Cyberscope

Audit Report

CrytoBox

June 2022

Type	BEP20
Network	BSC
Address	0xfD5ecb7b36313B606a6d6Ba60858514f40E1751C
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	CBOX
Compiler Version	v0.8.15+commit.e14f2714
Optimization	200 runs
Licence	Unlicense
Explorer	https://bscscan.com/token/0xfD5ecb7b36313B606a6d6Ba60858514f40E1751C
Symbol	CBOX
Decimals	9
Total Supply	1,000,000,000
Domain	cbox.finance

Source Files

Filename	**SHA256**
contract.sol	1fbc60a692c14bf114b9c561d9665e1d66cfab7da57586336db913ba775e298b

Audit Updates

Initial Audit	23rd 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#L583

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 `maxTxAmount` or `maxTokenPerWallet` to zero.

```
if(!isExcludedFromMaxTokenPerTx[from])
{
	require(amount <= maxTxAmount, "Transfer amount exceeds the
maxTxAmount.");
}

if(!isExcludedFromMaxTokenPerWallet[to] && !automatedMarketMakerPairs[to]){
	uint256 balanceRecepient = balanceOf(to);
	require(balanceRecepient + amount <= maxTokenPerWallet, "Exceeds maximum
token per wallet limit");
}
```

Recommendation

The contract could embody a check for not allowing setting the _maxTxAmount or maxTokenPerWallet 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.

ELFM - Exceed Limit Fees Manipulation

Criticality	minor
Location	contract.sol#L471,481,491,501

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling one of the fee setter functions with 3000.

```
function setDevelopmentFee(uint256 buy, uint256 sell, uint256 p2p) external
onlyOwner {
      require(marketingFee[0].add(liquidityFee[0]).add(rewardsFee[0]).add(buy)
<= 3000 , "Max fee limit reached for 'BUY'");
      require(marketingFee[1].add(liquidityFee[1]).add(rewardsFee[1]).add(sell)
<= 3000 , "Max fee limit reached for 'SELL'");
      require(marketingFee[2].add(liquidityFee[2]).add(rewardsFee[2]).add(p2p)
<= 3000 , "Max fee limit reached for 'P2P'");

      developmentFee[0] = buy;
      developmentFee[1] = sell;
      developmentFee[2] = p2p;
}
```

Recommendation

The contract could embody a check for 25% 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.

BC - Blacklisted Contracts

Criticality	medium
Location	contract.sol#L578

Description

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

```
require(!isBlackListed[from], "BEP20: transfer to is blacklisted");
require(!isBlackListed[to], "BEP20: transfer from is 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
	RV	Randomization Vulnerability
	CO	Code Optimization
	CR	Code Repetition
	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

RV - Randomization Vulnerability

Criticality	minor
Location	contract.sol#L887

Description

The contract is using an on chain technique in order to determine random numbers. The blockchain runtime environment is fully deterministic, as a result, the pseudo-random numbers could be predicted.

```
function random(uint256 number) public view returns(uint256){
       return
uint256(keccak256(abi.encodePacked(block.timestamp,block.difficulty,
msg.sender))) % number;
}
```

Recommendation

The contract could use an advanced randomization technique that quarandees an acceptable randomization factor. For instance, the Chainlink VRF (Verifiable Random Function).

https://docs.chain.link/docs/chainlink-vrf/

CO - Code Optimization

Criticality	minor
Location	contract.sol#L654

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 initially calculate the corresponding index and access all the fees with that.

```
uint256 _marketingFee = amount.mul(p2p ? marketingFee[2] : sell ?
marketingFee[1] : marketingFee[0]).div(10000);
            marketingFeeTotal = marketingFeeTotal.add(_marketingFee);

uint256 _liquidityFee = amount.mul(p2p ? liquidityFee[2] : sell ?
liquidityFee[1] : liquidityFee[0]).div(10000);
            liquidityFeeTotal = liquidityFeeTotal.add(_liquidityFee);

uint256 _developmentFee = amount.mul(p2p ? developmentFee[2] : sell ?
developmentFee[1] : developmentFee[0]).div(10000);
            developmentFeeTotal = developmentFeeTotal.add(_developmentFee);

uint256  _rewardsFee = amount.mul(p2p ? rewardsFee[2] : sell ? rewardsFee[1] :
rewardsFee[0]).div(10000);
            rewardsFeeTotal = rewardsFeeTotal.add(_rewardsFee);
```

Recommendation

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

CR - Code Repetition

Criticality	minor
Location	contract.sol#L727

Description

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

The buyBox method calculates the prize reward according to the boxID. The prize calculation statements are the same across the three boxes. The contract could move the calculation in one method that accepts the box as a parameter. As a result, the code size in the `buyBox` method will be decreased three times.

```
if(boxID==0)
{
       for(uint256 i = 0; i < box1.length; i++) {

              uint256 min  = box1[i][0];
              uint256 max  = box1[i][1];
              uint256 mult = box1[i][2];
//...
```

Recommendation

Create an internal function that contains the code segment and remove it from all the sections.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L36,41,85,89,93,105,110,114,119,125,130,462,467,511,534,539,544,700,704,708,727

Description

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

```
buyBox
resetFeeTotal
migrateBNB
transferTokens
setAutomatedMarketMakerPair
excludeFromMaxTokenPerWallet
excludeFromMaxTxAmount
setPoolShare
setSwapEnable
...
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L335,340

Description

Constant state variables should be declared constant to save gas.

```
tokenToRewards
DEAD
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L172,467,565,570,335

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.

```
DEAD
_wallet
_enabled
WETH
```

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#L340

Description

There are segments that contain unused state variables.

```
tokenToRewards
```

Recommendation

Remove unused state variables.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L452,457,462

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.

```
maxTokenPerWallet = amount
maxTxAmount = amount
swapTokensAtAmount = amount
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L159,280,287

Description

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

```
toInt256Safe
toUint256Safe
_setupDecimals
```

Recommendation

Remove unused functions.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L575

Description

Performing divisions before multiplications may cause lose of prediction.

```
liquidityHalf = tokenToLiquidity.div(2)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

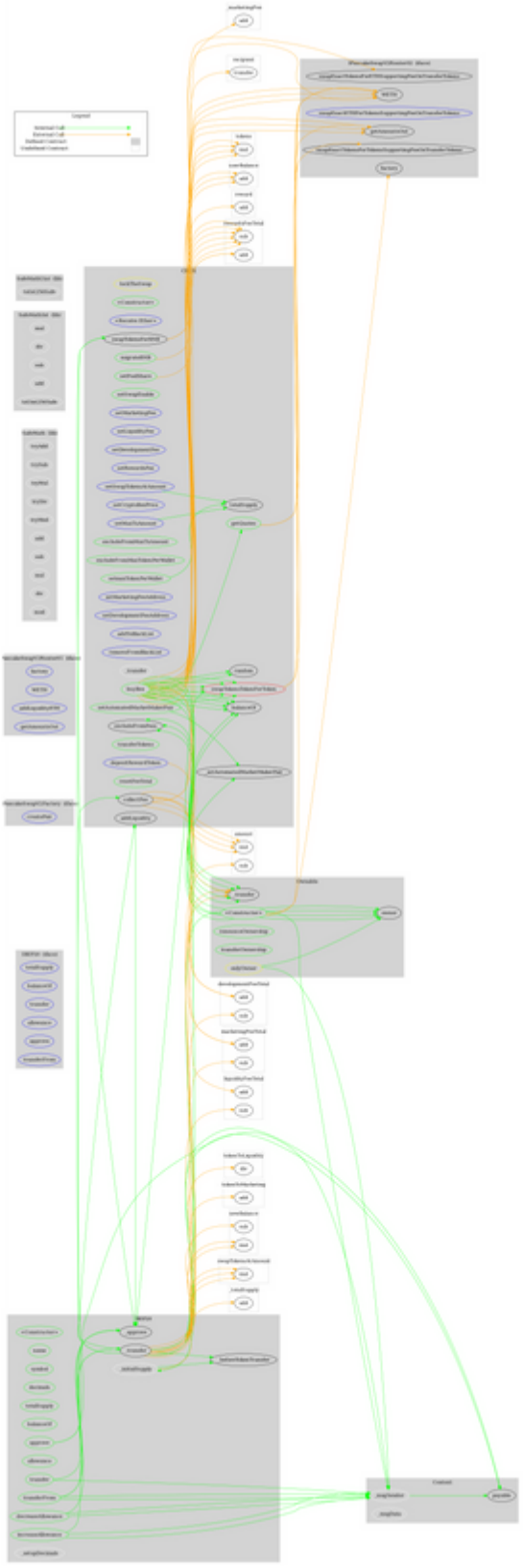
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IBEP20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
BEP20	Implementation	Context, IBEP20		
	<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	✓	
	_initialSupply	Internal	✓	
	_approve	Internal	✓	
	_setupDecimals	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
IPancakeSwapV2Factory	Interface			
	createPair	External	✓	-
IPancakeSwapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	getAmountsOut	External		-
IPancakeSwapV2Router02	Interface	IPancakeSwapV2Router01		
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		

	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
SafeMathInt	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		
	add	Internal		
	toUint256Safe	Internal		
SafeMathUint	Library			
	toInt256Safe	Internal		
CBOX	Implementation	BEP20, Ownable		
	<Constructor>	Public	✓	BEP20
	<Receive Ether>	External	Payable	-
	setSwapTokensAtAmount	External	✓	onlyOwner
	setMaxTxAmount	External	✓	onlyOwner
	setmaxTokenPerWallet	Public	✓	onlyOwner
	setSwapEnable	Public	✓	onlyOwner
	setMarketingFee	External	✓	onlyOwner
	setLiquidityFee	External	✓	onlyOwner
	setDevelopmentFee	External	✓	onlyOwner
	setRewardsFee	External	✓	onlyOwner
	setPoolShare	Public	✓	onlyOwner
	setCryptoBoxPrice	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	excludeFromMaxTxAmount	Public	✓	onlyOwner
	excludeFromMaxTokenPerWallet	Public	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	

	setMarketingFeeAddress	External	✓	onlyOwner
	setDevelopmentFeeAddress	External	✓	onlyOwner
	addToBlackList	External	✓	onlyOwner
	removeFromBlackList	External	✓	onlyOwner
	_transfer	Internal	✓	
	collectFee	Private	✓	
	addLiquidity	Private	✓	lockTheSwap
	swapTokensForBNB	Private	✓	lockTheSwap
	swapTokensTokenForToken	Private	✓	lockTheSwap
	transferTokens	Public	✓	onlyOwner
	migrateBNB	Public	✓	onlyOwner
	resetFeeTotal	Public	✓	onlyOwner
	getQuotes	Public		-
	buyBox	Public	✓	-
	random	Public		-
	depositRewardToken	External	✓	-

Contract Flow



Domain Info

Domain Name	cbox.finance
Registry Domain ID	c8130a82230c432caff64a0d2238b23b-DONUTS
Creation Date	2022-06-06T08:52:33Z
Updated Date	2022-06-11T08:52:53Z
Registry Expiry Date	2023-06-06T08:52:33Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	https://www.namecheap.com/
Registrar	NameCheap, Inc.
Registrar IANA ID	1068

The domain has been created 17 days before the creation of the audit. It will expire in 12 months.

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, manipulating fees and blacklisting addresses. 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 implements a lottery feature where users have the ability to deposit tokens in order to win a reward. The winner is picked pseudo-randomly using an on-chain technique. The reward payment pseudo-randomly picks one of the CBOX, SHIBA, DOGE, LTC, ETH, BTC as a currency.

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