Cyberscope

Audit Report

Nut2Earn

July 2022

SHA256 e31bf5189b3613d93ac15b3ef916ec0a1b604868205de38303c06f3e9f5f252c

Audited by © cyberscope

Table of Contents

Contract Review

Source Files

Filename	SHA256
contract.sol	e31bf5189b3613d93ac15b3ef916ec0a1b604868205de38303c06f3e9f5f252c

Audit Updates

Initial Audit	31st July 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	critical
Location	contract.sol#L1082

Description

The contract owner has the authority to stop selling actions for all users by setting the `nutRebaseDenominator` variable to zero. This way the contract will not be able to rebase and it will be converted into a **HONEYPOT**.

```
function setNutRebase(uint256 _nutRebase, uint256 _nutRebaseDenominator)
   external
   onlyOwner
 {
   nutRebase = _nutRebase;
   nutRebaseDenominator = _nutRebaseDenominator;
 }
```

Recommendation

The contract could embody a check for not allowing setting the nutRebaseDenominator to take the zero 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.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L986,1003,1029

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 `setReferralSettings` function and setting the feeDenominator to the value 4 - while previously setting fees to the max allowed value.

```
function setReferralSettings(
    uint256 _referee,
    uint256 _referrer,
    uint256 _feeDenominator
  ) external onlyOwner {
    require(
      _referee.add(_referrer) <= buytreasuryFee, // checking that the referral
fee is not higher than the treasury fee
      "wrong"
    );
    referee = _referee;
    referrer = _referrer;
    totalReferralFee = referee.add(referrer);
    feeDenominator = _feeDenominator;
    require(totalReferralFee < feeDenominator / 4);
  }
```

Recommendation

The contract could embody a check to make sure the values are within limits. The referrer fee should also be a part of the buy/sell fees or be checked as a total fee for users.

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.

ULTW - Unlimited Liquidity to Team Wallet

Criticality	minor
Location	contract.sol#L1055

Description

The contract owner has the authority to transfer funds without limit to the team wallet. These funds have been accumulated from fees collected from the contract. The owner may take advantage of it by calling the `clearStuckBalance` method.

```
function clearStuckBalance(address _receiver) external onlyOwner {
  uint256 balance = address(this).balance;
  payable(_receiver).transfer(balance);
}
```

Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped. 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.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	ZD	Zero Division
	MTS	Manipulate Total Supply
	CO	Code Optimization
	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
	L13	Divide before Multiply Operation

ZD - Zero Division

Criticality	critical
Location	contract.sol#L888

Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert. The NutRebaseDenominator should always be higher than 0.

```
  int256 supplyDelta = int256(
        circulatingSupply.mul(nutRebase).div(nutRebaseDenominator)
      );
```

Recommendation

The contract should prevent those variables to be set to zero or should not allow to execute the corresponding statements.

CO - Code Optimization

Criticality	minor
Location	contract.sol#L1

Description

There are code segments that could be optimized. This condition will never be executed, hence the block inside the condition is redundant.

```
if (supplyDelta < 0) {
```

Recommendation

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

MTS - Manipulate Total Supply

Criticality	minor
Location	contract.sol#L903

Description

Owner is able to manipulate total supply. This change will have a direct impact on the token price and Market Cap.

```
if (supplyDelta < 0) {
      _totalSupply = _totalSupply.sub(uint256(-supplyDelta));
    } else {
      _totalSupply = _totalSupply.add(uint256(supplyDelta));
    }
    if (_totalSupply > MAX_SUPPLY) {
      _totalSupply = MAX_SUPPLY;
    }
```

Recommendation

The contract owner should carefully manage the adjustment of the circulating supply (increases or decreases), according to the token's price fluctuations.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L165,169,173,260,269,274,350,354,390,398

Description

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

```
getReferralTotalFee
getDownlines
getUplineAddressByIndex
getTotalUpline
transferOwnership
renounceOwnership
owner
decimals
symbol
...
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L339,340,338,433

Description

Constant state variables should be declared constant to save gas.

```
gonSwapThreshold
busdToken
ZEROWalletAddress
DEADWalletAddress
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L181,418,508,931,955,968,974,975,976,977,987,988,989,1004,1005,1006,1007,1008,1030,1031,1032,1033,1034,1055,1072,1077,1082,1090,1095,1100,326,339,340,344,404

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.

```
_markerPairs
Downlines
ZEROWalletAddress
DEADWalletAddress
_isFeeExempt
_maxTxn
_nextRebase
_value
_nutRebaseDenominator
...
```

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#L7

Description

There are segments that contain unused state variables.

```
MAX_INT256
```

Recommendation

Remove unused state variables.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L960,986,1003,1029,1077,1082,1095,1100

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.

```
maxSellTransactionAmount = _maxTxn
nextRebase = _nextRebase
nutRebase = _nutRebase
rebaseFrequency = _rebaseFrequency
sellfirepitfee = _sellfirepitfee
buyliquidityFee = _buyliquidityFee
referee = _referee
targetLiquidity = target
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L479,140,135,35

Description

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

```
abs
remove
has
isContract
```

Recommendation

Remove unused functions.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L742,801

Description

Performing divisions before multiplications may cause lose of prediction.

```
feeAmount = gonAmount.div(feeDenominator).mul(_realFee - referee)
_uplineBuyerReward = gonAmount.div(feeDenominator).mul(referrer)
_uplineReward = gonAmount.div(feeDenominator).mul(referrer)
contractTokenBalance = gonSwapThreshold.div(_gonsPerFragment)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

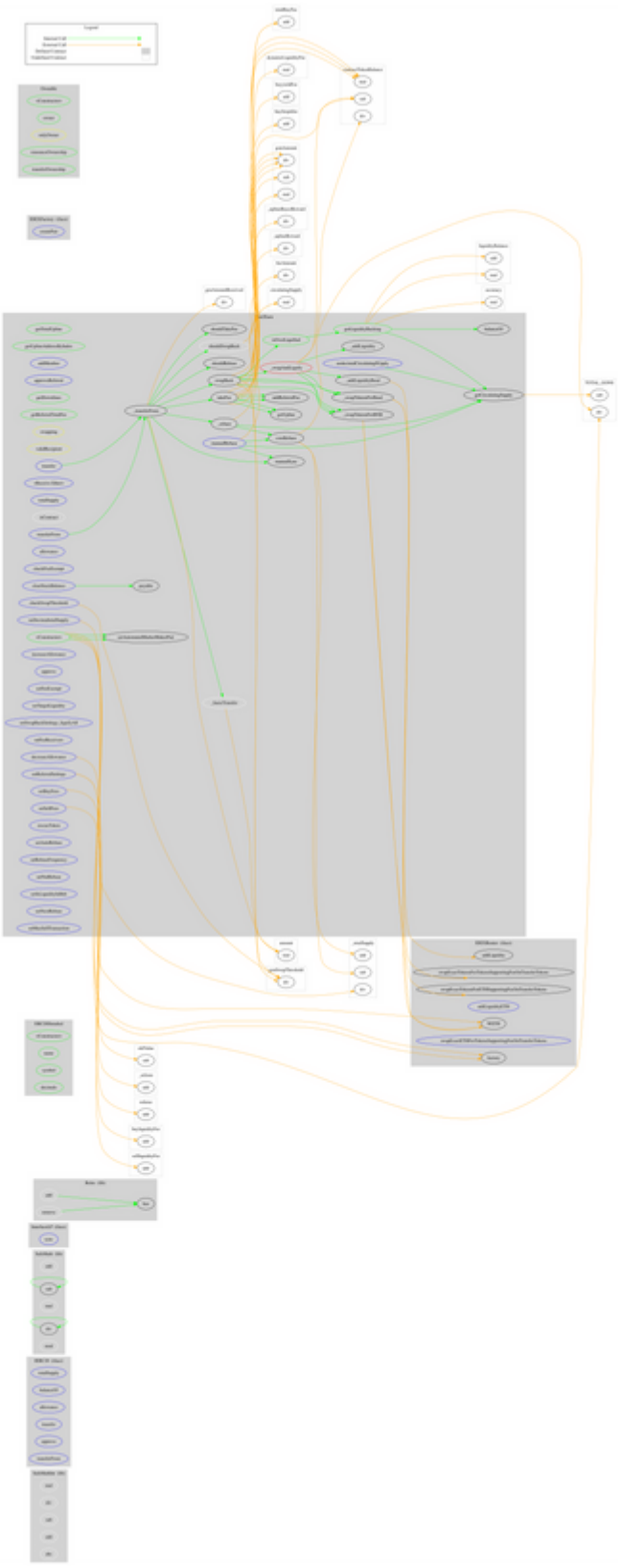
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
SafeMathInt	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		
	add	Internal		
	abs	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	transfer	External	✓	-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
InterfaceLP	Interface			
	sync	External	✓	-
Roles	Library			
	add	Internal	✓	

	remove	Internal	✓	
	has	Internal		
ERC20Detailed	Implementation	IERC20		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDEXFactory	Interface			
	createPair	External	✓	-
Ownable	Implementation			
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
Nut2Earn	Implementation	ERC20Detailed, Ownable		
	getTotalUpline	Public		-
	getUplineAddressByIndex	Public		-
	addMember	External	✓	onlyOwner
	approveReferral	External	✓	-

	getUpline	Public		-
	getDownlines	Public		-
	addReferralFee	Public	✓	-
	getReferralTotalFee	Public		-
	<Constructor>	Public	✓	ERC20Detailed
	<Receive Ether>	External	Payable	-
	totalSupply	External		-
	isContract	Internal		
	noDecimaltotalSupply	External		-
	nodecimalCirculatingSUpply	External		-
	allowance	External		-
	balanceOf	Public		-
	checkFeeExempt	External		-
	checkSwapThreshold	External		-
	shouldRebase	Internal		
	shouldTakeFee	Internal		
	shouldSwapBack	Internal		
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-
	manualSync	Public	✓	-
	transfer	External	✓	validRecipient
	_basicTransfer	Internal	✓	
	_transferFrom	Internal	✓	
	transferFrom	External	✓	validRecipient
	_swapAndLiquify	Private	✓	
	_addLiquidity	Private	✓	
	_addLiquidityBusd	Private	✓	
	_swapTokensForBNB	Private	✓	
	_swapTokensForBusd	Private	✓	
	swapBack	Internal	✓	swapping
	takeFee	Internal	✓	
	decreaseAllowance	External	✓	-
	increaseAllowance	External	✓	-
	approve	External	✓	-

	_rebase	Private	✓	
	coreRebase	Private	✓	
	manualRebase	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	setFeeExempt	External	✓	onlyOwner
	setTargetLiquidity	External	✓	onlyOwner
	setSwapBackSettings_liquifyAll	External	✓	onlyOwner
	setFeeReceivers	External	✓	onlyOwner
	setReferralSettings	External	✓	onlyOwner
	setBuyFees	External	✓	onlyOwner
	setSellFees	External	✓	onlyOwner
	clearStuckBalance	External	✓	onlyOwner
	rescueToken	External	✓	onlyOwner
	setAutoRebase	External	✓	onlyOwner
	setRebaseFrequency	External	✓	onlyOwner
	setNutRebase	External	✓	onlyOwner
	setIsLiquidityInBnb	External	✓	onlyOwner
	setNextRebase	External	✓	onlyOwner
	setMaxSellTransaction	External	✓	onlyOwner

Contract Flow



Summary

Nut2Earn is an interesting project that has a friendly and growing community. The Smart Contract users a totalsupply manipulation business model to provide value for the holders. The contact owner can abuse the admin functions to prevent users from selling their tokens and convert the contract into a Honeypot. He can also change the fees up to 100% and transfer tokens accumulated to the contract from fees into the team's wallet.

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.

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