Cyberscope

Audit Report

Maven

April 2023

Network	ETH
Address	0xCB245dF5BE6C263ce770951dC5650d278E6051da
Audited by	© cyberscope

Table of Contents

Review

Contract Name	Maven
Compiler Version	v0.8.15+commit.e14f2714
Optimization	5000 runs
Explorer	https://etherscan.io/address/0xcb245df5be6c263ce770951dc5650d278e6051da
Address	0xcb245df5be6c263ce770951dc5650d278e6051da
Network	ETH
Symbol	MVN
Decimals	18
Total Supply	1,000,000,000

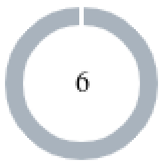
Audit Updates

Initial Audit	12 Apr 2023

Source Files

Filename	SHA256
Maven.sol	5d1c3daf525560d394b19f60bc62cf5ba1f629517140757fe83491941d11faa3

Findings Breakdown



●	Critical	0
●	Medium	0
●	Minor / Informative	6

Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	0	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	6	0	0	0

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	PVC	Price Volatility Concern	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L05	Unused State Variable	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L12	Using Variables before Declaration	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	Maven.sol#L456
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapAmount` 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 setSwapSettings(uint256 thresholdPercent, uint256
thresholdDivisor, uint256 amountPercent, uint256 amountDivisor) external
onlyOwner {
    swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
    swapAmount = (_tTotal * amountPercent) / amountDivisor;
    require(swapThreshold <= swapAmount, "Threshold cannot be above
amount.");
}
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Maven.sol#L33,122,124,125,126,127,142,149,155,156,157,158,159,170,198,400
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);
uint256 constant private startingSupply = 1_000_000_000
string constant private _name = "Maven"
string constant private _symbol = "MVN"
uint8 constant private _decimals = 18
uint256 constant private _tTotal = startingSupply * 10**_decimals

Fees public _taxRates = Fees({
        buyFee: 400,
        sellFee: 700,
        sellHighTxFee: 1000,
        transferFee: 1000
    })

...
```

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	Maven.sol#L108
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.

```
mapping (address => uint256) private _rOwned
```

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	Maven.sol#L439,445,457,464,475,494,499
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.

```
_maxTxAmountBuy = (_tTotal * percentBuy) / divisorBuy
_maxWalletSize = (_tTotal * percent) / divisor
swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor
piSwapPercent = priceImpactSwapPercent
_highTaxSize = (_tTotal * percent) / divisor
burnXLimit = (_tTotal * percent) / divisor
burnXTimer = timeInMinutes * 1 minutes
```

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.

L12 - Using Variables before Declaration

Criticality	Minor / Informative
Location	Maven.sol#L707
Status	Unresolved

Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or if the variable has been declared in a different scope. It is not a good practice to use a local variable before it has been declared.

```
bool check
```

Recommendation

By declaring local variables before using them, the contract ensures that it operates correctly. It's important to be aware of this rule when working with local variables, as using a variable before it has been declared can lead to unexpected behavior and can be difficult to debug.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	Maven.sol#L706,707
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.

```
bool checked
bool check
```

Recommendation

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

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IFactoryV2	Interface			
	getPair	External		-
	createPair	External	✓	-
IV2Pair	Interface			
	factory	External		-

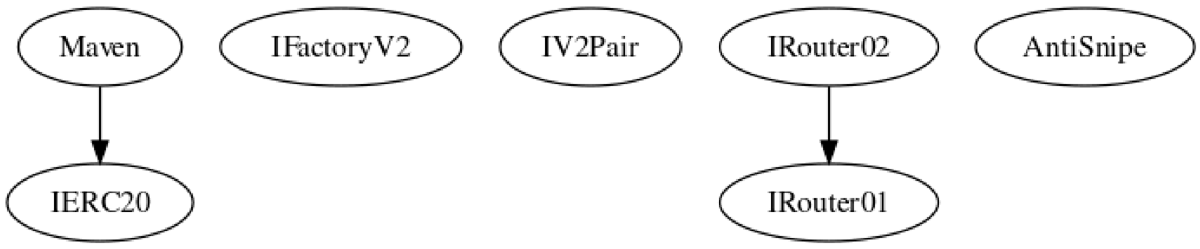
	getReserves	External		-
	sync	External	✓	-
IRouter01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	addLiquidity	External	✓	-
	swapExactETHForTokens	External	Payable	-
	getAmountsOut	External		-
	getAmountsIn	External		-
IRouter02	Interface	IRouter01		
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokens	External	✓	-
AntiSnipe	Interface			
	checkUser	External	✓	-
	setLaunch	External	✓	-
	setLpPair	External	✓	-
	setProtections	External	✓	-

	removeSniper	External	✓	-
	removeBlacklisted	External	✓	-
	isBlacklisted	External		-
	setBlacklistEnabled	External	✓	-
	setBlacklistEnabledMultiple	External	✓	-
	setBlockTXDelay	External	✓	-
Maven	Implementation	IERC20		
		Public	Payable	-
		External	Payable	-
	transferOwner	External	✓	onlyOwner
	renounceOwnership	External	✓	onlyOwner
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	allowance	External		-
	balanceOf	Public		-
	transfer	Public	✓	-
	approve	External	✓	-
	_approve	Internal	✓	
	approveContractContingency	External	✓	onlyOwner

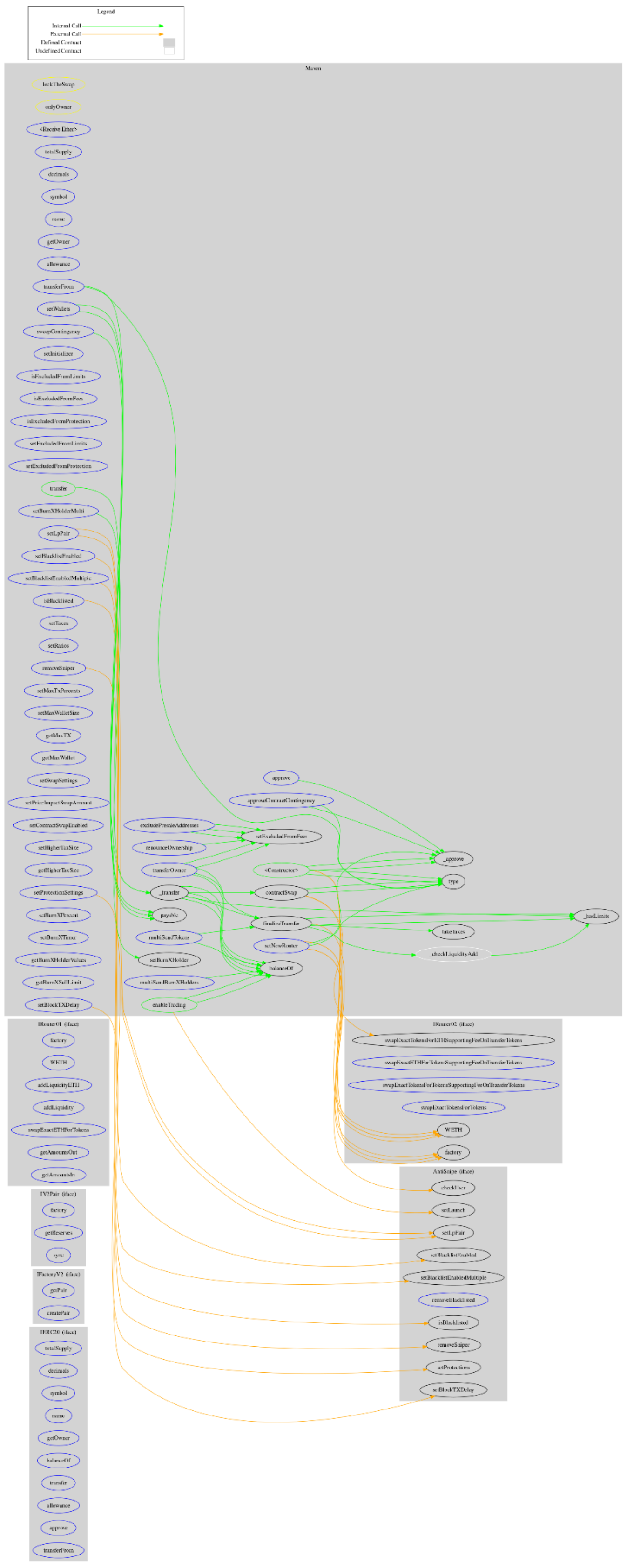
	transferFrom	External	✓	-
	setNewRouter	External	✓	onlyOwner
	setLpPair	External	✓	onlyOwner
	setInitializer	External	✓	onlyOwner
	isExcludedFromLimits	External		-
	isExcludedFromFees	External		-
	isExcludedFromProtection	External		-
	setExcludedFromLimits	External	✓	onlyOwner
	setExcludedFromFees	Public	✓	onlyOwner
	setExcludedFromProtection	External	✓	onlyOwner
	setBlacklistEnabled	External	✓	onlyOwner
	setBlacklistEnabledMultiple	External	✓	onlyOwner
	isBlacklisted	External		-
	removeSniper	External	✓	onlyOwner
	setProtectionSettings	External	✓	onlyOwner
	setBlockTXDelay	External	✓	onlyOwner
	setTaxes	External	✓	onlyOwner
	setRatios	External	✓	onlyOwner
	setWallets	External	✓	onlyOwner
	setMaxTxPercents	External	✓	onlyOwner
	setMaxWalletSize	External	✓	onlyOwner
	getMaxTX	External		-
	getMaxWallet	External		-

	setSwapSettings	External	✓	onlyOwner
	setPriceImpactSwapAmount	External	✓	onlyOwner
	setContractSwapEnabled	External	✓	onlyOwner
	setHigherTaxSize	External	✓	onlyOwner
	getHigherTaxSize	External		-
	setBurnXHolder	Public	✓	onlyOwner
	setBurnXHolderMulti	External	✓	onlyOwner
	setBurnXPercent	External	✓	onlyOwner
	setBurnXTimer	External	✓	onlyOwner
	getBurnXHolderValues	External		-
	getBurnXSellLimit	External		-
	excludePresaleAddresses	External	✓	onlyOwner
	_hasLimits	Internal		
	_transfer	Internal	✓	
	contractSwap	Internal	✓	lockTheSwap
	_checkLiquidityAdd	Internal	✓	
	enableTrading	Public	✓	onlyOwner
	sweepContingency	External	✓	onlyOwner
	multiSendTokens	External	✓	onlyOwner
	multiSendBurnXHolders	External	✓	onlyOwner
	finalizeTransfer	Internal	✓	
	takeTaxes	Internal	✓	

Inheritance Graph



Flow Graph



Summary

Maven contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. Maven 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% buy, sell, and transfer fees.

Disclaimer

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