Cyberscope

Audit Report

Pyramid Nodes

May 2022

SHA256 408a8ea82832e2bbd9a1082900860fa6a1089280e6ad6defd0478546bce282e9

Audited by © cyberscope

Table of Contents

Contract Review

Token Name	MIDToken
Node Name	NODERewardManagement

Source Files

Filename	**SHA256**
contract.sol	408a8ea82832e2bbd9a1082900860fa6a1089280e6ad6defd0478546bce282e9

Audit Updates

Initial Audit	9th June 2022
Corrected	

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	medium
Location	contract.sol#L1563,1573

Description

The contract owner has the authority to stop transactions for all users excluding the owner. The owner may take advantage of it by setting the `tokenLock` to true and whitelisting himself.

```
require(
    !tokenLock || whiteList[from] || whiteList[to],
    "ERC20: lock now"
);
```

The contract owner has the authority to stop transactions for all users. The owner may take advantage of it by setting the `isLimited` to true and `transacLimit` to zero.

```
if (isLimited == true) {
    require(amount <= transacLimit, "Sorry you can't transfer more than one token at a time");
}
```

Recommendation

The contract could embody a check for not allowing setting the _maxTxAmount less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply.

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#L1813

Description

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

```
function mint(address account, uint256 amount) public onlyOwner {
    _mint(account, 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#L1569

Description

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

```
require(
    !_isBlacklisted[from] && !_isBlacklisted[to],
    "Blacklisted address"
);
```

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
	CO	Code Optimization
	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
	L06	Missing Events Access Control
	L07	Missing Events Arithmetic
	L09	Dead Code Elimination
	L11	Unnecessary Boolean equality
	L13	Divide before Multiply Operation
	L14	Uninitialized Variables in Local Scope

CO - Code Optimization

Criticality	minor
Location	contract.sol

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 contract could swap the entire amount for native currency once, and send the proportional amount to each wallet.

```
if (balanceOfTreasury != 0) {
    swapAndSendToFee(treasuryAddress, balanceOfTreasury);
}

if (balanceOfTeam != 0) {
    swapAndSendToFee(teamAddress, balanceOfTeam);
}

swapAndLiquify(balanceOfLiquidity, sender);
```

There are code segments that are calculated inside the loops but do not depend on the loop's variables. As a result, the same value is calculated every time.

```
for (uint256 i = 0; i < nodesCount; i++) {
    rewardCount += (nodes[i].rewardAvailable +
        ((block.timestamp - nodes[i].lastClaimedTime) *
            nodes[i].tokenValue *
            rewardRate) /
        (100 * (nodePrice / (10**18)) * DetaforDay));
}
```

Since the index is initially zero, then the `if (numberOfNodeOwners > 0)` statement is redundant.

```
uint256 index;
if (numberOfNodeOwners > 0) {
    while (index < numberOfNodeOwners) {
```

Recommendation

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

CR - Code Repetition

Criticality	minor
Location	contract.sol

Description

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

- `_getRewardAmountOf()` could reuse the `_getRewardOfNode()`
- `_getRewardAmountOfAsWeek()` could reuse the `_getRewardOfNodeAsWeek()`
- `_cashoutAllNodesReward()` could reuse the `_cashoutNodeReward()`
- `_updateRewardOfNode()` could reuse the `_updateRewardOfOneNode()`
- `_updateRewardOfAllNodes()` could reusethe `_updateRewardOfOneNode()`

Recommendation

The contract could reuse the same code in order to heavily decrease the code size.

MC - Missing Check (1/2)

Criticality	medium
Location	contract.sol#L1675

Description

The properties rewardPoolRate, teamRate and treasuryRate have no limit to the maximum value that can be set. As a result, if the variables are configured with a high value, then the following expression will underflow and revert the transaction.

```
uint256 balanceOfRewardPool = nodePrice.mul(rewardPoolRate).div(100);
uint256 balanceOfTeam = nodePrice.mul(teamRate).div(100);
uint256 balanceOfTreasury = nodePrice.mul(treasuryRate).div(100);
uint256 balanceOfLiquidity = nodePrice
    .sub(balanceOfTeam)
    .sub(balanceOfTreasury)
    .sub(balanceOfRewardPool);
```

Recommendation

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

MC - Missing Check (2/2)

Criticality	medium
Location	contract.sol#L1474

Description

The contract depends on the reward manager. If the contract owner changes the address, then the implementation may point to a vulnerable contract. As a result, the contract's functionality will be vulnerable.

```
function setNodeManagement(address payable _nodeManagement)
    external
    onlyOwner
{
    require(_nodeManagement != address(0), "Manager CANNOT BE ZERO");
    nodeRewardManager = NODERewardManagement(_nodeManagement);
}
```

Recommendation

The reward manager should be set once.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L667,675,692,699,724,737,754,777,806,833,990,1009,1018,1108,1302,1321,1338,1363,1502,1529,1580,1585,1589,1655,1698,1726,1733,1762,1794,1800,1805,1809,1813,1817,1832

Description

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

```
updateRewardOfAllNodes
start
mint
getTotalCreatedNodes
getNodePrice
changeNodePrice
getRewardTotalAmount
cashoutNodeReward
...
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L1419

Description

Constant state variables should be declared constant to save gas.

```
deadWallet
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L7,277,279,310,1074,1104,1108,1112,1117,1124,1128,1133,1138,1143,1179,1204,1233,1254,1277,1290,1302,1321,1338,1363,1050,1053,1464,1469,1474,1482,1486,1490,1494,1498,1631,1726,1762,1825,1421

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.

```
_isBlacklisted
ID
_to
_treasuryRate
_liquidityRate
_teamRate
_maxNodeNumberOf
_rewardPoolRate
_nodeManagement
...
```

Recommendation

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

L06 - Missing Events Access Control

Criticality	minor
Location	contract.sol#L1112

Description

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

```
token = _token
```

Recommendation

Emit an event for critical parameter changes.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L1124,1128,1133,1138,1482,1486,1490,1498,1523

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.

```
transacLimit = amount
treasuryRate = _treasuryRate
teamRate = _teamRate
maxNodeNumberOf = _maxNodeNumberOf
rewardPoolRate = _rewardPoolRate
nodePrice = _nodePrice
rewardRate = _rewardRate
DetaforDay = _DetaforDay
taxRate = _taxRate
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L911,1074

Description

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

```
uint2str
_burn
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor
Location	contract.sol#L1179,1204,1233,1254,1516,1558

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.

```
isLimited == true
isNodeOwner(account) == false
```

Recommendation

Remove the equality to the boolean constant.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L1074,1179,1204,1233,1254,1302,1321,1338,1363

Description

Performing divisions before multiplications may cause lose of prediction.

```
_node.rewardAvailable = _node.rewardAvailable + ((block.timestamp -
_node.lastClaimedTime) * _node.tokenValue * rewardRate) / (100 * (nodePrice /
(10 ** 18)) * DetaforDay)
_node.rewardAvailable = nodes[i].rewardAvailable + ((block.timestamp -
nodes[i].lastClaimedTime) * nodes[i].tokenValue * rewardRate) / (100 *
(nodePrice / (10 ** 18)) * DetaforDay)
rewardOfNode = ((node.rewardAvailable + ((block.timestamp -
node.lastClaimedTime) * node.tokenValue * rewardRate) / (100 * (nodePrice / (10
** 18)) * DetaforDay)) * taxRate) / 100
rewardOfNode = (node.rewardAvailable + ((block.timestamp - node.lastClaimedTime)
* node.tokenValue * rewardRate) / (100 * (nodePrice / (10 ** 18)) * DetaforDay))
temp = ((nodes[i].rewardAvailable + ((block.timestamp -
nodes[i].lastClaimedTime) * nodes[i].tokenValue * rewardRate) / (100 *
(nodePrice / (10 ** 18)) * DetaforDay)) * taxRate) / 100
rewardCount += (nodes[i].rewardAvailable + ((block.timestamp -
nodes[i].lastClaimedTime) * nodes[i].tokenValue * rewardRate) / (100 *
(nodePrice / (10 ** 18)) * DetaforDay))
temp = (48 + uint8(_i - (_i / 10) * 10))
```

Recommendation

The multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	minor
Location	contract.sol#L1343,1368

Description

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

```
index
```

Recommendation

All the local scoped variables should be initialized.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IJoeRouter01	Interface			
	factory	External		-
	WAVAX	External		-
	addLiquidity	External	✓	-
	addLiquidityAVAX	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityAVAX	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityAVAXWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactAVAXForTokens	External	Payable	-
	swapTokensForExactAVAX	External	✓	-
	swapExactTokensForAVAX	External	✓	-
	swapAVAXForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IJoeRouter02	Interface	IJoeRouter01		
	removeLiquidityAVAXSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityAVAXWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-

	swapExactAVAXForTokensSupporting FeeOnTransferTokens	External	Payable	-
	swapExactTokensForAVAXSupporting FeeOnTransferTokens	External	✓	-
IJoeFactory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	migrator	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
	setMigrator	External	✓	-
IJoePair	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		-

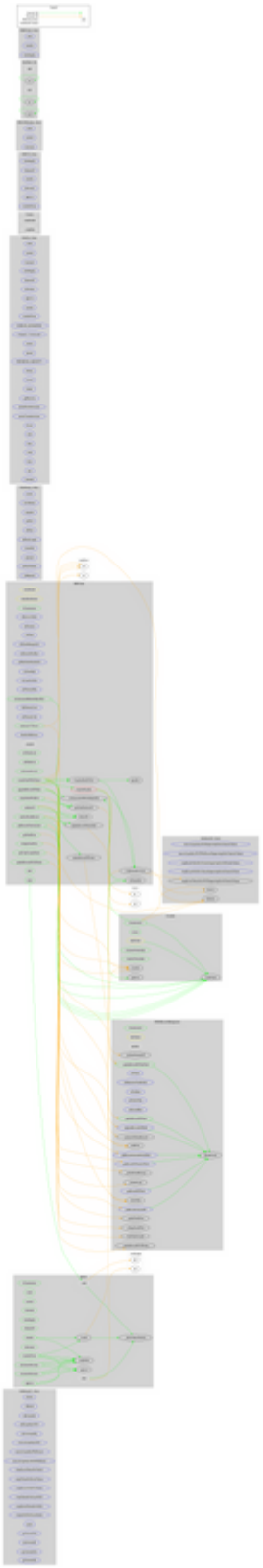
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		

	mod	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	✓	
	_beforeTokenTransfer	Internal	✓	
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IMIDToken	Interface			
	mint	External	✓	-
	transfer	External	✓	-
	totalSupply	External		-
NODERewardManagement	Implementation	Ownable		

	<Constructor>	Public	✓	-
	uint2str	Internal		
	isNodeOwner	Private		
	_isNodeOwner	External		-
	_getNodeNumberOf	Public		-
	setToken	External	✓	onlyOwner
	setMaximumTotalNodes	External	✓	onlyOwner
	setTaxRate	External	✓	onlyOwner
	setDetaforDay	External	✓	onlyOwner
	setRewardRate	External	✓	onlyOwner
	_changeNodePrice	External	✓	onlySentry
	_upgradeRewardOfNode	External	✓	onlySentry
	createNode	External	✓	onlySentry
	_getRewardAmountOf	External		-
	_getRewardAmountOfAsWeek	External		-
	_getRewardOfNode	External		-
	_getRewardOfNodeAsWeek	External		-
	_cashoutNodeReward	External	✓	onlySentry
	_cashoutAllNodesReward	External	✓	onlySentry
	_updateRewardOfNode	Public	✓	onlySentry
	_updateRewardOfOneNode	Public	✓	onlySentry
	_updateRewardOfAllNodes	Public	✓	onlySentry
	_updateNodePrice	Public	✓	onlySentry
MIDToken	Implementation	ERC20, Ownable		
	<Constructor>	Public	✓	ERC20
	<Receive Ether>	External	Payable	-
	setTreasury	External	✓	onlyOwner
	setTeam	External	✓	onlyOwner
	setNodeManagement	External	✓	onlyOwner
	setRewardPoolRate	External	✓	onlyOwner
	setMaxNodeNumberOf	External	✓	onlyOwner
	setTeamRate	External	✓	onlyOwner
	setLiquidityRate	External	✓	onlyOwner
	setTreasuryRate	External	✓	onlyOwner

	updatejoeV2Router	Public	✓	onlyOwner
	flipTransacLimit	External	✓	onlyOwner
	setTransacLimit	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	blacklistMalicious	External	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	_transfer	Internal	✓	
	setTokenLock	Public	✓	onlyOwner
	setWhiteList	Public	✓	onlyOwner
	setUnlockForever	Public	✓	onlyOwner
	swapAndLiquify	Private	✓	
	swapTokensForAvax	Private	✓	
	addLiquidity	Private	✓	
	createNodeWithTokens	Public	✓	onlyStarted whenDistributi on
	createNodeWithFree	Public	✓	onlyStarted onlyOwner whenDistributi on
	swapAndSendToFee	Private	✓	
	upgradeRewardOfNode	Public	✓	whenDistributi on
	cashoutAll	Public	✓	whenDistributi on
	cashoutNodeReward	Public	✓	whenDistributi on
	getNodeNumberOf	Public		-
	getRewardTotalAmount	Public		-
	changeNodePrice	Public	✓	onlyOwner
	getNodePrice	Public		-
	getTotalCreatedNodes	Public		-
	mint	Public	✓	onlyOwner
	start	Public	✓	onlyOwner
	updateRewardOfNodes	Private	✓	whenDistributi on
	updateRewardOfOneNode	Private	✓	whenDistributi on
	updateRewardOfAllNodes	Public	✓	onlyOwner

Contract Flow



Summary

There are some functions that can be abused by the owner like stopping transactions, minting tokens and blacklisting addresses. The contract owner has many privileges that heavily affect the contract's functionality. For instance, the owner role can create nodes without charge, set tax and multipliers in the rewards calculation process. If the mint functionality is abused by the contract owner, then the contract tokens will be highly inflated. 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 provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io