Cyberscope

Audit Report

DecentralZone

June 2023

Network	ETH
Address	0x218c62A3cC89FF54C28287Facdd094dDa6931e00
Audited by	© cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	FSA	Fixed Swap Address	Unresolved
●	RLS	Redundant Liquidity Swaps	Unresolved
●	RVD	Redundant Variable Declaration	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	ULTW	Transfers Liquidity to Team Wallet	Unresolved
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Analysis 1
Diagnostics 2
Table of Contents 3
Review 5
- Audit Updates 5
- Source Files 5

Findings Breakdown 6
- ST - Stops Transactions 7
  - Description 7
  - Recommendation 7
- FSA - Fixed Swap Address 8
  - Description 8
  - Recommendation 8
- RLS - Redundant Liquidity Swaps 9
  - Description 9
  - Recommendation 9
- RVD - Redundant Variable Declaration 10
  - Description 10
  - Recommendation 10
- MEE - Missing Events Emission 11
  - Description 11
  - Recommendation 11
- RSW - Redundant Storage Writes 12
  - Description 12
  - Recommendation 12
- ULTW - Transfers Liquidity to Team Wallet 13
  - Description 13
  - Recommendation 13
- PTRP - Potential Transfer Revert Propagation 14
  - Description 14
  - Recommendation 14
- IDI - Immutable Declaration Improvement 15
  - Description 15
  - Recommendation 15
- L04 - Conformance to Solidity Naming Conventions 16
  - Description 16
  - Recommendation 17
- L07 - Missing Events Arithmetic 18
  - Description 18

Recommendation 18

L13 - Divide before Multiply Operation 19

Description 19

Recommendation 19

L14 - Uninitialized Variables in Local Scope 20

Description 20

Recommendation 20

L19 - Stable Compiler Version 21

Description 21

Recommendation 21

L20 - Succeeded Transfer Check 22

Description 22

Recommendation 22

Functions Analysis 23

Inheritance Graph 27

Flow Graph 28

Summary 29

Disclaimer 30

About Cyberscope 31

Review

Contract Name	DecentralZone
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://etherscan.io/address/0x218c62a3cc89ff54c28287facdd094dda6931e00
Address	0x218c62a3cc89ff54c28287facdd094dda6931e00
Network	ETH
Symbol	DZ
Decimals	18
Total Supply	800,000,000

Audit Updates

Initial Audit	29 Jun 2023

Source Files

Filename	SHA256
DecentralZone.sol	429970afc7214e8ca1c9d587ae3e442c7006c68c4a71951ede85735c8c83a6d8

Findings Breakdown

15
Critical 1
Medium 0
Minor / Informative 14

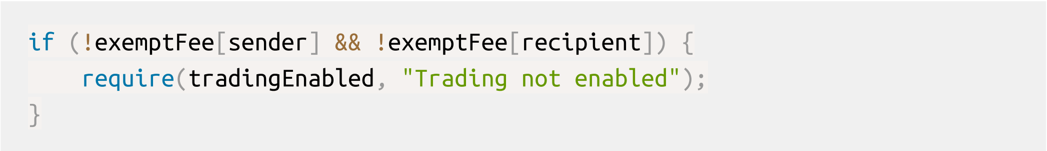
Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	1	0	0	0
● Medium	0	0	0	0
● Minor / Informative	14	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	DecentralZone.sol#L555
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enabled the owner will not be able to disable them again.



```
if (!exemptFee[sender] && !exemptFee[recipient]) {
    require(tradingEnabled, "Trading not enabled");
}
```

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. Some suggestions are:

- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

FSA - Fixed Swap Address

Criticality	Minor / Informative
Location	DecentralZone.sol#L480,481
Status	Unresolved

Description

The swap address is assigned once and it can not be changed. It is a common practice in decentralized exchanges to create new swap versions. A contract that cannot change the swap address may not be able to catch up to the upgrade. As a result, the contract will not be able to migrate to a new liquidity pool pair or decentralized exchange.

```
IRouter _router = IRouter(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
address _pair =
IFactory(_router.factory()).createPair(address(this),_router.WETH());
router = _router;
pair = _pair;
```

Recommendation

The team is advised to add the ability to change the pair and router address in order to cover potential liquidity pool migrations. It would be better to support multiple pair addresses so the token will be able to have the same behavior in all the decentralized liquidity pairs.

RLS - Redundant Liquidity Swaps

Criticality	Minor / Informative
Location	DecentralZone.sol#L607
Status	Unresolved

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 `tokenLiquidityThreshold` variable can be set to zero. Hence, the contract will trigger the swap functionality even if the contract's balance is zero. As a result, the contract performs redundant liquidity swaps.

```
if (contractBalance >= tokenLiquidityThreshold) { ... }
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

RVD - Redundant Variable Declaration

Criticality	Minor / Informative
Location	DecentralZone.sol#L461,462
Status	Unresolved

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 `BuyTaxes` and `SellTaxes` variables are declared and initialized but are not used in a meaningful way by the contract. As a result, these variables are redundant.

```
uint256 public BuyTaxes = taxes.marketing + taxes.liquidity;
uint256 public SellTaxes = sellTaxes.marketing + sellTaxes.liquidity;
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	DecentralZone.sol#L601,668,700,707,711,715
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
function Liquify(uint256 feeswap, Taxes memory swapTaxes) private
lockTheSwap { ... }
providingLiquidity = state;
tradingEnabled = true;
marketingWallet = newWallet;
exemptFee[_address] = true;
exemptFee[_address] = false;
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	DecentralZone.sol#L668,711,715
Status	Unresolved

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 modifies the state of certain variables even if their current state is equal to the one given as an argument. As a result, the contract performs redundant storage writes.

```
providingLiquidity = state;
exemptFee[_address] = true;
exemptFee[_address] = false;
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

ULTW - Transfers Liquidity to Team Wallet

Criticality	Minor / Informative
Location	DecentralZone.sol#L718
Status	Unresolved

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 `rescueEHT` method.

```
function rescueEHT() external onlyOwner {
    uint256 contractETHBalance = address(this).balance;
    payable(owner()).transfer(contractETHBalance);
}
```

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. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	DecentralZone.sol#L632
Status	Unresolved

Description

The contract sends funds to a `marketingWallet` 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.

```
if (marketingAmt > 0) {
    payable(marketingWallet).sendValue(marketingAmt);
}
```

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 achieved by not allowing set contract addresses or by sending the funds in a non-revertable way.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	DecentralZone.sol#L482,483
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
router
pair
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	DecentralZone.sol#L65,67,409,447,448,451,461,462,601,671,676,683,690,710,714
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.

```
mapping(address => uint256) internal _balances
mapping(address => mapping(address => uint256)) internal _allowances
function WETH() external pure returns (address);
uint256 public MaxTxAmount = 800000000 * 10**18
uint256 public MaxWalletSize = 800000000 * 10**18
address public constant deadWallet =
0x000000000000000000000000000000000000dEaD
uint256 public BuyTaxes = taxes.marketing + taxes.liquidity
uint256 public SellTaxes = sellTaxes.marketing + sellTaxes.liquidity

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, and 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	DecentralZone.sol#L673
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.

```
tokenLiquidityThreshold = new_amount * 10**decimals()
```

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.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	DecentralZone.sol#L622,623,630
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause a loss of prediction.

```
uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity)
uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	DecentralZone.sol#L563,564,566
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 feeswap
uint256 feesum
Taxes memory currentTaxes
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	DecentralZone.sol#L3
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.19;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	DecentralZone.sol#L725
Status	Unresolved

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.

```
IBEP20(tokenAdd).transfer(owner(), amount)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

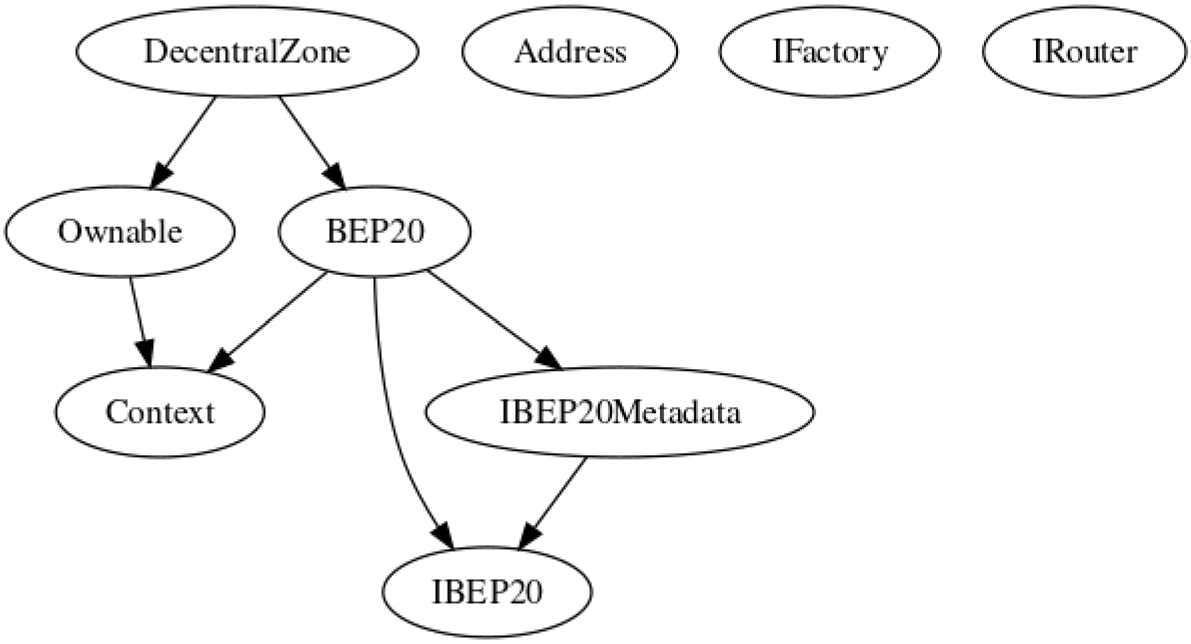
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IBEP20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IBEP20Metadata	Interface	IBEP20		
	name	External		-
	symbol	External		-
	decimals	External		-

BEP20	Implementation	Context, IBEP20, IBEP20Metadata		
		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	✓	
	_tokengeneration	Internal	✓	
	_approve	Internal	✓	
Address	Library			
	sendValue	Internal	✓	
Ownable	Implementation	Context		
		Public	✓	-

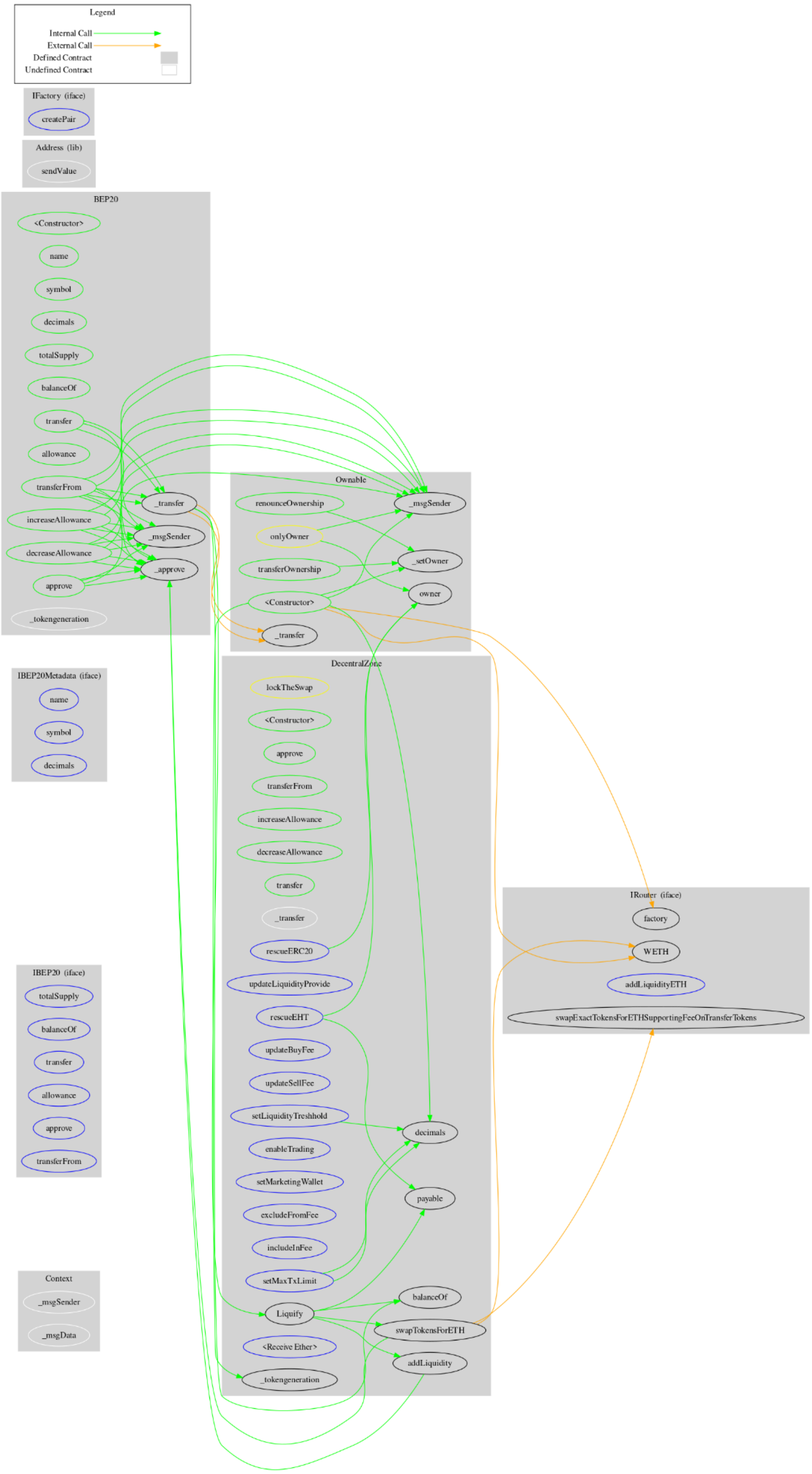
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	
IFactory	Interface			
	createPair	External	✓	-
IRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
DecentralZone	Implementation	BEP20, Ownable		
		Public	✓	BEP20
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	transfer	Public	✓	-
	_transfer	Internal	✓	
	Liquify	Private	✓	lockTheSwap

	swapTokensForETH	Private	✓	
	addLiquidity	Private	✓	
	updateLiquidityProvide	External	✓	onlyOwner
	setLiquidityTreshhold	External	✓	onlyOwner
	updateBuyFee	External	✓	onlyOwner
	updateSellFee	External	✓	onlyOwner
	setMaxTxLimit	External	✓	onlyOwner
	enableTrading	External	✓	onlyOwner
	setMarketingWallet	External	✓	onlyOwner
	excludeFromFee	External	✓	onlyOwner
	includeInFee	External	✓	onlyOwner
	rescueEHT	External	✓	onlyOwner
	rescueERC20	External	✓	onlyOwner
		External	Payable	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IFactory (iface)
createPair
Address (lib)
sendValue
BEP20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
increaseAllowance
decreaseAllowance
approve
_tokengeneration
_transfer
_msgSender
_approve
Ownable
renounceOwnership
onlyOwner
transferOwnership
<Constructor>
_msgSender
_setOwner
owner
_transfer
DecentralZone
lockTheSwap
<Constructor>
approve
transferFrom
increaseAllowance
decreaseAllowance
transfer
_transfer
rescueERC20
updateLiquidityProvide
rescueEHT
updateBuyFee
updateSellFee
setLiquidityTreshhold
enableTrading
setMarketingWallet
excludeFromFee
includeInFee
setMaxTxLimit
Liquify
<Receive Ether>
_tokengeneration
decimals
payable
balanceOf
swapTokensForETH
addLiquidity
IRouter (iface)
factory
WETH
addLiquidityETH
swapExactTokensForETHSupportingFeeOnTransferTokens
IBEP20Metadata (iface)
name
symbol
decimals
IBEP20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData

Summary

DecentralZone contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like stopping transactions. 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 5% buy, and sell 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