Cyberscope

Audit Report

Harmony Nodes Token

April 2022

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

Table of Contents

Contract Review

Github	https://github.com/harmonynodes/harmonynodes
Commit	71007f66ea6f560be6f9533aaeb0bbb4b0b84bfa
Contract Name	HONE
Symbol	HONE
Decimals	18
Total Supply	600,000

Audit Updates

Initial Audit	26th April 2022
Corrected	

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	75e3c97011e75627ffb36f4a2799a4e887e1a3e27ed427490e82d7b6f51cc5c9
@openzeppelin/contracts/token/ERC20/ERC20.sol	f7831910f2ed6d32acff6431e5998baf50e4a00121303b27e974aab0ec637d79
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	c2b06bb4572bb4f84bfc5477dadc0fcc497cb66c3a1bd53480e68bedc2e154a6
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
contracts/HONE.sol	4e51ce1ac602e0cd6a264d5c488543a845800191584e413dfc27c1cbbf02b239
contracts/interfaces/ICommonStruct.sol	747b4cac1a9d9313bbfee2b29add000dc8b36296037270cc5c3ffc874adefed7
contracts/interfaces/IHarmony_NodeManage.sol	c3d34c58d90424cce78b1232f055ba396fc5b156ae4fa58e29e162be9589f222
contracts/interfaces/IUniswapV2Factory.sol	4158fa477eb2e55aec14343d2e917ab085c71ed068ff2a56a51ee9fa6311879e
contracts/interfaces/IUniswapV2Pair.s	6a7c6cf1bee1404140c33be5415d887c47a1433869a2f3763c46396e287a52eb

ol	
contracts/interfaces/IUniswapV2Router02.sol	8630a0478e76aca1807ded7d149e51b75c7f142e4ad1e3a32df1ea823dc801c5
hardhat/console.sol	27d7e349617dc857b040f2186bf577fe6169ede8bfc98be714ab4289b5793548

Contract Analysis

Critical Medium Minor Pass

Severity	Code	Description
●	ST	Contract Owner is not able to stop or pause transactions
●	OCTD	Contract Owner is not able to transfer tokens from specific address
●	OTUT	Owner Transfer User's Tokens
●	ELFM	Contract Owner is not able to increase fees more than a reasonable percent (25%)
●	ULTW	Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent
●	MT	Contract Owner is not able to mint new tokens
●	BT	Contract Owner is not able to burn tokens from specific wallet
●	BC	Contract Owner is not able to blacklist wallets from selling

ST - Stop Transactions

Criticality	critical
Location	contract.sol#L79

Description

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting the `checkHasNode` to one and preventing all the non-node holders from selling or by setting the `saleFee` to a high value.

```
if (to == address(uniswapV2Pair)) {//sell
  if (checkHasNode == 1)
  {
    uint countNode = HONE_NODE.getNodeCount(from);
    require(countNode >= 1, "you haven't any Node.");
  }
}
if(from == address(uniswapV2Pair)) {   // buy fee : 0

} else if(to == address(uniswapV2Pair)) {  // sell fee logic
  tax = baseUnit * saleFee;
} else { // transfer fee
  tax = baseUnit * transferFee;
}
```

Recommendation

The contract could embody a check for not allowing the non-node holders to buy as well. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply.

Read more about the fees manipulation in the corresponding section.

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.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L52

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setTaxFeePercent` function with a high percentage value.

```
function setSaleFee(uint _saleFee) public onlyOwner{
  saleFee = _saleFee;
}
```

Recommendation

The contract could embody a check for the maximum acceptable value.

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.

MT - Mint Tokens

Criticality	critical
Location	contract.sol#L142

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `setTaxFeePercent` function. As a result the contract tokens will be highly inflated

```
function mint(uint256 _amount) public {
  require(msg.sender == honeNode || msg.sender == _owner, 'Can only be used by honeNode or owner.');
  _mint(msg.sender, _amount);
}
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

BC - Blacklisted Contracts

Criticality	medium
Location	contract.sol#L109

Description

The contract owner has the authority to stop contracts from transactions. The owner may take advantage of it by calling the `enableBlacklist` function.

```
require(!isBlacklisted(msg.sender), "HONE TOKEN: sender blacklisted");
require(!isBlacklisted(recipient), "HONE TOKEN: recipient blacklisted");
```

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.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	L01	Public Function could be Declared External
	L02	State Variables could be Declared Constant
	L04	Conformance to Solidity Naming Conventions
	L05	Unused State Variable
	L13	Divide before Multiply Operation
	L14	Uninitialized Variables in Local Scope

L01 - Public Function could be Declared External

Criticality	minor
Location	contracts/HONE.sol#L52,56,60,64,118,126,139,142,147

Description

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

```
burn
mint
setCheckHasNode
disableBlacklist
enableBlacklist
setNodeManagementContract
setFeeCollectWallet
setTransferFee
setSaleFee
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contracts/HONE.sol#L14,13,22

Description

Constant state variables should be declared constant to save gas.

```
usdcAddress
initalSupply
denominator
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/HONE.sol#L52,56,60,64,139,142,147,24

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.

```
HONE_NODE
_amount
_checkHasNode
_honeNode
_feeCollectWallet
_transferFee
_saleFee
```

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/HONE.sol#L25,26

Description

There are segments that contain unused state variables.

```
uniswapV2Factory
uniswapV2Router02
```

Recommendation

Remove unused state variables.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contracts/HONE.sol#L71

Description

Performing divisions before multiplications may cause lose of prediction.

```
baseUnit = amount / denominator
```

Recommendation

The multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	minor
Location	contracts/HONE.sol#L76

Description

The are variables that are defined in the local scope and are not initialized.

```
tax
```

Recommendation

All the local scoped variables should be initialized.

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	✓	
ERC20	Implementation	Context, IERC20, IERC20Metadata		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	

	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
HONE	Implementation	ERC20, Ownable		
	<Constructor>	Public	✓	ERC20
	setSaleFee	Public	✓	onlyOwner
	setTransferFee	Public	✓	onlyOwner
	setFeeCollectWallet	Public	✓	onlyOwner
	setNodeManagementContract	Public	✓	onlyOwner
	handleTax	Private	✓	
	_transfer	Internal	✓	
	enableBlacklist	Public	✓	onlyOwner
	disableBlacklist	Public	✓	onlyOwner
	isBlacklisted	Public		-
	setCheckHasNode	Public	✓	onlyOwner
	mint	Public	✓	-
	burn	Public	✓	-

	<Receive Ether>	External	Payable	-
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		-
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

There are some functions that can be abused by the owner, like manipulating fees, stopping transactions, blacklisting wallets and minting. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. We state that the owner privileges are necessary and required for proper protocol operations. Thus, we emphasise the contract owner to be extra careful with the credentials. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

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