e3c97011e75627ffb36f4a2799a4e887e1a3e27ed427490e82d7b6f51cc5c9
@openzeppelin/contracts/security/ReentrancyGuard.sol	aa73590d5265031c5bb64b5c0e7f84c44cf5f8539e6d8606b763adac784e8b2e
@openzeppelin/contracts/token/ERC20/IERC20.sol	c2b06bb4572bb4f84bfc5477dadc0fcc497cb66c3a1bd53480e68bedc2e154a6
@openzeppelin/contracts/token/ERC721/IERC721.sol	a88e8e63c7a737436f7ec62542620609ab07bb9a772e77146ed4dc98539e03d3
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/introspection/IERC165.sol	701e025d13ec6be09ae892eb029cd83b3064325801d73654847a5fb11c58b1e5
contracts/Harmony_NodeManage.sol	3cdfdb2d45723af1bcc31f0b3ac81b5e6e5007a93ea670c6cca13dce472ee7c6
contracts/HarmonyLibrary.sol	fe394e4673030daa1fa2ee37d44f41c9a223fe2223f80401f3d59831c758d61e
contracts/interfaces/ICommonStruct.sol	747b4cac1a9d9313bbfee2b29add000dc8b36296037270cc5c3ffc874adefed7
contracts/interfaces/IHarmony_Node	c3d34c58d90424cce78b1232f055ba396fc5b156ae4fa58e29e162be9589f222

Manage.sol	
contracts/interfaces/IHone_Node.sol	7d852ab677485fc387d2af128c38f3e4ecb41c47165c3e9627ae0db462813a89
contracts/interfaces/IHONE.sol	7c0640d6b05f69c78ece2a10afad3ea67703e4b5994c7f61e5a3e3508da45fdf
contracts/interfaces/IUniswapV2Factory.sol	4158fa477eb2e55aec14343d2e917ab085c71ed068ff2a56a51ee9fa6311879e
contracts/interfaces/IUniswapV2Pair.sol	6a7c6cf1bee1404140c33be5415d887c47a1433869a2f3763c46396e287a52eb
contracts/interfaces/IUniswapV2Router02.sol	8630a0478e76aca1807ded7d149e51b75c7f142e4ad1e3a32df1ea823dc801c5
hardhat/console.sol	27d7e349617dc857b040f2186bf577fe6169ede8bfc98be714ab4289b5793548

Harmony Nodes Workflow

Harmony nodes tokens implement a reward mechanism. Users have the ability to create nodes. The user pays in HONE tokens in order to get a node. The nodes have a variation of cost and maintenance fee. The nodes cost & maintenance fee is the following:

No	Type	Node Cost	Maintenance Fee
0	Nano	10	10
2	Pico	20	20
3	Mega	50	30
4	Giga	100	45

Each address can update the owned node without limit. For instance, an address could have one Nano and two Mega nodes.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	STC	Succeeded Transfer Check
	MC	Missing Check
	L02	State Variables could be Declared Constant
	L04	Conformance to Solidity Naming Conventions
	L05	Unused State Variable
	L09	Dead Code Elimination
	L11	Unnecessary Boolean equality
	L13	Divide before Multiply Operation

MNO - Mint Node Overcharge

Criticality	medium
Location	contract.sol#L94

Description

The `createNode()` provides a node according to a charge. If the issuer provides more tokens than the specification, they are all transferred to the contract. The `createNode()` does not provide a way to track these amounts. Thus, the user will essentially pay more than the required without any reward.

```
function createNode(
   uint256 honeAmount_,
   uint8 createNodeType_
) external nonReentrant override {
   address sender = _msgSender();
   require (createNodeType_ < 4, 'wrong node type');
   require (sender != address(0), 'zero address');
   require ((uint256(nodePrice[createNodeType_]) * 1e18) <= honeAmount_, 'no
correct cost');

   uint256 curTime = block.timestamp;
   honeToken.transferFrom(sender, address(this), honeAmount_);
```

Recommendation

The mintHarmonyNode() should transfer only the amount of tokens that is required for the specific node type.

STC - Succeeded Transfer Check

Criticality	minor
Location	contract.sol#L75,104,233,304

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.

```
honeNFT.transferFrom(sender, address(this), tokenID_);
honeToken.transfer(to_, amount_);
...
```

Recommendation

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

MC - Missing Check

Criticality	minor
Location	contract.sol#L171

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 `nodeType_` should be checked to be less than the node types.

```
function upgradeNode(
   uint8 nodeType_,
   uint256[] memory useNodes_
) external nonReentrant override {
   address sender = _msgSender();
   require (sender != address(0), 'zero address');
   require (nodeType_ > 0 && useNodes_.length > 0, 'bad condition');
   uint256 curTime = block.timestamp;

   uint256 deadline = 0;
   if (usdcToken.allowance(sender, address(this)) >=
uint256(maintenanceFee[nodeType_]) * 1e6) {
      deadline = 30;
   }
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contracts/Harmony_NodeManage.sol#L25

Description

Constant state variables should be declared constant to save gas.

```
deadAddress
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/Harmony_NodeManage.sol#L31,32,33,34 contracts/interfaces/IHarmony_NodeManage.sol#L6

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.

```
IHarmony_NodeMange
NODE_TYPE_GIGA
NODE_TYPE_MEGA
NODE_TYPE_PICO
NODE_TYPE_NANO
```

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	contracts/Harmony_NodeManage.sol#L25,32,33,34

Description

There are segments that contain unused state variables.

```
NODE_TYPE_GIGA
NODE_TYPE_MEGA
NODE_TYPE_PICO
deadAddress
```

Recommendation

Remove unused state variables.

L09 - Dead Code Elimination

Criticality	minor
Location	contracts/HarmonyLibrary.sol#L46,72,130,23

Description

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

```
_getRewardAmount
_getLendStatus
_getLendReward
_getGeneralReward
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor
Location	contracts/HarmonyLibrary.sol#L130 contracts/Harmony_NodeManage.sol#L68,94,213,223,252

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.

```
lendOffers[i].forLend == true
lendOffers[i_scope_0].forLend == true
require(bool,string)(lendOffers[offerIndex_].forLend == true,offer closed)
require(bool,string)(lendOffers[offerIndex_].forLend == true,already closed)
usdcToken.transferFrom(sender,address(this),fee) == true
rentStatus_.lendStatus == false || (rentStatus_.lendStatus == true &&
rentStatus_.rentDeadline <= curTime_)
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contracts/HarmonyLibrary.sol#L10,23

Description

Performing divisions before multiplications may cause lose of prediction.

```
reward = reward * 95 ** 30 / 100 ** 30
reward = uint256(nodePrice_) * 1e18 * uint256(nodePercent_) / 1e4
liquidity = honeAmount_ / 10
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ReentrancyGuard	Implementation			
	<Constructor>	Public	✓	-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC721	Interface	IERC165		
	balanceOf	External		-
	ownerOf	External		-
	safeTransferFrom	External	✓	-
	transferFrom	External	✓	-
	approve	External	✓	-
	getApproved	External		-
	setApprovalForAll	External	✓	-
	isApprovedForAll	External		-
	safeTransferFrom	External	✓	-

Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC165	Interface			
	supportsInterface	External		-
HarmonyNode Manage	Implementation	Ownable, ReentrancyGuard, IHarmony_NodeMange		
	<Constructor>	Public	✓	-
	setRewardPool	External	✓	onlyOwner
	claimRewards	External	✓	nonReentrant
	getClaimableRewards	External		-
	swapNode	External	✓	nonReentrant
	createNode	External	✓	nonReentrant
	payAllMaintenanceFee	External	✓	nonReentrant
	payMaintenanceFee	External	✓	nonReentrant
	upgradeNode	External	✓	nonReentrant
	listLendOffer	External	✓	nonReentrant
	closeLendOffer	External	✓	-
	acceptLendOffer	External	✓	-
	getLendOffers	External		-
	_getLendOffers	Internal		
	getNodeCount	External		-
	getNodes	External		-
	_transfer	Internal	✓	
HarmonyLibrary	Library			
	_calcAmount	Internal		
	_getRewardAmount	Internal		
	_getGeneralReward	Internal		
	_getLendReward	Internal		

	_getLendStatus	Internal		
ICommonStruct	Interface			
IHarmony_NodeMange	Interface	ICommonStruct		
	setRewardPool	External	✓	-
	claimRewards	External	✓	-
	createNode	External	✓	-
	swapNode	External	✓	-
	payMaintenanceFee	External	✓	-
	payAllMaintenanceFee	External	✓	-
	upgradeNode	External	✓	-
	listLendOffer	External	✓	-
	closeLendOffer	External	✓	-
	acceptLendOffer	External	✓	-
	getNodeCount	External		-
IHoneNode	Interface	ICommonStruct		
	claimRewards	External	✓	-
	getClaimableRewards	External		-
	createNode	External	✓	-
	upgradeNode	External	✓	-
	payMaintenanceFee	External	✓	-
	payAllMaintenanceFee	External	✓	-
	listLendOffer	External	✓	-
	closeLendOffer	External	✓	-
	acceptLendOffer	External	✓	-
	getNodeCount	External		-
	getNodes	External		-
IHONE	Interface	IERC20		
	setSaleFee	External	✓	-
	setTransferFee	External	✓	-

	setFeeCollectWallet	External	✓	-
	setNodeManagementContract	External	✓	-
	enableBlacklist	External	✓	-
	disableBlacklist	External	✓	-
	isBlacklisted	External		-
	mint	External	✓	-
	burn	External	✓	-
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		-
	mint	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	✓	-
console	Library			
	_sendLogPayload	Private		
	log	Internal		
	logInt	Internal		
	logUint	Internal		
	logString	Internal		
	logBool	Internal		
	logAddress	Internal		
	logBytes	Internal		
	logBytes1	Internal		
	logBytes2	Internal		
	logBytes3	Internal		
	logBytes4	Internal		
	logBytes5	Internal		
	logBytes6	Internal		
	logBytes7	Internal		
	logBytes8	Internal		
	logBytes9	Internal		
	logBytes10	Internal		
	logBytes11	Internal		
	logBytes12	Internal		
	logBytes13	Internal		
	logBytes14	Internal		
	logBytes15	Internal		

	logBytes16	Internal		
	logBytes17	Internal		
	logBytes18	Internal		
	logBytes19	Internal		
	logBytes20	Internal		
	logBytes21	Internal		
	logBytes22	Internal		
	logBytes23	Internal		
	logBytes24	Internal		
	logBytes25	Internal		
	logBytes26	Internal		
	logBytes27	Internal		
	logBytes28	Internal		
	logBytes29	Internal		
	logBytes30	Internal		
	logBytes31	Internal		
	logBytes32	Internal		
	log	Internal		

Contract Flow

Summary

Harmony Nodes Manager implements a vesting-related functionality. The users pay in HONE tokens and USDT in order to purchase nodes. The users can claim their rewards proportionally to their holding and the time period that has elapsed. The Harmony Nodes Manager heavily depends on the Harmony Nodes Node contract. Most of the functionality is delegated to the Node contract. This audit focus in the business logic flow, security concerns and potential 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