Cyberscope

Audit Report

JIMBOY

June 2023

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	Unresolved
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Medium	FRV	Fee Restoration Vulnerability	Unresolved
Minor	RVD	Redundant Variable Declaration	Unresolved
Minor	RED	Redundant Event Declaration	Unresolved
Minor	PVC	Price Volatility Concern	Unresolved
Minor	RSW	Redundant Storage Writes	Unresolved
Minor	RSML	Redundant SafeMath Library	Unresolved
Minor	L02	State Variables could be Declared Constant	Unresolved
Minor	L04	Conformance to Solidity Naming Conventions	Unresolved
Minor	L05	Unused State Variable	Unresolved
Minor	L07	Missing Events Arithmetic	Unresolved
Minor	L13	Divide before Multiply Operation	Unresolved
Minor	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	JIMBOY
Compiler Version	v0.8.9+commit.e5eed63a
Optimization	200 runs
Explorer	https://bscscan.com/address/0x0bcf961d6d1c13228d5e4c99cae48f5f8618ff6d
Address	0x0bcf961d6d1c13228d5e4c99cae48f5f8618ff6d
Network	BSC
Symbol	JIMBOY
Decimals	9
Total Supply	210,000,000,000,000

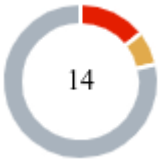
Audit Updates

Initial Audit	31 May 2023

Source Files

Filename	SHA256
JIMBOY.sol	70fae6cc75d4b52ba826e99f604fac49f2e8e0e7cfaf26880ba1c89020a34e5c

Findings Breakdown



Critical	2
Medium	1
Minor / Informative	11

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	1	0	0	0
Minor / Informative	11	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	JIMBOY.sol#L319,323,326
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users excluding the owner. The owner may take advantage of it by setting either the `tradingOpen` to false or the `_maxTxAmount` to zero.

```
if (!tradingOpen) {
    require(from == owner(), "TOKEN: This account cannot send tokens until
trading is enabled");
}
require(amount <= _maxTxAmount, "TOKEN: Max Transaction Limit");
```

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by setting either the `_maxWalletSize` to zero or the sell fees to a value greater than 100. As a result, the contract may operate as a honeypot.

```
if(to != uniswapV2Pair) {
    require(balanceOf(to) + amount < _maxWalletSize, "TOKEN: Balance
exceeds wallet size!");
}
...
function setFee(uint256 redisFeeOnBuy, uint256 redisFeeOnSell, uint256
taxFeeOnBuy, uint256 taxFeeOnSell) public onlyOwner {
    _redisFeeOnBuy = redisFeeOnBuy;
    _redisFeeOnSell = redisFeeOnSell;
    _taxFeeOnBuy = taxFeeOnBuy;
    _taxFeeOnSell = taxFeeOnSell;
}
```

Recommendation

The contract could embody a check for not allowing setting the `_maxTxAmount` and `_maxWalletSize` less than a reasonable amount. A suggested implementation could check that the minimum amount should be more than a fixed percentage of the total supply. 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.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	JIMBOY.sol#L526
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setFee` function with a high percentage value.

```
function setFee(uint256 redisFeeOnBuy, uint256 redisFeeOnSell, uint256
taxFeeOnBuy, uint256 taxFeeOnSell) public onlyOwner {
    _redisFeeOnBuy = redisFeeOnBuy;
    _redisFeeOnSell = redisFeeOnSell;
    _taxFeeOnBuy = taxFeeOnBuy;
    _taxFeeOnSell = taxFeeOnSell;
}
```

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. 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.

FRV - Fee Restoration Vulnerability

Criticality	Medium
Location	JIMBOY.sol#L281,291,410
Status	Unresolved

Description

The contract demonstrates a potential vulnerability upon removing and restoring the fees. This vulnerability can occur when both `_taxFee` and `_redisFee` have been set to zero. During a transaction, if the `takeFee` variable is false, then both `removeAllFee` and `restoreAllFee` function will be executed. The `removeAllFee` function is executed to temporarily remove the fees, ensuring the sender is not taxed during the transfer. However, the function prematurely returns without setting the variables that hold the previous fee values.

As a result, when the subsequent `restoreAllFee` function is called after the transfer, it restores the fees to their previous values. However, since the previous fee values were not properly set to zero, there is a risk that `_taxFee` and `_redisFee` will retain their non-zero values from before the fees were removed. This can lead to unintended consequences, potentially causing incorrect fee calculations or unexpected behavior within the contract.

```
function removeAllFee() private {
    if (_redisFee == 0 && _taxFee == 0) return;

    _previousredisFee = _redisFee;
    _previoustaxFee = _taxFee;

    _redisFee = 0;
    _taxFee = 0;
}

function restoreAllFee() private {
    _redisFee = _previousredisFee;
    _taxFee = _previoustaxFee;
}

function _tokenTransfer(
    address sender,
    address recipient,
    uint256 amount,
    bool takeFee
) private {
    if (!takeFee) removeAllFee();
    _transferStandard(sender, recipient, amount);
    if (!takeFee) restoreAllFee();
}
```

Recommendation

The team is advised to modify the `removeAllFee` function to ensure that the previous fee values are correctly set to zero, regardless of their initial values. A recommended approach would be to remove the early return when both fees are zero.

```
function removeAllFee() private {
    _previousredisFee = _redisFee;
    _previoustaxFee = _taxFee;

    _redisFee = 0;
    _taxFee = 0;
}
```

RVD - Redundant Variable Declaration

Criticality	Minor / Informative
Location	JIMBOY.sol#L153,171
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 declares certain variables that are not used in a meaningful way by the contract. As a result, these variables are redundant.

```
mapping(address => uint256) private _tOwned;
mapping(address => bool) public bots;
mapping (address => uint256) public _buyMap;
```

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.

RED - Redundant Event Declaration

Criticality	Minor / Informative
Location	JIMBOY.sol#L186
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 declares certain events that are not used by the contract. As a result, these events are redundant.

```
event MaxTxAmountUpdated(uint256 _maxTxAmount);
```

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.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	JIMBOY.sol#L534
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `_swapTokensAtAmount` 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 setMinSwapTokensThreshold(uint256 swapTokensAtAmount) public
onlyOwner {
    _swapTokensAtAmount = swapTokensAtAmount;
}
```

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.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	JIMBOY.sol#L539
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 without checking if their current state is the same as the one passed as an argument. As a result, the contract performs redundant storage writes.

```
function toggleSwap(bool _swapEnabled) public onlyOwner {
    swapEnabled = _swapEnabled;
}
```

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.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	JIMBOY.sol
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 to 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 gas consumption unnecessarily.

```
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	JIMBOY.sol#L44,172,173
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.

```
address private _previousOwner
address payable private _developmentAddress =
payable(0xc597291Fc1341Daf6Fdd56C0325697b7804145CC)
address payable private _marketingAddress =
payable(0x9268c469e4651cB1faf30275bC95d1F1E68B08D7)
```

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	JIMBOY.sol#L141,148,149,150,157,171,182,183,184,390,539
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);
string private constant _name = "JIMBOY"
string private constant _symbol = "JIMBOY"
uint8 private constant _decimals = 9
uint256 private constant _tTotal = 210_000_000_000_000 * 10**_decimals
mapping (address => uint256) public _buyMap
uint256 public _maxTxAmount =  1 * (_tTotal/100)
uint256 public _maxWalletSize = 1 * (_tTotal/100)
uint256 public _swapTokensAtAmount = 20 *(_tTotal/1000)
bool _tradingOpen
bool _swapEnabled
```

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.

L05 - Unused State Variable

Criticality	Minor / Informative
Location	JIMBOY.sol#L44,153
Status	Unresolved

Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
address private _previousOwner
mapping(address => uint256) private _tOwned
```

Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality, and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	JIMBOY.sol#L527,535,545,549
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.

```
_redisFeeOnBuy = redisFeeOnBuy
_swapTokensAtAmount = swapTokensAtAmount
_maxTxAmount = maxTxAmount
_maxWalletSize = maxWalletSize
```

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	JIMBOY.sol#L182,183,184
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.

```
uint256 public _swapTokensAtAmount = 20 *(_tTotal/1000)
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	JIMBOY.sol#L9
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 ^0.8.9;
```

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.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
SafeMath	Library			

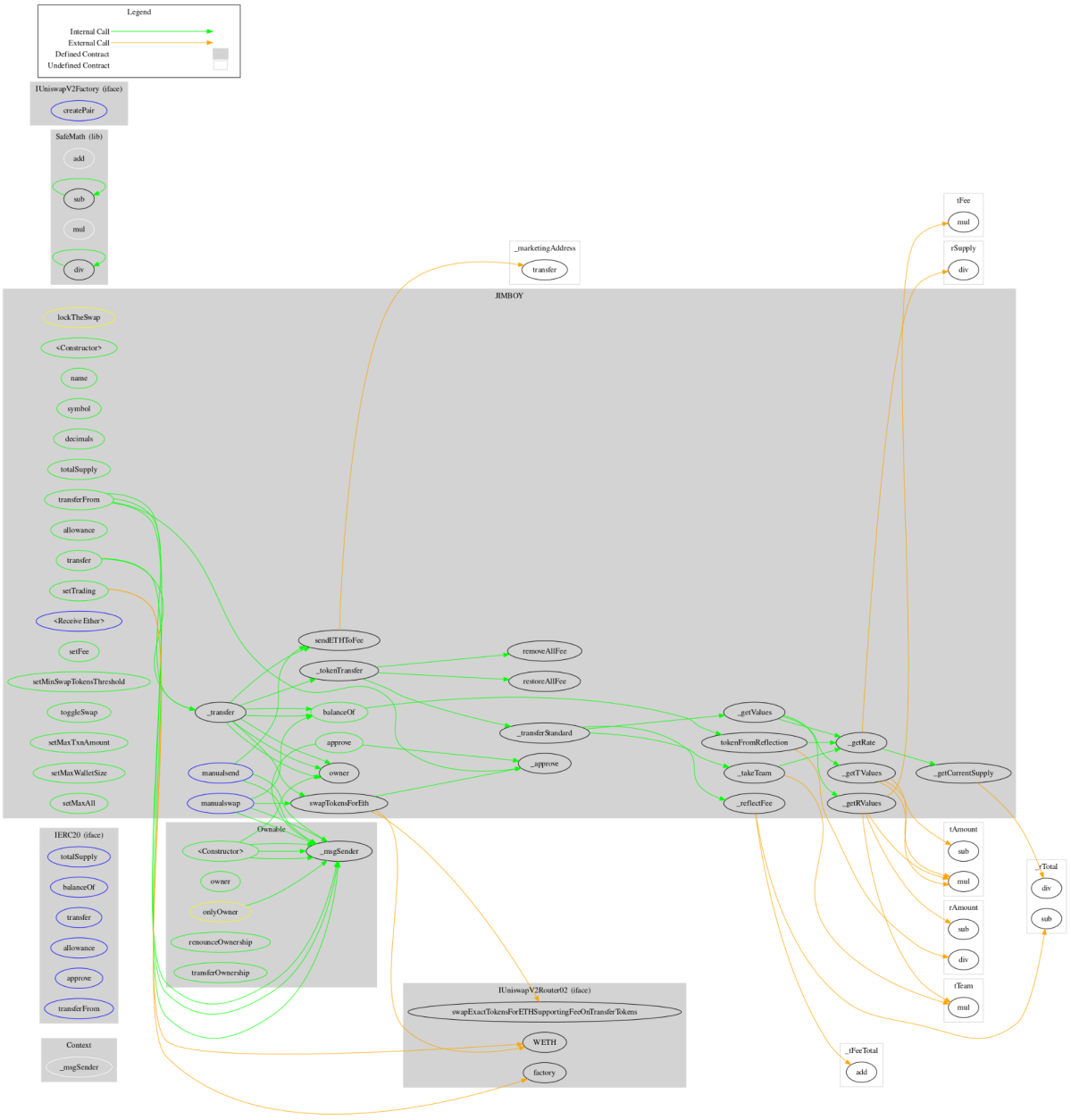
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
IUniswapV2Factory	Interface			
	createPair	External	✓	-
IUniswapV2Router02	Interface			
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	factory	External		-
	WETH	External		-
JIMBOY	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-

	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	tokenFromReflection	Private		
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	
	_approve	Private	✓	
	_transfer	Private	✓	
	swapTokensForEth	Private	✓	lockTheSwap
	sendETHToFee	Private	✓	
	setTrading	Public	✓	onlyOwner
	manualswap	External	✓	-
	manualsend	External	✓	-
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_takeTeam	Private	✓	
	_reflectFee	Private	✓	
		External	Payable	-
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		

	_getCurrentSupply	Private		
	setFee	Public	✓	onlyOwner
	setMinSwapTokensThreshold	Public	✓	onlyOwner
	toggleSwap	Public	✓	onlyOwner
	setMaxTxnAmount	Public	✓	onlyOwner
	setMaxWalletSize	Public	✓	onlyOwner
	setMaxAll	Public	✓	onlyOwner

Inheritance Graph

Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Factory (iface)
createPair
SafeMath (lib)
add
sub
mul
div
_marketingAddress
transfer
JIMBOY
lockTheSwap
<Constructor>
name
symbol
decimals
totalSupply
transferFrom
allowance
transfer
setTrading
<Receive Ether>
setFee
setMinSwapTokensThreshold
toggleSwap
setMaxTxnAmount
setMaxWalletSize
setMaxAll
sendETHToFee
_tokenTransfer
_transfer
balanceOf
approve
owner
manualsend
manualswap
swapTokensForEth
removeAllFee
restoreAllFee
_transferStandard
_approve
_getValues
tokenFromReflection
_takeTeam
_reflectFee
_getRate
_getTValues
_getRValues
_getCurrentSupply
tFee
mul
rSupply
div
tAmount
sub
mul
rAmount
sub
div
rTeam
mul
_rTotal
div
sub
_tFeeTotal
add
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
Ownable
<Constructor>
owner
onlyOwner
renounceOwnership
transferOwnership
_msgSender
IUniswapV2Router02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory

Summary

JIMBOY 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 and manipulating the fees. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. 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

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