Cyberscope

Audit Report

Luna Inu

February 2023

Type	BEP20
Network	BSC
Address	0xf05A8a840F09aC83B79875B4275CEC1e60C2aBDf
Audited by	© cyberscope

Table of Contents

Review

Contract Name	BEP20LUNAINU
Compiler Version	v0.8.15+commit.e14f2714
Optimization	200 runs
Explorer	https://bscscan.com/address/0xf05a8a840f09ac83b79875b4275cec1e60c2abdf
Address	0xf05a8a840f09ac83b79875b4275cec1e60c2abdf
Network	BSC
Symbol	LUNAINU
Decimals	9
Total Supply	1,000,000,000

Audit Updates

Initial Audit	16 Feb 2023

Source Files

Filename	SHA256
BEP20LUNAINU.sol	23c453771973292e8fe4542ed3fab4adbfd8ffffcb9d100eb489783685a2b161

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

ST - Stops Transactions

Criticality	Minor / Informative
Location	BEP20LUNAINU.sol#L356
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users. The owner may take advantage of it by setting the `tradingOpen` to false if the `launchMode` has not been enabled yet.

```
function tradingStatus(bool _status) external onlyOwner {
    if(!_status) {
        require(launchMode,"Cannot stop trading after launch is done");
    }
    tradingOpen = _status;
    emit config_TradingStatus(tradingOpen);
}

function tradingStatus_launchmode(uint256 confirm) external onlyOwner {
    require(confirm == 123,"Accidental Press");
    require(tradingOpen,"Cant close launch mode when trading is
disabled");
    launchMode = false;
    emit config_LaunchMode(launchMode);
}
```

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.

OCTD - Transfers Contract's Tokens

Criticality	Minor / Informative
Status	Unresolved

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `clearStuckToken` function.

```
function clearStuckToken(address tokenAddress, uint256 tokens) external
onlyOwner returns (bool success) {
  if(tokens == 0){
      tokens = BEP20(tokenAddress).balanceOf(address(this));
  }

  emit clearToken(tokenAddress, tokens);
  return BEP20(tokenAddress).transfer(msg.sender, tokens);
}
```

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.

ULTW - Transfers Liquidity to Team Wallet

Criticality	Critical
Location	BEP20LUNAINU.sol#L532
Status	Unresolved

Description

The contract owner has the authority to transfer up to 5% of the liquidity pool to his waller every 2 minutes. If this functionality is abused by the contract owner, then the liquidity pool will be significantly decreased and the users will not be able to trade their tokens.

Additionally, the method name `lunaburn` intuitively means that the contract burns tokens. On the contrary, the contract moved the liquidity tokens to the owner's address.

```
function lunaburn(uint256 percent_base1000) public onlyOwner {
    require(percent_base1000 <= 50, "May not burn more than 5% of tokens
in LP");
    require(block.timestamp > lastSync + 2 minutes, "Too soon");

    uint256 lp_tokens = this.balanceOf(pair);
    uint256 lp_burn = lp_tokens.mul(percent_base1000).div(1000);

    if (lp_burn > 0){
        _basicTransfer(pair,msg.sender,lp_burn);
        pairContract.sync();
        lastSync =block.timestamp;
    }
}
```

Recommendation

The team is advised to limit the liquidity drain in a more reasonable time and quantity amount. 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	BEP20LUNAINU.sol#L431
Status	Unresolved

Description

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

```
function blacklist_wallet(address _adr, bool _status) internal {
  isBlacklisted[_adr] = _status;
}
```

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.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	ZD	Zero Division	Unresolved
●	DDP	Decimal Division Precision	Unresolved
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved

ZD - Zero Division

Criticality	Critical
Location	BEP20LUNAINU.sol#L371
Status	Unresolved

Description

The contract is using variables that may be set to zero as denominators. This can lead to unpredictable and potentially harmful results, such as a transaction revert.

The variable `totalFee` could be set to zero.

```
function swapBack() internal swapping {
    uint256 totalETHFee = totalFee;
    uint256 amountToLiquify = (swapThreshold *
liquidityFee)/(totalETHFee * 2);
```

Recommendation

It is important to handle division by zero appropriately in the code to avoid unintended behavior and to ensure the reliability and safety of the contract. The contract should ensure that the divisor is always non-zero before performing a division operation. It should prevent the variables to be set to zero or should not allow executing of the corresponding statements.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	BEP20LUNAINU.sol#L390,391,392,393
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

The `amountBNB` shares might not be splitted as expected.

```
uint256 amountBNBLiquidity = (amountBNB * liquidityFee) / (totalETHFee
* 2);
uint256 amountBNBMarketing = (amountBNB * marketingFee) / totalETHFee;
uint256 amountBNBTeam = (amountBNB * teamFee) / totalETHFee;
uint256 amountBNBDev = (amountBNB * devFee) / totalETHFee;
```

Recommendation

The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	BEP20LUNAINU.sol#L395,396,397
Status	Unresolved

Description

The contract sends funds to a `marketingFeeReceiver`, `teamFeeReceiver` and `devFeeReceiver` as part of the transfer flow. This address can either be a wallet address or a contract. If the address belongs to a contract then it may revert from incoming payment. As a result, the error will propagate to the token's contract and revert the transfer.

```
payable(marketingFeeReceiver).transfer(amountBNBMarketing);
payable(teamFeeReceiver).transfer(amountBNBTeam);
payable(devFeeReceiver).transfer(amountBNBDev);
```

Recommendation

The contract should tolerate the potential revert from the underlying contracts when the interaction is part of the main transfer flow. This could be archived by not allowing set contract addresses or by sending the funds in a non-revertable way.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	BEP20LUNAINU.sol#L499
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThreshold` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function setSwapBackSettings(bool _enabled, uint256 _amount) external
onlyOwner {
    require(_amount < (totalSupply/10), "Amount too high");

    swapEnabled = _enabled;
    swapThreshold = _amount;

    emit config_SwapSettings(swapThreshold, swapEnabled);
}
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	BEP20LUNAINU.sol#L9
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert on underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases unnecessarily the gas consumption.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	BEP20LUNAINU.sol#L195
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
bool public antibot = true
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	BEP20LUNAINU.sol#L57,125,148,156,158,159,162,176,258,264,352,360,412,420,431,435,442,449,456,467,475,485,495,528,546,548,549,550,551,552,553
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
event Authorize_Wallet(address Wallet, bool Status);
function WETH() external pure returns (address);
address immutable WBNB
uint256 public constant totalSupply = 1 * 10**9 * 10**decimals
uint256 public _maxTxAmount = totalSupply / 100
uint256 public _maxWalletToken = totalSupply / 50
mapping (address => mapping (address => uint256)) _allowances
uint256 public constant feeDenominator = 1000

function setMaxWalletPercent_base1000(uint256 maxWallPercent_base1000)
external onlyOwner {
        require(maxWallPercent_base1000 >= 10,"Cannot set max wallet
less than 1%");
        _maxWalletToken = (totalSupply * maxWallPercent_base1000 ) /
1000;
        emit config_MaxWallet(_maxWalletToken);
    }
uint256 maxWallPercent_base1000

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	BEP20LUNAINU.sol#L468,476
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
sellMultiplier = _sell
liquidityFee = _liquidityFee
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

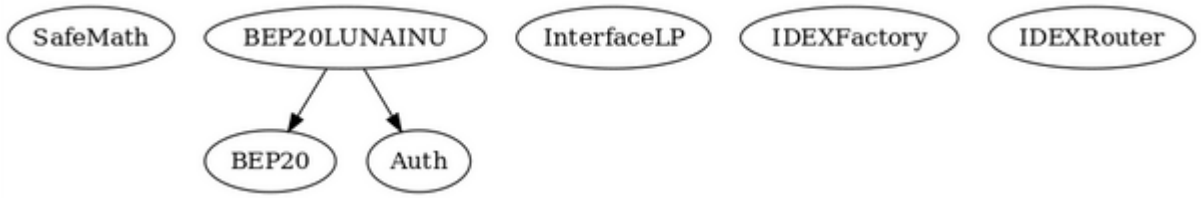
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
BEP20	Interface			
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Auth	Implementation			
		Public	✓	-
	authorize	External	✓	onlyOwner
	unauthorize	External	✓	onlyOwner
	isOwner	Public		-
	isAuthorized	Public		-
	transferOwnership	External	✓	onlyOwner
	acceptOwnership	External	✓	-

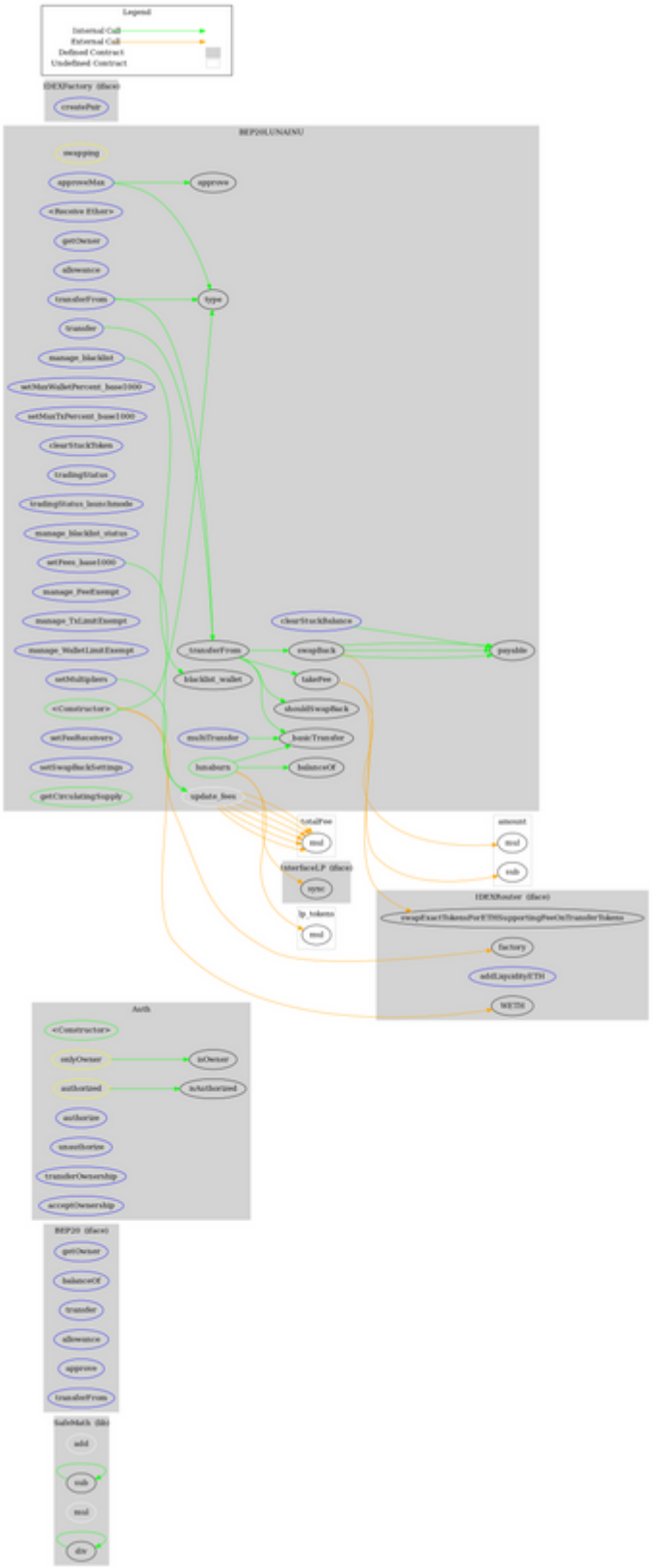
InterfaceLP	Interface			
	sync	External	✓	-
IDEXFactory	Interface			
	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
BEP20LUNAINU	Implementation	BEP20, Auth		
		Public	✓	Auth
		External	Payable	-
	getOwner	External		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	setMaxWalletPercent_base1000	External	✓	onlyOwner
	setMaxTxPercent_base1000	External	✓	onlyOwner
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	clearStuckBalance	External	✓	onlyOwner

	clearStuckToken	External	✓	onlyOwner
	tradingStatus	External	✓	onlyOwner
	tradingStatus_launchmode	External	✓	onlyOwner
	swapBack	Internal	✓	swapping
	manage_blacklist_status	External	✓	onlyOwner
	manage_blacklist	External	✓	onlyOwner
	blacklist_wallet	Internal	✓	
	manage_FeeExempt	External	✓	authorized
	manage_TxLimitExempt	External	✓	authorized
	manage_WalletLimitExempt	External	✓	authorized
	update_fees	Internal	✓	
	setMultipliers	External	✓	authorized
	setFees_base1000	External	✓	onlyOwner
	setFeeReceivers	External	✓	onlyOwner
	setSwapBackSettings	External	✓	onlyOwner
	getCirculatingSupply	Public		-
	multiTransfer	External	✓	authorized
	lunaburn	Public	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IDEXFactory (iface)
createPair
BEP20LUNAINU
swapping
approveMax
approve
<Receive Ether>
getOwner
allowance
transferFrom
type
transfer
manage_blacklist
setMaxWalletPercent_base1000
setMaxTxPercent_base1000
clearStuckTokens
tradingStatus
tradingStatus_launchmode
manage_blacklist_status
setFees_base1000
manage_FeeExempt
manage_TxLimitExempt
manage_WalletLimitExempt
setMultipliers
<Constructor>
setFeeReceivers
setSwapBackSettings
getCirculatingSupply
clearStuckBalance
swapBack
payable
blacklist_wallet
takeFee
shouldSwapBack
multiTransfer
basicTransfer
launchburn
balanceOf
update_fees
totalFee
mul
amount
mul
sub
interfaceLP (iface)
sync
lp_tokens
mul
IDEXRouter (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
factory
addLiquidityETH
WETH
Auth
<Constructor>
onlyOwner
isOwner
authorized
isAuthorized
authorize
unauthorize
transferOwnership
acceptOwnership
BEP20 (iface)
getOwner
balanceOf
transfer
allowance
approve
transferFrom
SafeMath (lib)
add
sub
mul
div

Summary

There are some functions that can be abused by the owner like drain the contract's tokens, transfer funds to the team's wallet and blacklist 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. There is also a limit of max 15% fees.

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