Cyberscope

Audit Report

Madiba

March 2022

Type	BEP20
Network	BSC
Address	0xE5094E58042e5332CfD25Ed18e4d805e86Fb8f05
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	MadibaBEP20
Compiler Version	v0.8.4+commit.c7e474f2
Optimization	1000 runs
Licence	
Explorer	https://bscscan.com/token/0xE5094E58042e5332CfD25Ed18e4d805e86Fb8f05
Symbol	DIBA
Decimals	18
Total Supply	505,000,000
Source	@openzeppelin/contracts/access/Ownable.sol, @openzeppelin/contracts/token/ERC20/IERC20.sol, @openzeppelin/contracts/utils/Address.sol, @openzeppelin/contracts/utils/Context.sol, @openzeppelin/contracts/utils/math/SafeMath.sol, @uniswap/lib/contracts/libraries/TransferHelper.sol, contracts/extensions/BaseToken.sol, contracts/interfaces/IMadibaSwap.sol, contracts/MadibaBEP20.sol
Domain	madiba.app

Audit Updates

Initial Audit	10th March 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

ULTW - Unlimited Liquidity to Team Wallet

Criticality	critical
Location	contract.sol#L338

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 takeLiquidity() function

```
function takeLiquidity(uint256 tLiquidity) external onlyOperator {
    _balances[address(this)] = _balances[address(this)].add(tLiquidity);
}
```

Recommendation

The contract could embody a check for the 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.

MT - Mint Tokens

Criticality	critical
Location	contract.sol#L202,236,503,338

Description

The contract owner has the authority to mint tokens. The owner may take advantage in multiple ways. Even if there is a maximum mint cap in the mint funcion, the mint functionality is unlimited in combination with the burn functionality. That means that the contract owner can sequentially burn tokens from other wallets and mint tokens to the owner's wallet. As a result the contract tokens will be highly inflated.

```
function mint(address _receiver, uint256 amount) public onlyOperator {
    _mint(_receiver, amount);
}
```

```
function mintReward(address to) public onlyOperator {
    rewardReserveUsed = rewardReserveUsed.add(REWARD_RESERVE);
    if (rewardReserveUsed <= REWARD_RESERVE) {
        _mint(to, REWARD_RESERVE);
    }
}
```

```
if (rewardReserveUsed > REWARD_RESERVE) {
    rExcess = rewardReserveUsed.sub(REWARD_RESERVE);
    rewardReserveUsed = REWARD_RESERVE;
    _mint(owner(), rExcess);
}
```

The following is a sub-category of minting functionality. It mints new tokens without updating the total supply.

```
function takeLiquidity(uint256 tLiquidity) external onlyOperator {
    _balances[address(this)] = _balances[address(this)].add(tLiquidity);
}
```

```
function takeMarketingFee(uint256 tMarketing) external onlyOperator {
    if (tMarketing > 0) {
        _balances[_marketingFeeReceiver] = _balances[_marketingFeeReceiver]
            .add(tMarketing);
        emit Transfer(_msgSender(), _marketingFeeReceiver, tMarketing);
    }
}
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

BT - Burn Tokens

Criticality	critical
Location	contract.sol#L206

Description

The contract owner has the authority to burn tokens from any address. The owner may take advantage of it by calling the `burn` function. As a result the targeted contract address will lose the corresponding tokens.

```
function burn(address to, uint256 amount) external onlyOperator {
    _burn(to, amount);
}
```

Recommendation

The owner should carefully manage the credentials of the owner's account. We advised considering an extra-strong security mechanism that the actions may be quarantined by many users instead of one. The owner could also renounce the contract ownership for a period of time or pass the access to the zero address.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	MAL	Misused Algorithmic Logic
	L01	Public Function could be Declared External
	L02	State Variables could be Declared Constant
	L05	Unused State Variable
	L04	Conformance to Solidity Naming Conventions
	L09	Dead Code Elimination
	L11	Unnecessary Boolean equality
	L08	Tautology or Contradiction
	L13	Divide before Multiply Operation

MAL - Misused Algorithmic Logic

Criticality	critical
Location	contract.sol#L338,342

Description

The contract owner has the authority to increase the token holdings of the contract and marketing address without updating the total supply. This may break the balance of the token since the total supply will not equal the sum of all the balances.

```
function takeLiquidity(uint256 tLiquidity) external onlyOperator {
    _balances[address(this)] = _balances[address(this)].add(tLiquidity);
}

function takeMarketingFee(uint256 tMarketing) external onlyOperator {
    if (tMarketing > 0) {
        _balances[_marketingFeeReceiver] = _balances[_marketingFeeReceiver]
            .add(tMarketing);
        emit Transfer(_msgSender(), _marketingFeeReceiver, tMarketing);
    }
}
```

Recommendation

The total supply should reflect the mutations in the balances mapping.

L01 - Public Function could be Declared External

Criticality	minor
Location	@openzeppelin/contracts/access/Ownable.sol#L54,62 contracts/MadibaBEP20.sol#L118,127,136,145,154,172,185,202,236,243 and 9 more contracts/extensions/BaseToken.sol#L44,48

Description

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

```
symbol
name
holderInfo
closeWhitelist
setSwapAddress
setTreasuryAddress
setMarketingAddress
setStakingAddress
isExcludedFromFee
...
```

Recommendation

Use the external attribute for functions never called from the contract

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contracts/MadibaBEP20.sol#L60

Description

Constant state variables should be declared constant to save gas.

```
maxTxFeeBps
```

Recommendation

Add the constant attribute to state variables that never change.

L05 - Unused State Variable

Criticality	minor
Location	contracts/MadibaBEP20.sol#L34,36

Description

There are segments that contain unused state variables.

```
_excluded
_isExcluded
```

Recommendation

Remove unused state variables.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/MadibaBEP20.sol#L202,243,350,358,422,430,436,443,448,453 and 6 more

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.

```
_marketingFeeReceiver
_marketingFee
_liquidityFee
_holderAddress
_account
_newSwapAddress
_newAddress
_status
_operator
...
```

Recommendation

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

L09 - Dead Code Elimination

Criticality	minor
Location	@openzeppelin/contracts/utils/Address.sol#L80,90,109,123,169,179,142,152,27,55 and 1 more contracts/MadibaBEP20.sol#L321,302,253,350,358 @uniswap/lib/contracts/libraries/TransferHelper.sol#L7,20,47,33

Description

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

```
safeTransferFrom
safeTransferETH
safeTransfer
safeApprove
calculateMarketingFee
calculateLiquidityFee
_transferBothExcluded
_getValues
_getTValues
...
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor
Location	contracts/MadibaBEP20.sol#L453,485

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.

```
require(bool,string)(_isWhitelistClosed == false,Whitelisting is already
closed.)
require(bool,string)(_isWhitelistClosed == false,Whitelisting is no longer in
session.)
```

Recommendation

Remove the equality to the boolean constant.

L08 - Tautology or Contradiction

Criticality	minor
Location	contracts/MadibaBEP20.sol#L67

Description

Detects expressions that are tautologies or contradictions. For instance, an uint variable will always be greater than or equal to zero.

```
require(bool,string)(liquidityFeeBps_ >= 0,Invalid liquidity fee)
require(bool,string)(marketingFeeBps_ >= 0,Invalid marketing fee)
```

Recommendation

Fix the incorrect comparison by changing the value type or the comparison.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contracts/MadibaBEP20.sol#L485

Description

Performing divisions before multiplications may cause lose of prediction.

```
tokensHeld = holder.amount.div(10 ** decimals()).mul(dibaInBNB)
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

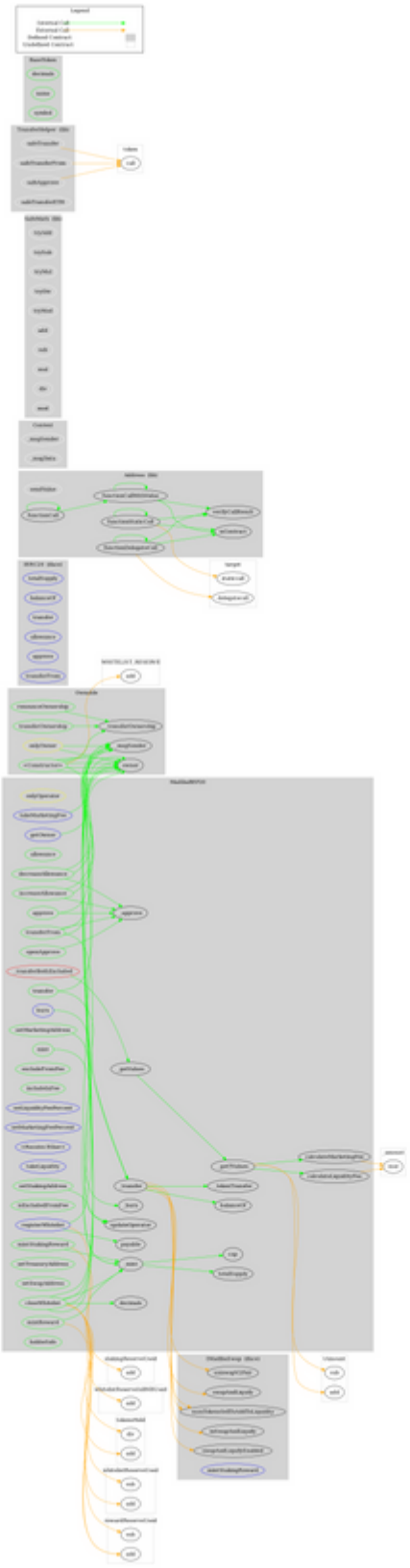
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	verifyCallResult	Internal		
Context	Implementation			

	_msgSender	Internal		
	_msgData	Internal		
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		
TransferHelper	Library			
	safeApprove	Internal	✓	
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeTransferETH	Internal	✓	
BaseToken	Implementation			
	decimals	Public		-
	name	Public		-
	symbol	Public		-
IMadibaSwap	Interface			
	numTokensSellToAddToLiquidity	External	✓	-
	inSwapAndLiquify	External	✓	-
	swapAndLiquifyEnabled	External	✓	-
	uniswapV2Pair	External	✓	-
	swapAndLiquify	External	✓	-

	mintStakingReward	External	✓	-
MadibaBEP20	Implementation	IERC20, Ownable, BaseToken		
	<Constructor>	Public	✓	-
	totalSupply	Public		-
	getOwner	External		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	openApprove	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	mint	Public	✓	onlyOperator
	burn	External	✓	onlyOperator
	cap	Public		-
	_mint	Private	✓	
	_burn	Private	✓	
	mintReward	Public	✓	onlyOperator
	mintStakingReward	Public	✓	onlyOperator
	_transferBothExcluded	Private	✓	
	excludeFromFee	Public	✓	onlyOperator
	includeInFee	Public	✓	onlyOperator
	setLiquidityFeePercent	External	✓	onlyOperator
	setMarketingFeePercent	External	✓	onlyOperator
	<Receive Ether>	External	Payable	-
	_getValues	Private		
	_getTValues	Private		
	takeLiquidity	External	✓	onlyOperator
	takeMarketingFee	External	✓	onlyOperator
	calculateLiquidityFee	Private		
	calculateMarketingFee	Private		
	isExcludedFromFee	Public		-

	_approve	Private	✓	
	_transfer	Private	✓	
	_tokenTransfer	Private	✓	
	updateOperator	Public	✓	onlyOperator
	setStakingAddress	Public	✓	onlyOperator
	setMarketingAddress	Public	✓	onlyOperator
	setTreasuryAddress	Public	✓	onlyOperator
	setSwapAddress	Public	✓	onlyOperator
	registerWhitelist	External	Payable	-
	closeWhitelist	Public	✓	onlyOperator
	holderInfo	Public		-

Contract Flow



Domain Info

Domain Name	madiba.app
Registry Domain ID	4876F2487-APP
Creation Date	2022-02-07T18:15:59Z
Updated Date	2022-02-24T11:43:06Z
Registry Expiry Date	2026-02-07T18:15:59Z
Registrar WHOIS Server	whois.namecheap.com
Registrar URL	https://www.namecheap.com/
Registrar	Namecheap Inc.
Registrar IANA ID	1068

The domain has been created about 1 month before the creation of the audit. It will expire in almost 4 years.

There is no public billing information, the creator is protected by the privacy settings.

Summary

The smart contract analysis reported some issues that can potentially break the balance of the token. There are also some functions that can be abused by the owner, like minting and burning tokens. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

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 provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io