Cyberscope

Audit Report

Mavrodi

December 2023

Network	ETH
Address	0x460440c8deb7680d32a87dfb2d6392c2c83bfa09
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

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	MEE	Missing Events Emission	Unresolved
●	RED	Redudant Event Declaration	Unresolved
●	AOI	Arithmetic Operations Inconsistency	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	CR	Code Repetition	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	MMMem
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://etherscan.io/address/0x460440c8deb7680d32a87dfb2d6392c2c83bfa09
Address	0x460440c8deb7680d32a87dfb2d6392c2c83bfa09
Network	ETH
Symbol	MMMem
Decimals	18
Total Supply	91,000,000

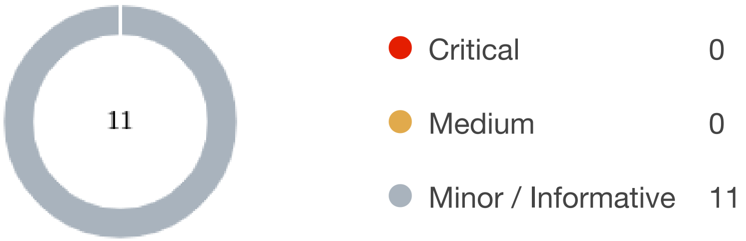
Audit Updates

Initial Audit	14 Dec 2023

Source Files

Filename	SHA256
MMMem.sol	5d5be445ad6858b946dd7e9b507e9b56094d64058d7e2431fb335731ab56e147

Findings Breakdown



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

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	MMMem.sol#L742,751,796,804,815
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 excludeFromReward(address account) public onlyOwner {
        // require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D,
'We can not exclude Uniswap router.');
        require(!_isExcluded[account], "Account is already excluded");
        if (_rOwned[account] > 0) {
            _tOwned[account] = tokenFromReflection(_rOwned[account]);
        }
        _isExcluded[account] = true;
        _excluded.push(account);
    }

    function includeInReward(address account) external onlyOwner {
        ...
    }

    function setTaxFeePercent(uint256 taxFeeBps) external onlyOwner {
        _taxFee = taxFeeBps;
        require(
            _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
            "Total fee is over 25%"
        );
    }

    function setTaxToTheCashie(uint256 taxToTheCashier)
        external
        onlyOwner
    {
        _taxToTheCashier = taxToTheCashier;
        require(
            _taxFee + _taxToTheCashier + _supportForThePeople <= MAX_FEE,
            "Total fee is over 25%"
        );
    }

    function setSupportForThePeoplePercent(uint256 supportForThePeople)
external onlyOwner {
        _supportForThePeople = supportForThePeople;
        require(
            _taxFee + _taxToTheCashier + _supportForThePeople <= MAX_FEE,
            "Total fee is over 25%"
        );
    }
```

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.

RED - Redudant Event Declaration

Criticality	Minor / Informative
Location	MMMem.sol#L527
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 event `MinTokensBeforeSwapUpdated` is declared and not being used in the contract. As a result, it is redundant.

```
event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

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.

AOI - Arithmetic Operations Inconsistency

Criticality	Minor / Informative
Location	MMMem.sol#L810,837
Status	Unresolved

Description

The contract uses both the SafeMath library and native arithmetic operations. The SafeMath library is commonly used to mitigate vulnerabilities related to integer overflow and underflow issues. However, it was observed that the contract also employs native arithmetic operators (such as +, -, *, /) in certain sections of the code.

The combination of SafeMath library and native arithmetic operations can introduce inconsistencies and undermine the intended safety measures. This discrepancy creates an inconsistency in the contract's arithmetic operations, increasing the risk of unintended consequences such as inconsistency in error handling, or unexpected behavior.

```
_taxFee + _taxToTheCashier + _supportForThePeople <= MAX_FEE,
...
_tFeeTotal = _tFeeTotal.add(tFee);
```

Recommendation

To address this finding and ensure consistency in arithmetic operations, it is recommended to standardize the usage of arithmetic operations throughout the contract. The contract should be modified to either exclusively use SafeMath library functions or entirely rely on native arithmetic operations, depending on the specific requirements and design considerations. This consistency will help maintain the contract's integrity and mitigate potential vulnerabilities arising from inconsistent arithmetic operations.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	MMMem.sol#L1028
Status	Unresolved

Description

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

```
       uint256 contractTokenBalance = balanceOf(address(this));

        bool overMinTokenBalance = contractTokenBalance >=
            numTokensSellToAddToLiquidity;
        if (
            overMinTokenBalance &&
            !inSwapAndLiquify &&
            from != uniswapV2Pair &&
            swapAndLiquifyEnabled
        ) {
            contractTokenBalance = numTokensSellToAddToLiquidity;
            //add liquidity
            swapAndLiquify(contractTokenBalance);
        }
```

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.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	MMMem.sol#L788
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
   function excludeFromFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = true;
    }

    function includeInFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = false;
    }
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

CR - Code Repetition

Criticality	Minor / Informative
Location	MMMem.sol#L537
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
   constructor(
        string memory name_,
        string memory symbol_,
        uint256 totalSupply_,
        address router_,
        address charityAddress_,
        uint16 taxFee_,
        uint16 taxToTheCashier_,
        uint16 supportForThePeople_
    ) {
        ...
        require(
            taxFee_ + taxToTheCashier_ + supportForThePeople_
<= MAX_FEE,
            "Total fee is over 25%"
        );

        _name = name_;
        _symbol = symbol_;
        _decimals = 18;

        _tTotalMavro = totalSupply_;
        require(
            taxFee_ + taxToTheCashier_ + supportForThePeople_
<= MAX_FEE,
            "Total fee is over 25%"
        );

        _name = name_;
        _symbol = symbol_;
        _decimals = 18;

        ...
        _rTotalMavro = (MAX - (MAX % _tTotalMavro));

        ...
        _rTotalMavro = (MAX - (MAX % _tTotalMavro));
        ...
        }
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	MMMem.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.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	MMMem.sol#L560,562,571,577,583
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
_tTotalMavro
_charityAddress
swapAndLiquifyEnabled
uniswapV2Pair
```

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	MMMem.sol#L166,431,505,508,511,516,819,958,962,970
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 Mavrodi() public view virtual returns (address) {
        return _owner;
    }
function WETH() external pure returns (address);
uint256 public _taxFee
uint256 public _taxToTheCashier
uint256 public _supportForThePeople
address public _charityAddress
uint256 _amount
```

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	MMMem.sol#L793,804,812
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.

```
_taxFee = taxFee
_taxToTheCashier = taxToTheCashier
_supportForThePeople = supportForThePeople
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	MMMem.sol#L2
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.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	Mavrodi	Public		-
	powerToThePeople	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	

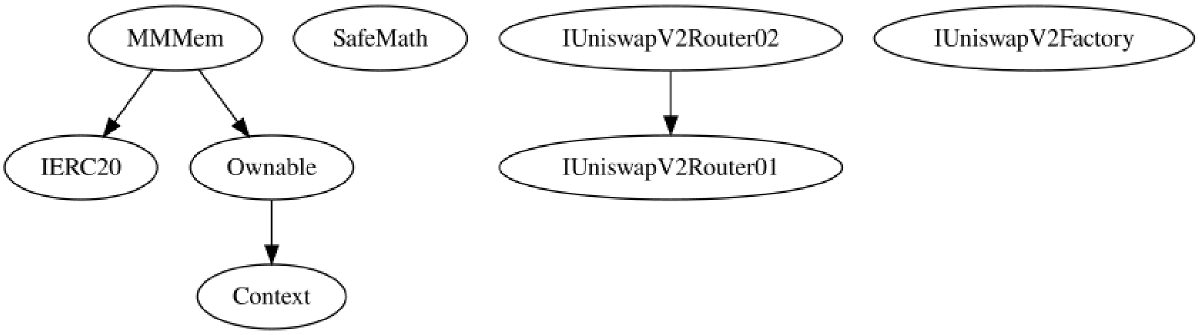
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-

IUniswapV2Factory	Interface			
	createPair	External	✓	-
MMMem	Implementation	IERC20, Ownable		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	isExcludedFromReward	Public		-
	totalFees	Public		-
	deliver	Public	✓	-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-
	excludeFromReward	Public	✓	onlyOwner

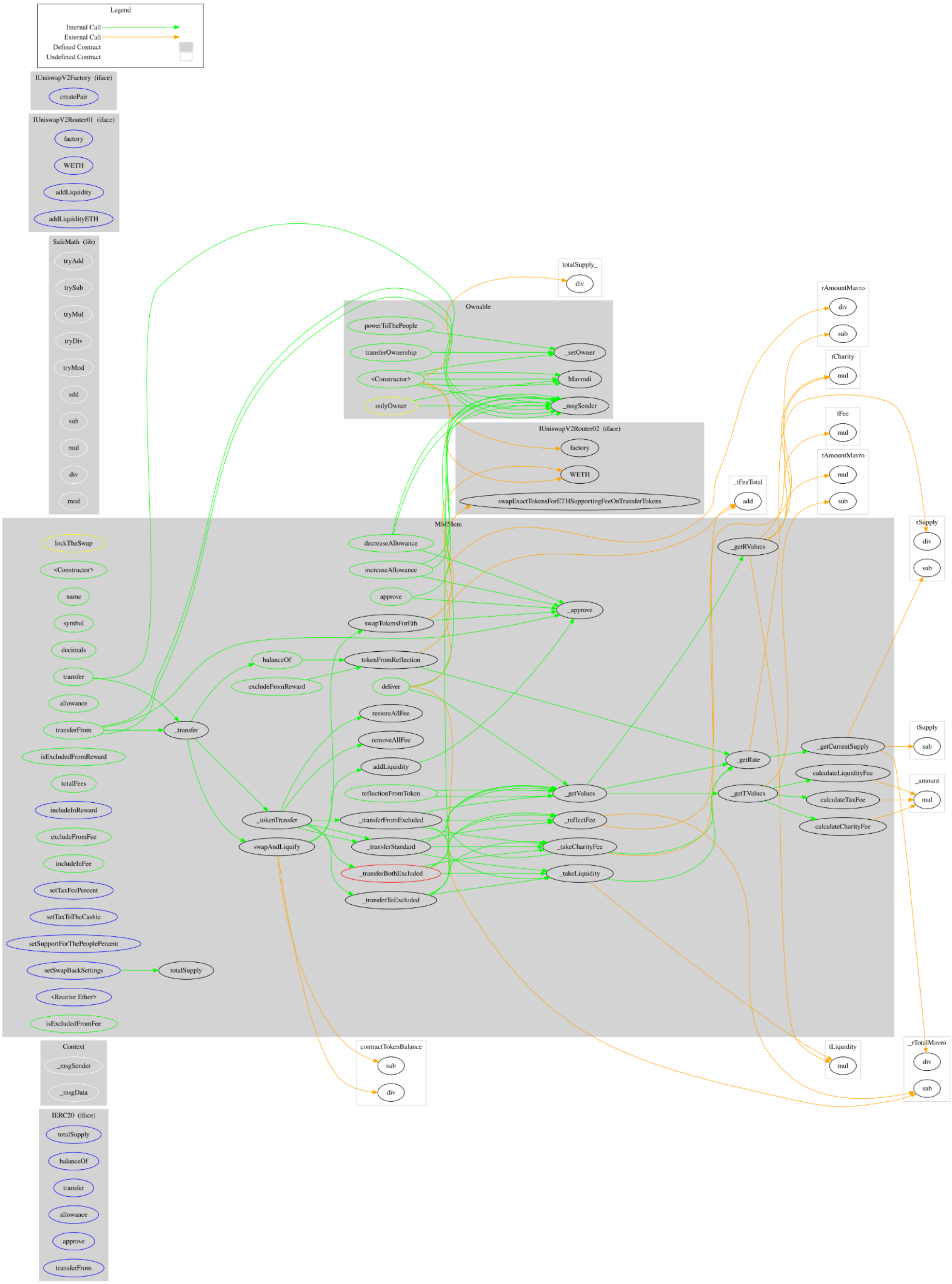
	includeInReward	External	✓	onlyOwner
	_transferBothExcluded	Private	✓	
	excludeFromFee	Public	✓	onlyOwner
	includeInFee	Public	✓	onlyOwner
	setTaxFeePercent	External	✓	onlyOwner
	setTaxToTheCashie	External	✓	onlyOwner
	setSupportForThePeoplePercent	External	✓	onlyOwner
	setSwapBackSettings	External	✓	onlyOwner
		External	Payable	-
	_reflectFee	Private	✓	
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidity	Private	✓	
	_takeCharityFee	Private	✓	
	calculateTaxFee	Private		
	calculateLiquidityFee	Private		
	calculateCharityFee	Private		
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	
	isExcludedFromFee	Public		-

	_approve	Private	✓	
	_transfer	Private	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Factory (iface)
createPair
IUniswapV2Router01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
totalSupply_
div
Ownable
powerToThePeople
transferOwnership
<Constructor>
onlyOwner
_setOwner
Mavrodi
_msgSender
IUniswapV2Router02 (iface)
factory
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
rAmountMavro
div
sub
tCharity
mul
tFee
mul
tAmountMavro
mul
sub
_tFeeTotal
add
rSupply
div
sub
MMMem
lockTheSwap
<Constructor>
name
symbol
decimals
transfer
allowance
transferFrom
isExcludedFromReward
totalFees
includeInReward
excludeFromFee
includeInFee
setTaxFeePercent
setTaxToTheCashie
setSupportForThePeoplePercent
setSwapBackSettings
<Receive Ether>
isExcludedFromFee
totalSupply
_transfer
decreaseAllowance
increaseAllowance
approve
swapTokensForEth
balanceOf
excludeFromReward
tokenFromReflection
deliver
restoreAllFee
removeAllFee
addLiquidity
reflectionFromToken
_tokenTransfer
swapAndLiquify
_transferFromExcluded
_transferStandard
_transferBothExcluded
_transferToExcluded
_approve
_getRValues
_getValues
_reflectFee
_takeCharityFee
_takeLiquidity
_getRate
_getTValues
_getCurrentSupply
calculateLiquidityFee
calculateTaxFee
calculateCharityFee
tSupply
sub
_amount
mul
Context
_msgSender
_msgData
contractTokenBalance
sub
div
tLiquidity
mul
_rTotalMavro
div
sub
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom

Summary

Mavrodi contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Mavrodi 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 25% 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