Cyberscope

Audit Report

Dont Buy This

April 2024

Network	BSC
Address	0xddf6796a39ca8c38a5ea279c248bbecd297262e5
Audited by	© cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

Diagnostics

Severity	Code	Description	Status
●	ZD	Zero Division	Unresolved
●	PAMAR	Pair Address Max Amount Restriction	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	IBFA	Inconsistent Buy Fees Application	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	TTLIB	Token Transfer Logic Inconsistent Behavior	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Contract Name	REFLECTIONS_TOKEN
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://bscscan.com/address/0xddf6796a39ca8c38a5ea279c248bbecd297262e5
Address	0xddf6796a39ca8c38a5ea279c248bbecd297262e5
Network	BSC
Symbol	DUMB
Decimals	3
Total Supply	333,333,333,333
Badge Eligibility	Yes

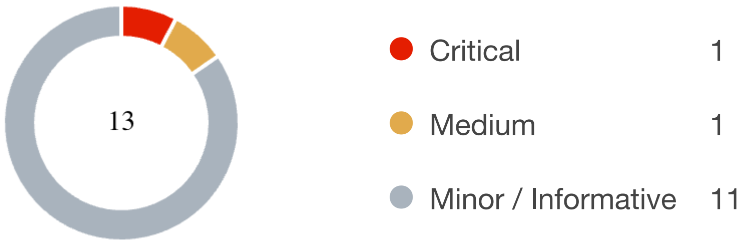
Audit Updates

Initial Audit	13 Apr 2024

Source Files

Filename	SHA256
REFLECTIONS_TOKEN.sol	4e300fb2eaf0c3189d88ddbe9c9e4b0af422fd4b989f77806a1df4ad73b9f9f5

Findings Breakdown



Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	1	0	0	0
●	Medium	1	0	0	0
●	Minor / Informative	11	0	0	0

ZD - Zero Division

Criticality	Critical
Location	REFLECTIONS_TOKEN.sol#L941
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.

```
function processFees(uint256 Tokens) private {

    // Lock Swap
    processingFees = true;

    // Totals for buy and sell fees
    uint8 _LiquidityTotal    = _fee__Buy_Liquidity +
_fee__Sell_Liquidity;
    uint8 _FeesTotal         = _SwapFeeTotal_Buy +
_SwapFeeTotal_Sell;

    // Calculate tokens for swap
    uint256 LP_Tokens        = Tokens * _LiquidityTotal / _FeesTotal
/ 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 the execution of the corresponding statements.

PAMAR - Pair Address Max Amount Restriction

Criticality	Medium
Location	BASEDSWAP.sol#L719
Status	Unresolved

Description

The contract owner has the authority to set the maximum wallet amount that the recipient can receipt. This percentage could be more than 1%. If the owner excludes the pair address from the `_isLimitExempt` then the sale will stop since the pair addresses usually accumulate a large amount of tokens that is much more than 0.5%. This behavior may lead the contract to become a honeypot.

```
function Wallet_Exempt_From_Limits(

        address Wallet_Address,
        bool true_or_false

        ) external onlyOwner {
        _isLimitExempt[Wallet_Address] = true_or_false;
}
...
if (!_isLimitExempt[to] && from != owner()) {
        uint256 heldTokens = balanceOf(to);
        require((heldTokens + amount) <= max_Hold, "WL"); // Over max
wallet limit
}
```

Recommendation

The team is adviced to conduct a thorough review of the criteria used to exclude addresses from the maximum wallet amount limit. Consider the implications of excluding pair addresses and assess whether alternative approaches or additional safeguards are necessary to address the unintended consequences described.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	REFLECTIONS_TOKEN.sol#L135
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.

```
_decimals
```

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.

IBFA - Inconsistent Buy Fees Application

Criticality	Minor / Informative
Location	REFLECTIONS_TOKEN.sol#L1030
Status	Unresolved

Description

The contract applies buy fees even when tokens are transferred from one address to another within the same transaction. This behavior contradicts the typical application of buy fees, which are traditionally intended to be imposed when tokens are purchased from an external source, such as an exchange, and added to the contract's liquidity pool. However, in this case, the contract mistakenly applies buy fees to all token transfers, regardless of whether they involve external purchases or internal address-to-address transfers.

```
if(_isExcludedFromFee[from] || _isExcludedFromFee[to] ||
(no_Fee_Transfers && !_isPair[to] && !_isPair[from])){
    takeFee = false;
} else {
    takeFee = true;
}
...
if(_isPair[recipient]){

    // Sell fees
    tSwapFeeTotal    = tAmount * _SwapFeeTotal_Sell     / 100;
    tReflect         = tAmount * _fee__Sell_Reflection / 100;
    tBurn            = tAmount * _fee__Sell_Burn       / 100;

} else {

    // Buy fees
    tSwapFeeTotal    = tAmount * _SwapFeeTotal_Buy      / 100;
    tReflect         = tAmount * _fee__Buy_Reflection  / 100;
    tBurn            = tAmount * _fee__Buy_Burn        / 100;

}
```

Recommendation

Conduct a thorough review of the fee application logic within the smart contract to identify and rectify the inconsistency in applying buy fees. Ensure that buy fees are only imposed when tokens are acquired from an external source and added to the liquidity pool, in accordance with standard buy fee practices.

MEE - Missing Events Emission

Criticality	Minor / Informative
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 addLiquidityPair(

    address Wallet_Address,
    bool true_or_false)

    external onlyOwner {

    _isPair[Wallet_Address] = true_or_false;
    _isLimitExempt[Wallet_Address] = true_or_false;

}
...
function burnFromTotalSupply(bool true_or_false) external onlyOwner
{
    burnFromSupply = true_or_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.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	REFLECTIONS_TOKEN.sol#L945
Status	Unresolved

Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
uint256 contract_BNB     = address(this).balance;
swapTokensForBNB(Swap_Tokens);
uint256 returned_BNB     = address(this).balance - contract_BNB;
```

Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	REFLECTIONS_TOKEN.sol#L901
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `max_Tran` 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.

```
if (contractTokens <= max_Tran) {

    processFees (contractTokens);

    } else {

    processFees (max_Tran);

}
```

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 exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

TTLIB - Token Transfer Logic Inconsistent Behavior

Criticality	Minor / Informative
Location	REFLECTIONS_TOKEN.sol#L1075
Status	Unresolved

Description

when the recipient address is designated as the burn address (Wallet_Burn) and the burnFromSupply flag is enabled, the contract deducts the transferred tokens directly from the total token supply (_tTotal) instead of transferring them to the burn address. This deviation from the standard token transfer behavior may lead to inconsistencies in the token supply and could potentially cause issues with decentralized applications (DApps) and fail format verification formulas.

```
if (recipient == Wallet_Burn && burnFromSupply) {

    _tTotal -= tTransferAmount;
    _rTotal -= rTransferAmount;

    } else {

        _rOwned[recipient] += rTransferAmount;

        if(_isExcludedFromRewards[recipient]){
            _tOwned[recipient] += tTransferAmount;
        }
    }
```

Recommendation

Review and standardize the token transfer logic to ensure consistency with ERC20 token standards and industry best practices. Tokens transferred to the burn address should follow the conventional approach of transferring tokens to the designated burn address, rather than directly deducting them from the total supply.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	REFLECTIONS_TOKEN.sol#L39,77,80,81,82,83,100,101,105,106,107,108,110,111,112,113,116,117,173,174,175,176,194,195,196,197,198,210,211,219,265,274,276,277,278,279,281,282,283,284,326,328,329,363,426,427,449,469,477,482,488,502,503,520,522,530,532,540,542,550,552,562,564,603,614,640,642,643,650,652,653,661,663,664,685,698,776,815,927,993
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.

```
function WETH() external pure returns (address);

...
```

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	REFLECTIONS_TOKEN.sol#L484
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.

```
swapTrigger = Transaction_Count + 1
```

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	REFLECTIONS_TOKEN.sol#L1036,1037,1038,1057,1058,1059
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 loss of prediction.

```
tBurn           = tAmount * _fee__Buy_Burn        / 100
uint256 rBurn           = tBurn          * RFI
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	REFLECTIONS_TOKEN.sol#L141
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
_owner = _OwnerWallet
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	REFLECTIONS_TOKEN.sol#L508
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.

```
IERC20(random_Token_Address).transfer(msg.sender, number_of_Tokens)
```

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.

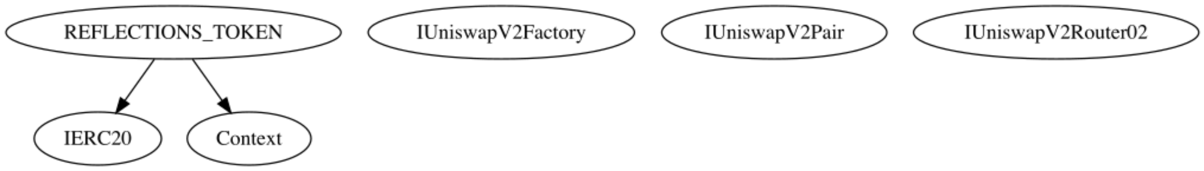
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
REFLECTIONS_TOKEN	Implementation	Context, IERC20		
		Public	✓	-
	Project_Information	External		-
	Set_Presale_CA	External	✓	onlyOwner
	Set_Fees	External	✓	onlyOwner
	Set_Wallet_Limits	External	✓	onlyOwner
	Open_Trade	External	✓	onlyOwner
	addLiquidityPair	External	✓	onlyOwner
	burnFromTotalSupply	External	✓	onlyOwner
	noFeeWalletTransfers	External	✓	onlyOwner
	swapAndLiquifySwitch	External	✓	onlyOwner
	swapTriggerCount	External	✓	onlyOwner
	swapAndLiquifyNow	External	✓	onlyOwner
	rescueTrappedTokens	External	✓	onlyOwner
	Update_Links_LP_Lock	External	✓	onlyOwner
	Update_Links_Telegram	External	✓	onlyOwner
	Update_Links_Website	External	✓	onlyOwner
	Update_Wallet_Liquidity	External	✓	onlyOwner
	Update_Wallet_Marketing	External	✓	onlyOwner

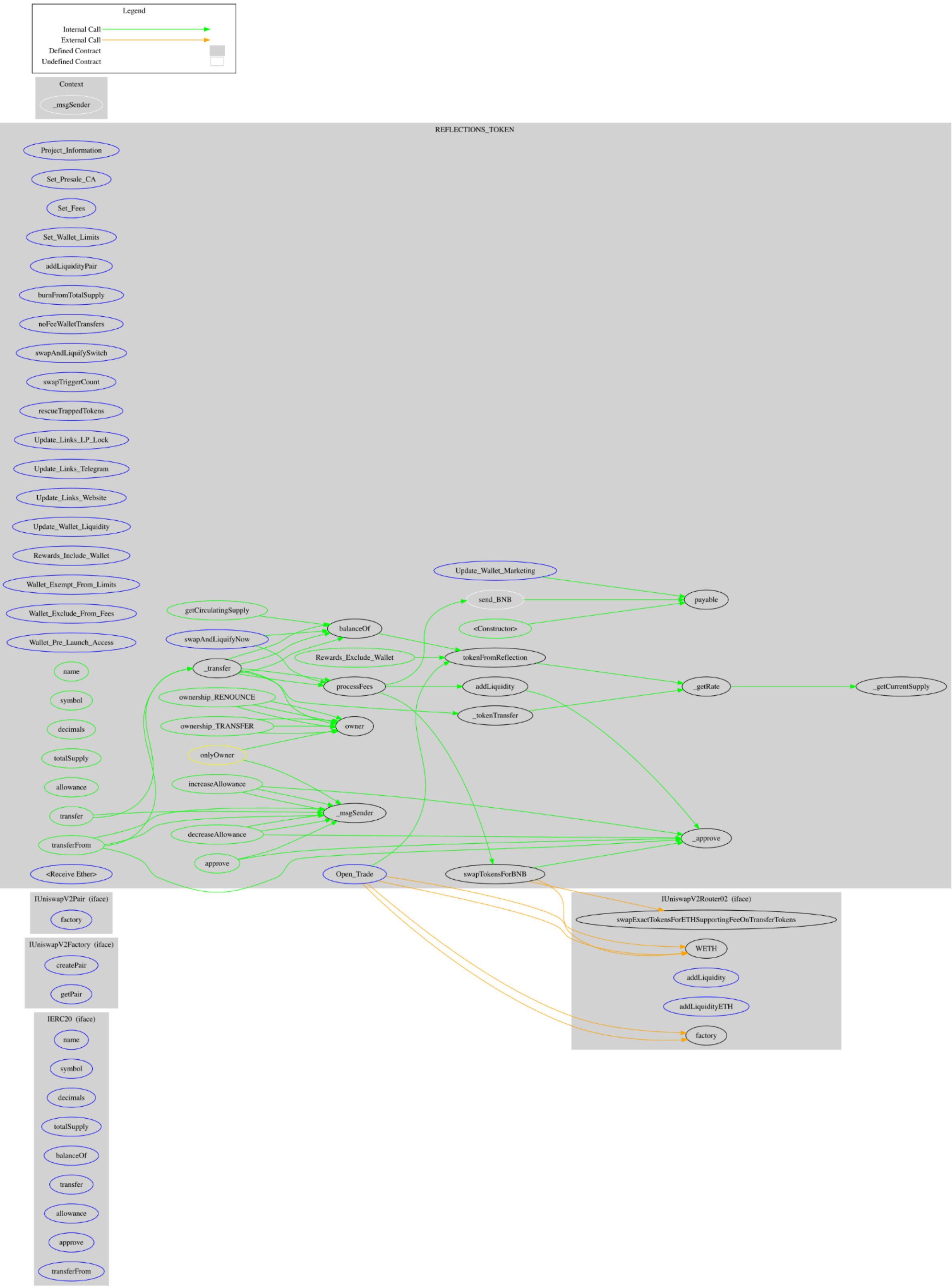
	Rewards_Exclude_Wallet	Public	✓	onlyOwner
	Rewards_Include_Wallet	External	✓	onlyOwner
	Wallet_Exempt_From_Limits	External	✓	onlyOwner
	Wallet_Exclude_From_Fees	External	✓	onlyOwner
	Wallet_Pre_Launch_Access	External	✓	onlyOwner
	ownership_RENOUNCE	Public	✓	onlyOwner
	ownership_TRANSFER	Public	✓	onlyOwner
	owner	Public		-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	approve	Public	✓	-
	_approve	Private	✓	
	tokenFromReflection	Internal		
	_getRate	Private		
	_getCurrentSupply	Private		
	transfer	Public	✓	-
	transferFrom	Public	✓	-

	send_BNB	Internal	✓	
	getCirculatingSupply	Public		-
	_transfer	Private	✓	
	processFees	Private	✓	
	swapTokensForBNB	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
		External	Payable	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Context
_msgSender
REFLECTIONS_TOKEN
Project_Information
Set_Presale_CA
Set_Fees
Set_Wallet_Limits
addLiquidityPair
burnFromTotalSupply
noFeeWalletTransfers
swapAndLiquifySwitch
swapTriggerCount
rescueTrappedTokens
Update_Links_LP_Lock
Update_Links_Telegram
Update_Links_Website
Update_Wallet_Liquidity
Rewards_Include_Wallet
Wallet_Exempt_From_Limits
Wallet_Exclude_From_Fees
Wallet_Pre_Launch_Access
name
symbol
decimals
totalSupply
allowance
transfer
transferFrom
<Receive Ether>
Update_Wallet_Marketing
send_BNB
payable
<Constructor>
getCirculatingSupply
balanceOf
swapAndLiquifyNow
Rewards_Exclude_Wallet
tokenFromReflection
_transfer
processFees
addLiquidity
_getRate
_getCurrentSupply
ownership_RENOUNCE
_tokenTransfer
ownership_TRANSFER
owner
onlyOwner
increaseAllowance
_msgSender
decreaseAllowance
_approve
approve
Open_Trade
swapTokensForBNB
IUniswapV2Pair (iface)
factory
IUniswapV2Factory (iface)
createPair
getPair
IERC20 (iface)
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IUniswapV2Router02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
addLiquidity
addLiquidityETH
factory

Summary

Dont Buy This contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Dont Buy This is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. 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