Cyberscope

Audit Report

Eggpot

August 2022

Type	BEP20
Network	BSC
Address	0x398c482d0967876d1662e7ed60a0488ce13673c0
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	Eggpot
Compiler Version	v0.8.15+commit.e14f2714
Optimization	200 runs
Licence	None
Explorer	https://bscscan.com/token/0x398c482d0967876d1662e7ed60a0488ce13673c0
Symbol	EGGPOT
Decimals	18
Total Supply	1,000,000,000
Domain	https://eggpot.io

Audit Updates

Initial Audit	19th August 2022
Corrected	

Source Files

Filename	SHA256
access/Ownable.sol	65b66e7a5f3633539fbb59bb0dbebd9c29121c76490151e15f589c6bce9d59f6
Eggpot.sol	1581a92e5f7939760e75682ac62805f16a63757ea9d510d6de0e95738ec2b48e
interface/IERC20.sol	9a9ce403bcf5796cccfc9c0eb7514128fc1dca540b0617c0dc3ba9f0c2090e95
interface/IUniswapV2Factory.sol	5626a8cec78d7abc17fdc61fe0a9b6b3527b9b471aed6247a0093889778d1b39
interface/IUniswapV2Pair.sol	944ec57bb4c13e8c79218b9c67ee2ca44248186c8c79b77f8b57c432dcffec37
interface/IUniswapV2Router02.sol	5324618037c9db4cd7a9a9e6e5b924efe1185def3a9cd07a97ecf85d6882cc52
token/ERC20.sol	0c2528c77318e3b660a57fc992c56640fc18ddafd60c2346b3c20ed7cbd609ca
utils/Context.sol	cee91680eba65e7ab59b0ae26401f8006cb78c3b8a0c65679f86e250752a98af
utils/EnumerableSet.sol	67bb227a532561b3f4765db93d0535aa139615053b44e33ecc370d7b4b90b600

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#L398

Description

The contract owner has the authority to potentially stop transactions for all users excluding the owner. The owner may take advantage of it by setting the `blockForPenaltyEnd` to a relatively long period, for instance, one week. As a result, all the buyers will not be able to trade when the time period elapsed.

```
if (takeFee) {
  // bot/sniper penalty.
  if (
    earlyBuyPenaltyInEffect() &&
    automatedMarketMakerPairs[from] &&
    !automatedMarketMakerPairs[to]
  ) {
    if (!restrictedWallet[to]) {
      restrictedWallet[to] = true;
      botsCaught += 1;
      emit CaughtBot(to);
    }
```

Recommendation

The contract could embody a check for not allowing setting the blockForPenaltyEnd more 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.

ULTW - Unlimited Liquidity to Team Wallet

Criticality	minor
Location	contract.sol#L647,658

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 withdrawStuckETH and forceSwapBack methods.

```
function withdrawStuckETH() external onlyOwner {
    bool success;
    (success, ) = address(owner()).call{ value: address(this).balance }('');
}

function forceSwapBack() external onlyOwner {
    require(
      balanceOf(address(this)) >= swapTokensAtAmount,
      'Can only swap when token amount is at or higher than restriction'
    );
    swapping = true;
    swapBack();
    swapping = false;
    emit OwnerForcedSwapBack(block.timestamp);
}
```

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.

BC - Blacklisted Contracts

Criticality	critical
Location	contract.sol#L191

Description

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

```
function manageRestrictedWallets(address[] calldata wallets, bool restricted)
    external
    onlyOwner
{
    for (uint256 i = 0; i < wallets.length; i++) {
      restrictedWallet[wallets[i]] = restricted;
    }
}
```

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
●	STC	Succeeded Transfer Check
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L07	Missing Events Arithmetic
●	L13	Divide before Multiply Operation
●	L15	Local Scope Variable Shadowing

CO - Code Optimization

Criticality	minor
Location	contract.sol#L469

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.

Since the method dexRouter.getAmountsOut returns an array, there is no need to initialize an array.

```
uint256[] memory amounts = new uint256[](2);
amounts = dexRouter.getAmountsOut(minBuyAmount, path);
```

Recommendation

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

STC - Succeeded Transfer Check

Criticality	minor
Location	contract.sol#L627,642,488

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.

```
(success, ) = address(operationsAddress).call{ value: ethForOperations }('');

_sent = IERC20(_token).transfer(_to, _contractBalance);

(success, ) = address(winner).call{ value: winnings }('');
if (success) {
    emit JackpotTriggered(winnings, winner);
}
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contracts/Eggpot.sol#L62

Description

Constant state variables should be declared constant to save gas.

```
FEE_DENOMINATOR
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/Eggpot.sol#L293,282,281,634,62,158,65,295,283,294,64,652

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.

```
_operationsAddress
_isExcludedFromFees
_liquidityFee
_jackpotFee
_to
_isExcludedMaxTransactionAmount
_presaleAddress
FEE_DENOMINATOR
_token
...
```

Recommendation

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

L07 - Missing Events Arithmetic

Criticality	minor
Location	contracts/Eggpot.sol#L538,533,280,543,219,241,292

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.

```
sellOperationsFee = _operationsFee
numberOfBuysForJackpot = num
swapTokensAtAmount = newAmount
minBuyAmount = minBuy
buyOperationsFee = _operationsFee
timeBetweenBuysForJackpot = timeInMinutes * 60
percentForJackpot = percent
```

Recommendation

Emit an event for critical parameter changes.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contracts/Eggpot.sol#L318

Description

Performing divisions before multiplications may cause lose of prediction.

```
fees = (amount * (buyTotalFees)) / FEE_DENOMINATOR
fees = (amount * (sellTotalFees)) / FEE_DENOMINATOR
```

Recommendation

The multiplications should be prior to the divisions.

L15 - Local Scope Variable Shadowing

Criticality	minor
Location	contracts/Eggpot.sol#L115

Description

The are variables that are defined in the local scope containing the same name from an upper scope.

```
totalSupply
```

Recommendation

The local variables should have different names from the upper scoped variables.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	External	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
Eggpot	Implementation	ERC20, Ownable		
	<Constructor>	Public	Payable	ERC20
	<Receive Ether>	External	Payable	-
	addPresaleAddressForExclusions	External	✓	onlyOwner
	enableTrading	External	✓	onlyOwner
	removeLimits	External	✓	onlyOwner
	enableLimits	External	✓	onlyOwner
	setJackpotEnabled	External	✓	onlyOwner
	manageRestrictedWallets	External	✓	onlyOwner
	updateMaxBuyAmount	External	✓	onlyOwner
	updateMaxSellAmount	External	✓	onlyOwner
	updateMaxWallet	External	✓	onlyOwner
	updateSwapTokensAtAmount	External	✓	onlyOwner
	_excludeFromMaxTransaction	Private	✓	
	airdropToWallets	External	✓	onlyOwner
	setNumberOfBuysForJackpot	External	✓	onlyOwner
	excludeFromMaxTransaction	External	✓	onlyOwner
	setAutomatedMarketMakerPair	External	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	updateBuyFees	External	✓	onlyOwner
	updateSellFees	External	✓	onlyOwner

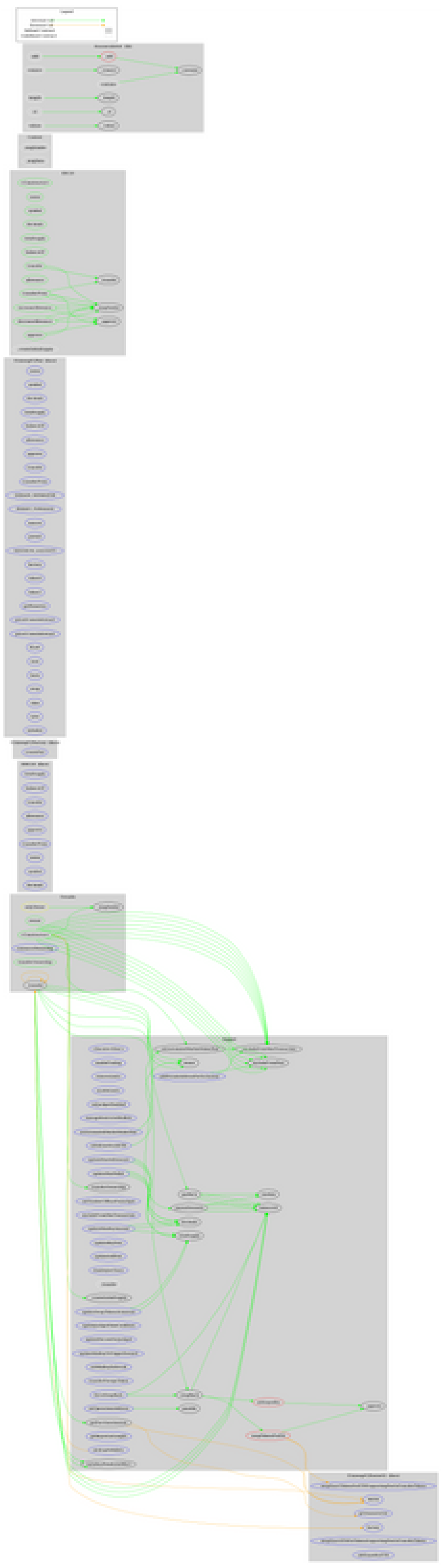
	disableJeetTaxes	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	_transfer	Internal	✓	
	earlyBuyPenaltyInEffect	Public		-
	getPurchaseAmount	Public		-
	gasBurn	Private	✓	
	payoutRewards	Private	✓	
	random	Private		
	updateJackpotTimeCooldown	External	✓	onlyOwner
	updatePercentForJackpot	External	✓	onlyOwner
	updateMinBuyToTriggerReward	External	✓	onlyOwner
	setMinBuyEnforced	External	✓	onlyOwner
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	swapBack	Private	✓	
	transferForeignToken	External	✓	onlyOwner
	withdrawStuckETH	External	✓	onlyOwner
	setOperationsAddress	External	✓	onlyOwner
	forceSwapBack	External	✓	onlyOwner
	getBuyerListLength	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
	name	External		-
	symbol	External		-
	decimals	External		-
IUniswapV2Factory	Interface			
	createPair	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	✓	-
IUniswapV2Router02	Interface			
	factory	External		-
	WETH	External		-

	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	addLiquidityETH	External	Payable	-
	getAmountsOut	External		-
ERC20	Implementation	Context, IERC20		
	<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	✓	
	_createInitialSupply	Internal	✓	
	_approve	Internal	✓	
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
EnumerableSet	Library			
	_add	Private	✓	
	_remove	Private	✓	
	_contains	Private		
	_length	Private		
	_at	Private		

	_values	Private		
	add	Internal	✓	
	remove	Internal	✓	
	contains	Internal		
	length	Internal		
	at	Internal		
	values	Internal		

Contract Flow



Domain Info

Domain Name	eggpot.io
Registry Domain ID	40e9b1c23d66463e9de89405c5e6ad50-DONUTS
Creation Date	2022-08-16T12:02:53Z
Updated Date	2022-08-16T12:09:39Z
Registry Expiry Date	2023-08-16T12:02:53Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	https://www.namecheap.com/
Registrar	NameCheap, Inc.
Registrar IANA ID	1068

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

There are some functions that can be abused by the owner like stopping transactions, transferring funds to the team’s wallet and massively blacklisting addresses. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. There is also a limit of max 15% buy fees and max limit of 20% for sell fees. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

The contract has a reward mechanism for every buyer. If the reward mechanism is enabled, the users that buy tokens greater than a threshold are applicable to win.

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