Cyberscope

Audit Report

LUSAIL STADIUM TOKEN

November 2022

Type	BEP20
Network	BSC
Address	0xe1E3C76af115Fb6aeaEE39315a741F43d6451774
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	Lusail
Compiler Version	v0.6.12+commit.27d51765
Optimization	200 runs
Licence	Unlicense
Explorer	https://bscscan.com/token/0xe1E3C76af115Fb6aeaEE39315a741F43d6451774
Symbol	Lusail
Decimals	9
Total Supply	1,000,000,000
Domain	lusailcoin.info

Source Files

Filename	**SHA256**
contract.sol	74cab134fc231195a9e508ab33976ba90985f72a7fbcd9f117834a9bcca2ef04

Audit Updates

Initial Audit	5th December 2022
Corrected	

Contract Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

ST - Stops Transactions

Criticality	minor / informative
Location	contract.sol#L1199,1239
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users excluding the owner. There is a chance after token launch that this check is true and the user will get blacklisted. As a result the next time he tries to make a transaction it will fail.

```
if (launchBlock + killNum > block.number) addBot(to);
```

TotalTax is fixed to 5, but BurnFee can have values higher than 5. As a result the transfer will revert.

```
uint256 denominator = (TotalTax.sub(BurnFee).sub(ReflectionFee)).mul(2);
```

Recommendation

The contract should remove the check that adds address to blacklist. The contract should check first if BurnFee is greater than 5.

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 - Blacklists Addresses

Criticality	medium
Location	contract.sol#L1222
Status	Unresolved

Description

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

```
function writeBlackList(address BotAddress, bool isBot) external onlyOwner {
    if (isBot == true)
        require(
            BotAddress != mainPair && BotAddress != address(this),
            "You cannot blacklist this address!!"
        );
    blackList[BotAddress] = isBot;
}
```

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 / Informative

Severity	Code	Description	Status
●	ZD	Zero Division	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L05	Unused State Variable	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved

ZD - Zero Division

Criticality	critical
Location	contract.sol#L1239
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert. TotalTax has a fixed value of 5. The sum of BurnFee and ReflectionFee can be 5 so the result of the operation will be zero.

```
uint256 denominator = (TotalTax.sub(BurnFee).sub(ReflectionFee)).mul(2);
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor / informative
Location	contract.sol#L610,637,647,206,641,612,611,629,639,606
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
_name
DevFeeWallet
numTokensSellToAddToLiquidity
_previousOwner
BurnFeeWallet
_decimals
_symbol
TotalTax
MarketingFeeWallet
...
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contract.sol#L617,888,596,620,385,910,884,637,1207,339,626,1078,1062,643,631,1094,614,306,308,623,641,1086,886,639,646,1070,629,887,885,1222
Status	Unresolved

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.

```
LiquidityFee
_BurnFee
Free
DevFee
WETH
_enabled
_ReflectionFee
DevFeeWallet
Launch
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-conventions.

L05 - Unused State Variable

Criticality	minor / informative
Location	contract.sol#L206
Status	Unresolved

Description

There are segments that contain unused state variables.

```
_previousOwner
```

Recommendation

Remove unused state variables.

L07 - Missing Events Arithmetic

Criticality	minor / informative
Location	contract.sol#L1213,905,883
Status	Unresolved

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.

```
killNum = killNumber
_maxTxAmount = _tTotal.mul(maxTxPercent).div(10 ** 2)
ReflectionFee = _ReflectionFee
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor / informative
Location	contract.sol#L120,146,160,173,110,138,131
Status	Unresolved

Description

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

```
sendValue
functionCallWithValue
_functionCallWithValue
isContract
functionCall
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor / informative
Location	contract.sol#L1222
Status	Unresolved

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.

```
isBot == true
```

Recommendation

Remove the equality to the boolean constant.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	

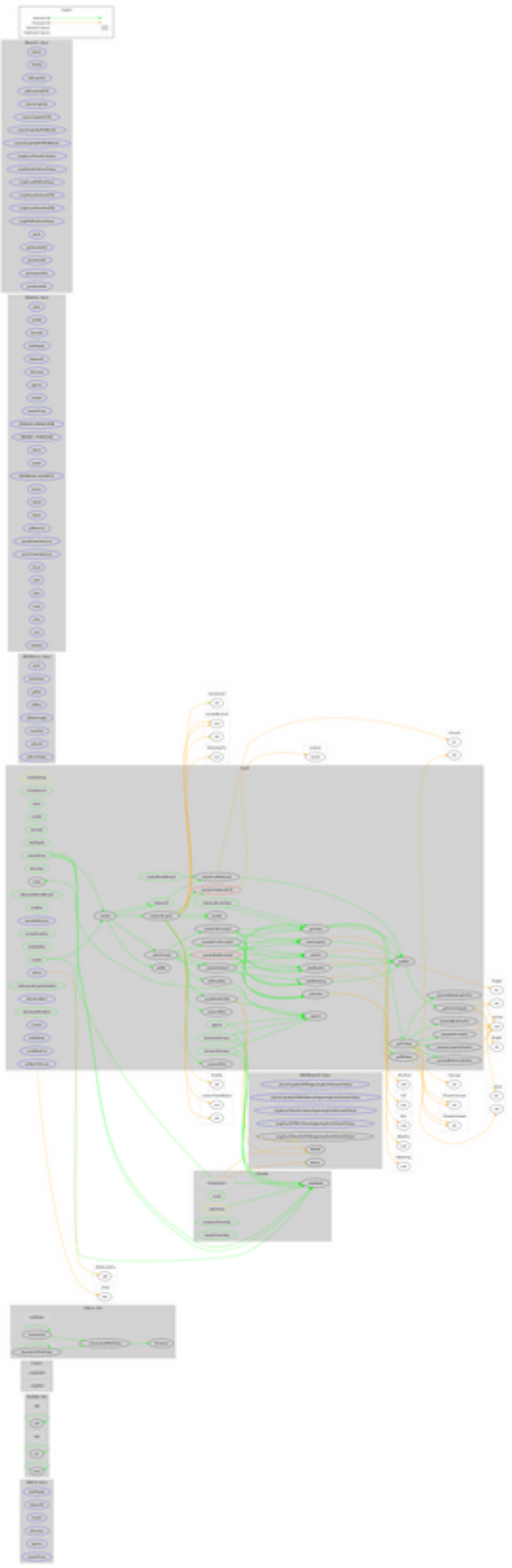
	_functionCallWithValue	Private	✓	
Ownable	Implementation	Context		
	<Constructor>	Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IDEXFactory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IMainPair	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	✓	-
IRouter01	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		-
IDEXRouter02	Interface	IRouter01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-

	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
Lusail	Implementation	Context, IERC20, Ownable		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	isExcludedFromReward	Public		-
	totalFees	Public		-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-
	excludeFromReward	Public	✓	onlyOwner
	includeInReward	External	✓	onlyOwner
	_transferBothExcluded	Private	✓	
	excludeFromFee	Public	✓	onlyOwner
	includeInFee	Public	✓	onlyOwner
	setFees	External	✓	onlyOwner
	setMaxTxPercent	External	✓	onlyOwner
	setSwapAndLiquifyEnabled	Public	✓	onlyOwner
	<Receive Ether>	External	Payable	-

	_reflectFee	Private	✓	
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidity	Private	✓	
	_takeDev	Private	✓	
	_takeBurnFee	Private	✓	
	_takeMarketing	Private	✓	
	calculateDevFeeFee	Private		
	calculateBurnFeeeFee	Private		
	calculateMarketingFeeFee	Private		
	calculateReflectionFeeFee	Private		
	calculateLiquidityFeeeFee	Private		
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	
	isExcludedFromFee	Public		-
	_approve	Private	✓	
	_transfer	Private	✓	
	Launch	External	✓	onlyOwner
	setKillNum	External	✓	onlyOwner
	addBot	Internal	✓	
	writeBlackList	External	✓	onlyOwner
	transferToAddressETH	Private	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	

Contract Flow



Domain Info

Domain Name	lusailcoin.info
Registry Domain ID	9b54260acbae4244a34efcfc0139e800-DONUTS
Creation Date	2022-12-03T11:28:44Z
Updated Date	2022-12-03T11:28:44Z
Registry Expiry Date	2023-12-03T11:28:44Z
Registrar WHOIS Server	whois.godaddy.com/
Registrar URL	http://www.godaddy.com/domains/search.aspx?ci=8990
Registrar	GoDaddy.com, LLC
Registrar IANA ID	146

The domain was created 2 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 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 allows users to sell up to 99,99% of their holdings.

```
uint256 maxSell = balanceOf(from).mul(9999).div(10000);
if (amount > maxSell) amount = maxSell;
```

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope’s prior written consent. This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope’s position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io