CMC Coin Smart Contract Audit Report - November 2021

Submitted By

Antier Solutions

Contents

Disclaimer

This is a limited audit report based on our analysis of the CMC Coin Smart Contract. It covers industry best practices as of the date of this report, concerning: smart contract best coding practices, cybersecurity vulnerabilities, issues in the framework and algorithms based on white paper, code, the details of which are set out in this report, (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

Smart contracts are deployed and executed on the blockchain. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks.

You are advised to read the full report to get a full view of our analysis. While we did our best in producing this report, it is important to note that you should not rely on this report, and cannot claim against us, based on what it says or does not say, or how we produced it, and you need to conduct your independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy all copies of this report downloaded and/or printed by you.

The report is provided "as is," without any condition, warranty, or other terms of any kind except as set out in this disclaimer. TAS hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose, and the use of reasonable care and skill) which, but for this clause, might affect the report.

Executive Summary

CMC Coin commissioned Antier Solutions to perform an end-to-end source code review of their Solidity Smart Contract. Team Antier Solutions (referred to as TAS throughout the report) performed the audit from 3rd to 17th November 2021.

The following report discusses severity issues and their scope of rectification through change recommendations. It also highlights activities that are successfully executed and others that need total reworking (if any).

The report emphasizes best practices in coding and the security vulnerabilities if any.

The information in this report should be used to understand the overall code quality, security, and correctness of the Smart Contract. The analysis is static and entirely limited to the Smart Contract code.

In the audit, we reviewed the Smart Contract's code that implements the token mechanism.

Scope

We performed an independent technical audit to identify Smart Contract uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the contract from frauds.

The audit was performed semi-manually. We analysed the Smart Contract code line-by-line and used an automation tool to report any suspicious code.

We considered the following standards for the Smart Contract code review:

- ERC20 [Token Contract best practices]
- Router02 (Uniswap) [Token swapping contract best practices]
- Dividend-Paying Token Contract Standards.

We used the following tools to perform automated tests:

- Manual Testing tool :
  Hardhat
- Framework :
  Remix Ethereum
- Automation tools:
  - Slither
  - Surya
  - Mythril

Audit Goals

The focus of the audit was to verify that the Smart Contract system was secure, resilient, and worked according to the specifications provided to the Auditing team.

TAS grouped the audit activities in the following three categories:

- **Security**
  Identifying security-related issues within each contract and the system of contracts

- **Architecture**
  Evaluation of the system architecture against smart contract conventions and general software best practices

- **Code Correctness and Quality**
  A full review of the contract source code. The primary areas of focus include:
  - Correctness
  - Readability
  - Sections of code with high complexity
  - Quantity and quality of test coverage
- **Functional Testing**
  Type of software testing that validates the software system against the functional requirements/specifications.

Recommendations

The CMC Coin development team demonstrated high technical capabilities, both in the design of the architecture and implementation of the Smart Contract. Overall, the code includes effective use of abstraction, separation of concerns, and modularity.

Compiler versions

CMC Coin should use the latest compiler version. Here, the compiler version is 0.6.2

Code security

TAS performed a static analysis of the code to identify possible loopholes. This verified whether the contract adhered to the Solidity best practices.

Implementation instructions

Include Unit Test cases for better understanding and testing of Gas limits. All the implementation instructions should be written in the README file.

Contracts Function Description

Contract Name	Functions	Visibility	Mutability	Modifiers
CryptoMarketingCompany	<Constructor>	Public	modifies state	ERC20
	<Receive Ether>	External	payable	
	updateWeeklyDividendTracker	Public	modifies state	onlyOwner
	updateMontlyDividendTracker	Public	modifies state	onlyOwner
	updateUniswapV2Router	Public	modifies state	onlyOwner
	excludeFromFees	Public	modifies state	onlyOwner
	excludeMultipleAccountsFromFees	Public	modifies state	onlyOwner
	setMarketingWallet	External	modifies state	onlyOwner
	switchRewardTokenAddress	External	modifies state	onlyOwner
	setRewardTokenWeeklyRewardsFees	External	modifies state	onlyOwner
	setRewardTokenMonthlyRewardsFees	External	modifies state	onlyOwner
	setLiquidityFees	External	modifies state	onlyOwner
	setMarketingFees	External	modifies state	onlyOwner
	setAutomatedMarketMakerPair	Public	modifies state	onlyOwner
	blacklistAddress	External	modifies state	
	_setAutomatedMarketMakerPair	Private	modifies state	
	updateGasForProcessing	Public	modifies state	
	updateWeeklyClaimWait	External	modifies state	onlyOwner
	updateMonthlyClaimWait	External	modifies state	onlyOwner
	getWeeklyClaimWait	External	modifies state	
	getMonthlyClaimWait	External		
	getWeeklyTotalDividendsDistributed	External		
	getMonthlyTotalDividendsDistributed	External		
	isExcludedFromFees	Public		
	withdrawableDividendsOf	Public		
	dividendTokenBalancesOf	Public		
	excludeFromDividends	External	modifies state	onlyOwner
	getWeeklyAccountDividendsInfo	External		

	getMonthlyAccountDividendsInfo getWeeklyAccountDividendsInfoAtIndex getMontlyAccountDividendsInfoAtIndex processDividendTrackers claim getLastProcessedIndex getNumberOfWeeklyDividendTokenHolders getNumberOfMonthlyDividendTokenHolders getTransferAmounts _transfer swapAndSendToFee swapAndLiquify swapTokensForEth swapTokensForRewardToken addLiquidity swapAndSendDividends	External External External External External External External External Private Internal Private Private Private Private Private Private	 modifies state modifies state modifies state modifies state modifies state modifies state modifies state modifies state modifies state	
CMCDividendTracker	<Constructor> DividendPayingToken _transfer | | withdrawDividend excludeFromDividends updateClaimWait getLastProcessedIndex getNumberOfTokenHolders getAccount getAccountAtIndex canAutoClaim setBalance process processAccount	Public Internal Public External External External External Public Public Private External Public Public	modifies state modifies state modifies state modifies state modifies state	 onlyOwner onlyOwner onlyOwner onlyOwner
DividendPayingToken	<Constructor> distributeRewardTokenDividends withdrawDividend _withdrawDividendOfUser dividendOf withdrawableDividendOf withdrawnDividendOf accumulativeDividendOf _transfer _mint _burn _setBalance switchRewardTokenAddress	Public Public Public Internal Public Public Public Public Internal Internal Internal Internal External	modifies state modifies state modifies state modifies state modifies state modifies state modifies state modifies state modifies state	ERC20 onlyOwner

Severity Level References

The following severity levels will describe the degree of every issue:

High severity issues

The issue puts the majority of, or large numbers of, users' sensitive information at risk, or are reasonably likely to lead to a catastrophic impact on the client's reputation or serious financial implications for the client and users.

Medium severity issues

The issue puts a subset of individual users' sensitive information at risk; exploitation would be detrimental to the client's reputation or is reasonably likely to lead to moderate financial impact.

Low severity issues

The risk is relatively low and could not be exploited regularly, or it's a risk not indicated as important or impactful by the client because of the client's business circumstances.

Informational

The issue does not pose an immediate threat to continued operation or usage but is relevant for security best practices, software engineering best practices, or defensive redundancy.

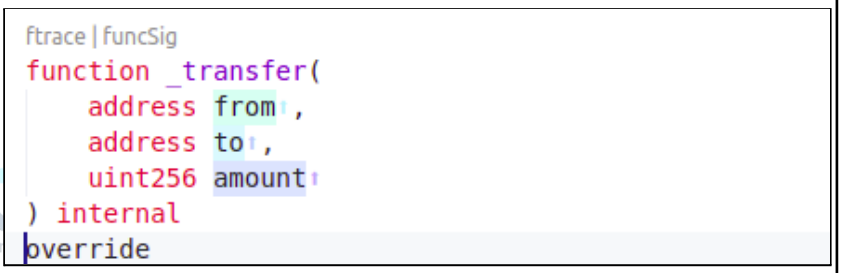
Optimization issue

The issue does not pose an immediate threat to continued operation or usage but is relevant for code optimization and gas efficiency best practices.

Number of vulnerabilities per severity

High	Medium	Low	Informational
0	7	1	9

Medium Severity Vulnerabilities

Severity	Medium
Contract	**CryptoMarketingCompany.sol**
Description	**There are multiple scenarios where transfer of tokens failed** • When the contract's CMC token balance reaches a threshold amount for swapping and liquifying, it failed. • Marketing, weekly dividend and monthly dividend values aren't updated after they are distributed to the respective addresses.
Code Snippet	
Recommendation	Subtract the transferred token amounts from monthly, weekly, marketing tokens after the transfer.
Status	**Fixed**

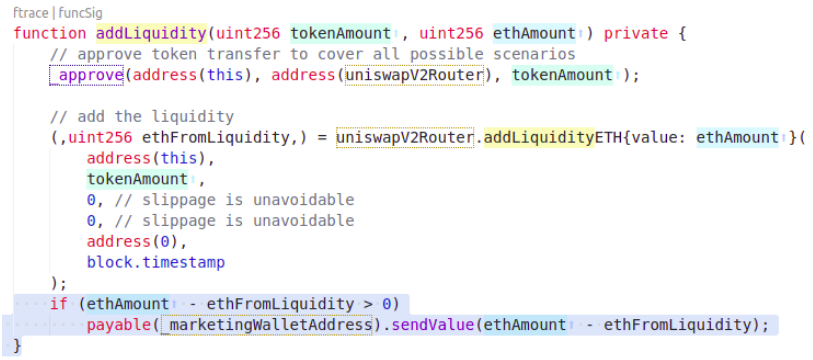
Severity	Medium
Contract	**CryptoMarketingCompany.sol**
Description	**Unchecked transfer.** The return value of an external transfer/transferFrom call is not checked

Code Snippet	
Recommendation	Use SafeERC20, or ensure that the transfer/transferFrom return value is checked.
Status	**Fixed**(Function removed)

Severity	**Medium**
Contract	**CryptoMarketingCompany.sol**
Description	**Lack of balance check.**

Function	ftrace | funcSig
	function swapAndSendToFeeInRewardToken(uint256 tokens, address feeAddress) private {
	uint256 initialRewardTokenBalance =
	IERC20(_rewardTokenAddress).balanceOf(address(this));
	swapTokensForRewardToken(tokens);
	uint256 newBalance =
	(IERC20(_rewardTokenAddress).balanceOf(address(this))).sub(initialRewardTokenBalance)
	IERC20(_rewardTokenAddress).transfer(feeAddress, newBalance);
	}

Recommendation	Include a *require* statement to check the balance.
Status	**Fixed**

Severity	Medium
Contract	**CryptoMarketingCompany.sol**
Description	**Inefficient usage of conditional statements.**
Function	
Recommendation	The statement ***if(ethAmount-ethFromLiquidity>0)*** will fail if *ethFromLiquidity* is greater than *ethAmount.* Usage of ***if(ethAmount>ethFromLiquidity)*** can be more efficient and reduces operations.
Status	**Fixed**

```
ftrace | funcSig
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    // approve token transfer to cover all possible scenarios
    approve(address(this), address(uniswapV2Router), tokenAmount);

    // add the liquidity
    (,uint256 ethFromLiquidity,) = uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        address(0),
        block.timestamp
    );
    if (ethAmount - ethFromLiquidity > 0)
        payable(marketingWalletAddress).sendValue(ethAmount - ethFromLiquidity);
}
```

Severity	Medium
Contract	CryptoMarketingCompany.sol
Description	Functions with the same functionality.

Function	ftrace | funcSig
	`function updateWeeklyDividendTracker(address newAddress ) public onlyOwner {`
	`require(`
	`newAddress != address(weeklyDividendTracker),`
	`"CMC: The dividend tracker already exists"`
	`);`
	`CMCDividendTracker newDividendTracker = CMCDividendTracker(`
	`payable(newAddress )`
	`);`
	`require(`
	`newDividendTracker.owner() == address(this),`
	`"CMC: The new dividend tracker must be owned by the CMC token contract"`
	`);`
	`newDividendTracker.excludeFromDividends(address(newDividendTracker));`
	`newDividendTracker.excludeFromDividends(address(monthlyDividendTracker));`
	`newDividendTracker.excludeFromDividends(address(this));`
	`newDividendTracker.excludeFromDividends(owner());`
	`newDividendTracker.excludeFromDividends(address(uniswapV2Router));`
	`emit UpdateDividendTracker(newAddress , address(weeklyDividendTracker));`
	`weeklyDividendTracker = newDividendTracker;`
	`}`
	ftrace | funcSig
	`function updateMontlyDividendTracker(address newAddress ) public onlyOwner {`
	`require(`
	`newAddress != address(monthlyDividendTracker),`
	`"CMC: The dividend tracker already exists"`
	`);`
	`CMCDividendTracker newDividendTracker = CMCDividendTracker(`
	`payable(newAddress )`
	`);`
	`require(`
	`newDividendTracker.owner() == address(this),`
	`"CMC: The new dividend tracker must be owned by the CMC token contract"`
	`);`
	`newDividendTracker.excludeFromDividends(address(weeklyDividendTracker));`
	`newDividendTracker.excludeFromDividends(address(newDividendTracker));`
	`newDividendTracker.excludeFromDividends(address(this));`
	`newDividendTracker.excludeFromDividends(owner());`
	`newDividendTracker.excludeFromDividends(address(uniswapV2Router));`
	`emit UpdateDividendTracker(newAddress , address(monthlyDividendTracker));`
	`monthlyDividendTracker = newDividendTracker;`
	`}`

Recommendation	These functions can be merged into a single function by passing an additional argument of dividendTrackerType. This will save gas used for deployment.

Status	**Fixed**

Severity	**Medium**
Contract	**CryptoMarketingCompany.sol**
Description	**Lack of update in token balance.**

Function	ftrace | funcSig
	`function swapAndSendToFee(uint256 tokens ) private {`
	`  payable( _marketingWalletAddress).transfer(swapTokensForEth(tokens ));`
	`}`
	
	ftrace | funcSig
	`function swapAndSendDividends(uint256 tokens , bool isWeekly ) private {`
	`  swapTokensForRewardToken(tokens );`
	`  uint256 dividends = IERC20( _rewardTokenAddress).balanceOf(address(this));`
	`  address txAddress =`
	`  isWeekly ? address (weeklyDividendTracker) : address(monthlyDividendTracker);`
	`  bool success = IERC20( _rewardTokenAddress).transfer(txAddress,dividends);`
	
	`  if (success) {`
	`  if (isWeekly )`
	`  weeklyDividendTracker.distributeRewardTokenDividends(dividends);`
	`  else`
	`  monthlyDividendTracker.distributeRewardTokenDividends(dividends);`
	
	`  emit SendDividends(tokens , dividends);`
	`  }`
	`}`

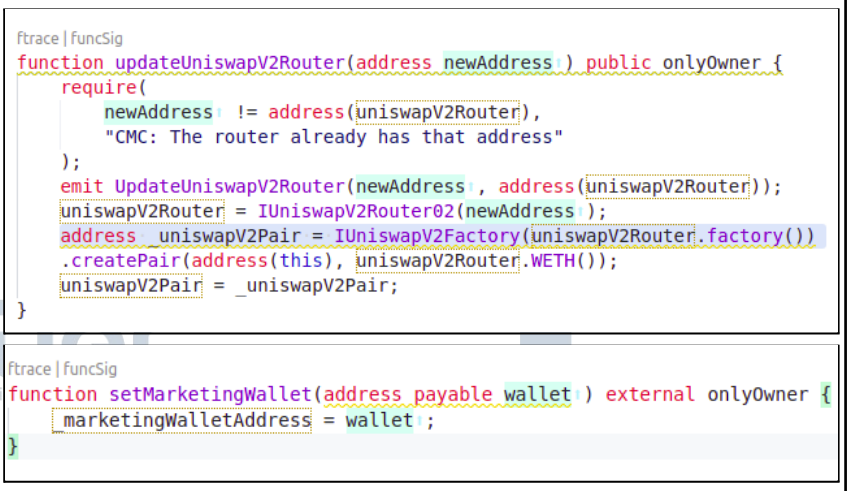
Recommendation	Once the "tokens" are transferred, they should be deducted from the marketing tokens to update their value. Otherwise the tokens will keep on accumulating.
Status	**Fixed**

Severity	Medium
Contract	CryptoMarketingCompany.sol
Description	Usage of same events

Function	ftrace | funcSig
	function processDividendTrackers(uint256 gas) external {
	(
	uint256 iterations,
	uint256 claims,
	uint256 lastProcessedIndex
	) = weeklyDividendTracker.process(gas);
	emit ProcessedDividendTracker(
	iterations,
	claims,
	lastProcessedIndex,
	false,
	gas,
	tx.origin
	);
	(
	iterations,
	claims,
	lastProcessedIndex
	) = monthlyDividendTracker.process(gas);
	emit ProcessedDividendTracker(
	iterations,
	claims,
	lastProcessedIndex,
	false,
	gas,
	tx.origin
	);
	}

Recommendation	Specify tracker type in the events.
Status	**Fixed**

Low Severity Vulnerabilities

Severity	Low
Contract	CryptoMarketingCompany.sol
Description	Lack of zero-check
Code Snippet	
Recommendation	Constructor should have a method to check owner address given is not a zero address
Status	Fixed

```
ftrace | funcSig
function updateUniswapV2Router(address newAddress) public onlyOwner {
    require(
        newAddress != address(uniswapV2Router),
        "CMC: The router already has that address"
    );
    emit UpdateUniswapV2Router(newAddress, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress);
    address _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory())
    .createPair(address(this), uniswapV2Router.WETH());
    uniswapV2Pair = _uniswapV2Pair;
}
```

```
ftrace | funcSig
function setMarketingWallet(address payable wallet) external onlyOwner {
    marketingWalletAddress = wallet;
}
```

Informational Severity Vulnerabilities

Severity	Informational
Contract	CryptoMarketingCompany.sol
Description	Dead-code: Functions that are not used.

Function	ftrace | funcSig
	function swapAndSendToFeeInRewardToken(uint256 tokens, address feeAddress) private {
	uint256 initialRewardTokenBalance =
	IERC20(_rewardTokenAddress).balanceOf(address(this));
	swapTokensForRewardToken(tokens);
	uint256 newBalance =
	(IERC20(_rewardTokenAddress).balanceOf(address(this))).sub(initialRewardTokenBalance);
	IERC20(_rewardTokenAddress).transfer(feeAddress, newBalance);
	}

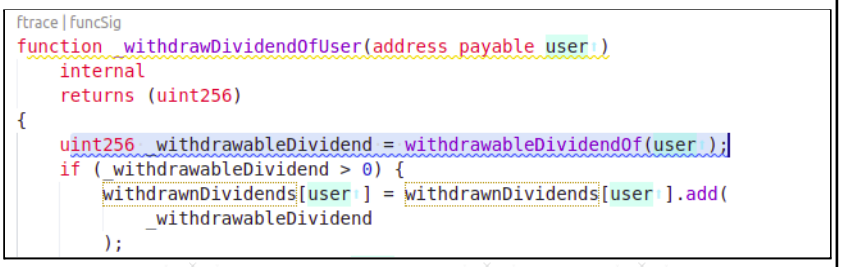
Recommendation	Remove unused functions.
Status	Fixed

Severity	Informational
Contract	**CryptoMarketingCompany.sol**
Description	**Conformance to Solidity naming conventions**
Recommendation	Follow the Solidity naming convention. Solidity defines a naming convention that should be followed. Rule exceptions ● Allow constant variable name/symbol/decimals to be lowercase (ERC20). ● Allow _ at the beginning of the mixed_case match for private variables and unused parameters.
Status	**Acknowledged**

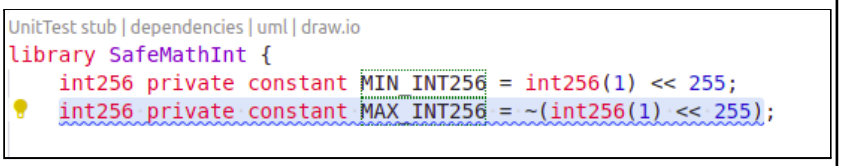
Severity	Informational
Contract	Context.sol
Description	**Redundant Statements** **Detect the usage of redundant statements that have no effect.**

Function	ftrace | funcSig
	`function _msgData() internal view virtual returns (bytes memory) {`
	`  this; // silence state mutability warning without generating bytecode`
	`  return msg.data;`
	`}`

Recommendation	Remove redundant statements if they congest code but offer no value.
Status	**Acknowledged**

Severity	Informational
Contract	DividendPayingToken.sol
Description	Prevent variables from having similar names.
Function	
Recommendation Description	Variable names too similar. Detect variables with names that are too similar.
Status	Acknowledged

```
ftrace | funcSig
function _withdrawDividendOfUser(address payable user)
    internal
    returns (uint256)
{
    uint256 _withdrawableDividend = withdrawableDividendOf(user);
    if (_withdrawableDividend > 0) {
        withdrawnDividends[user] = withdrawnDividends[user].add(
            _withdrawableDividend
        );
```

Severity	Informational
Contract	SafeMathInt.sol
Description	Unused state variable
Function	
Recommendation	Remove unused state variables.
Status	Acknowledged

Severity	Informational
Contract	CryptoMarketingCompany.sol
Description	Public function that could be declared external. public functions that are never called by the contract should be declared external to save gas.

Function	ftrace | funcSig
	function updateWeeklyDividendTracker(address newAddress) public onlyOwner {
	
	ftrace | funcSig
	function updateMontlyDividendTracker(address newAddress) public onlyOwner {
	
	ftrace | funcSig
	function updateUniswapV2Router(address newAddress) public onlyOwner {

Recommendation	Use the external attribute for functions never called from the contract.
Status	Fixed

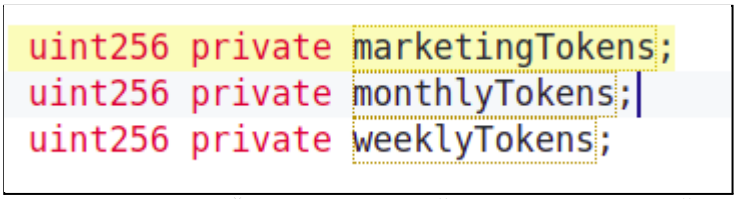
Severity	Informational
Contract	**CryptoMarketingCompany.sol**
Description	**Function name spelled wrongly.**
Function	`function updateMontlyDividendTracker(address newAddress) public onlyOwner { require( newAddress != address(monthlyDividendTracker), "CMC: The dividend tracker already exists" );`
Recommendation	Ensure the names are spelled right in order to ease implementation.
Status	**Fixed**(Function merged)

Severity	Informational
Contract	**CryptoMarketingCompany.sol**
Description	**Usage of longer string messages.**

Function	`ftrace | funcSig`
	`function setAutomatedMarketMakerPair(address pair, bool value)`
	`public`
	`onlyOwner`
	`{`
	`require(`
	`pair != uniswapV2Pair,`
	`"CMC: The PanusdtSwap pair cannot be removed from automatedMarketMakerPairs"`
	`);`
	`_setAutomatedMarketMakerPair(pair, value);`
	`}`

Recommendation	Usage of longer string messages can result in more usage of gas.

Status	**Fixed**

Severity	**Informational**
Contract	**CryptoMarketingCompany.sol**
Description	**Private variables that could be declared public.**
Function	
Recommendation	These variables are private in the contract but they should be public because the getter function gets automatically generated for a public variable.
Status	**Fixed**

Functional Test Status

CryptoMarketingCompany

Function	Testing Result	Status
Constructor(deploy)	Passed	Passed
updateWeeklyDividendTracker	Passed	Passed
updateMontlyDividendTracker	Passed	Passed
updateUniswapV2Router	Passed	Passed
excludeFromFees	Passed	Passed
excludeMultipleAccountsFromFees	Passed	Passed
setMarketingWallet	Passed	Passed
switchRewardTokenAddress	Passed	Passed
setRewardTokenWeeklyRewardsFees	Passed	Passed
setRewardTokenMonthlyRewardsFees	Passed	Passed
setLiquidityFees	Passed	Passed
setMarketingFees	Passed	Passed
setAutomatedMarketMakerPair	Passed	Passed
blacklistAddress	Passed	Passed
updateGasForProcessing	Passed	Passed
updateWeeklyClaimWait	Passed	Passed
updateMonthlyClaimWait	Passed	Passed
excludeFromDividends	Passed	Passed
processDividendTrackers	Passed	Passed
claim	Passed	Passed
Transfer	Failed	Fixed

Technical Analysis

The following is our automated and manual analysis of the CMC Coin Smart Contract code:

Checked Vulnerabilities

We checked CMC Coin Smart Contract for commonly known and specific business logic vulnerabilities. Following is the list of vulnerabilities tested in the Smart Contract code:

- Reentrancy
- Timestamp Dependence
- Gas limit and loops
- DoS with (unexpected) throw
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fall back function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unused returns
- Unused state variables
- Unused functions
- Unchecked transfer
- Unsafe type inference
- Implicit visibility level
- Address hardcoded
- Using delete for arrays
- Integer overflow/underflow
- Locked money
- Private modifier
- Revert/require functions
- Using var
- Visibility

Automation Report

TAS used Slither and Mythril to generate the automation report.

Slither

Slither is a Solidity static analysis framework written in Python 3. It runs a suite of vulnerability detectors, prints visual information about contract details, and provides an API to write custom analyses. Slither enables developers to find vulnerabilities, enhance their code comprehension, and quickly prototype custom analyses.

```
Compiled with Builder
Number of lines: 5201 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 19 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 28
Number of informational issues: 73
Number of low issues: 41
Number of medium issues: 13
Number of high issues: 4
ERCs: ERC20

+-----------------------+-------------+-------+--------------------+-------------+--------------------+
|          Name         | # functions | ERCS  |     ERC20 info     | Complex code |      Features      |
+-----------------------+-------------+-------+--------------------+-------------+--------------------+
|        Address        |      7      |       |                    |      No      |      Send ETH      |
|                       |             |       |                    |              |      Assembly      |
|   CMCDividendTracker  |      65     | ERC20 |     No Minting     |     Yes      | Tokens interaction |
|                       |             |       | Approve Race Cond. |              |                    |
|                       |             |       |                    |              |                    |
| CryptoMarketingCompany |      79     | ERC20 |     No Minting     |     Yes      |    Receive ETH     |
|                       |             |       | Approve Race Cond. |              |      Send ETH      |
|                       |             |       |                    |              | Tokens interaction |
|   IUniswapV2Factory   |      8      |       |                    |      No      |                    |
|     IUniswapV2Pair    |      27     | ERC20 |     ∞ Minting      |      No      |                    |
|                       |             |       | Approve Race Cond. |              |                    |
|                       |             |       |                    |              |                    |
|   IUniswapV2Router02  |      24     |       |                    |      No      |    Receive ETH     |
|    IterableMapping    |      6      |       |                    |      No      |                    |
|       SafeMath        |      8      |       |                    |      No      |                    |
|      SafeMathInt      |      7      |       |                    |      No      |                    |
|      SafeMathUint     |      1      |       |                    |      No      |                    |
+-----------------------+-------------+-------+--------------------+-------------+--------------------+
INFO:Slither:. analyzed (19 contracts)
```

```
INFO:Detectors:
CryptoMarketingCompany.swapAndSendToFee(uint256) (contracts/CryptoMarketingCompany.sol#574-576) sends eth to arb
itrary user
        Dangerous calls:
        - address(_marketingWalletAddress).transfer(swapTokensForEth(tokens)) (contracts/CryptoMarketingCompany.
sol#575)
CryptoMarketingCompany.addLiquidity(uint256,uint256) (contracts/CryptoMarketingCompany.sol#629-644) sends eth to
 arbitrary user
        Dangerous calls:
        - (ethFromLiquidity) = uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,a
ddress(0),block.timestamp) (contracts/CryptoMarketingCompany.sol#634-641)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-
destinations
```

```
INFO:Detectors:
Reentrancy in CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#496-572):
        External calls:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
        - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketingCompany.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#656)
        - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketingCompany.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#656)
        - swapAndLiquify(liquidityTokens) (contracts/CryptoMarketingCompany.sol#536)
                - (ethFromLiquidity) = uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmou
```

```
nt,0,0,address(0),block.timestamp) (contracts/CryptoMarketingCompany.sol#634-641)
                - (success) = recipient.call{value: amount}() (contracts/Address.sol#58)
                - address(_marketingWalletAddress).sendValue(ethAmount - ethFromLiquidity) (contracts/CryptoMarketingCompany.sol#643)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
        External calls sending eth:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - address(_marketingWalletAddress).transfer(swapTokensForEth(tokens)) (contracts/CryptoMarketingCompany.sol#575)
        - swapAndLiquify(liquidityTokens) (contracts/CryptoMarketingCompany.sol#536)
                - (ethFromLiquidity) = uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/CryptoMarketingCompany.sol#634-641)
                - (success) = recipient.call{value: amount}() (contracts/Address.sol#58)
        State variables written after the call(s):
        - super._transfer(from,address(this),fees) (contracts/CryptoMarketingCompany.sol#551)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (contracts/ERC20.sol#274-277)
                - _balances[recipient] = _balances[recipient].add(amount) (contracts/ERC20.sol#278)
        - super._transfer(from,to,amount) (contracts/CryptoMarketingCompany.sol#554)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (contracts/ERC20.sol#274-277)
                - _balances[recipient] = _balances[recipient].add(amount) (contracts/ERC20.sol#278)
        - (transferAmount,fees) = getTransferAmounts(amount,feeType) (contracts/CryptoMarketingCompany.sol#549)
                - marketingTokens = marketingTokens.add(_marketingTokens) (contracts/CryptoMarketingCompany.sol#489)
        - (transferAmount,fees) = getTransferAmounts(amount,feeType) (contracts/CryptoMarketingCompany.sol#549)
                - monthlyTokens = monthlyTokens.add(_monthlyRewardTokens) (contracts/CryptoMarketingCompany.sol#490)
```

```
        - swapping = false (contracts/CryptoMarketingCompany.sol#538)
        - (transferAmount,fees) = getTransferAmounts(amount,feeType) (contracts/CryptoMarketingCompany.sol#549)
                - weeklyTokens = weeklyTokens.add(_weeklyRewardTokens) (contracts/CryptoMarketingCompany.sol#491)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
CryptoMarketingCompany.swapAndSendToFeeInRewardToken(uint256,address) (contracts/CryptoMarketingCompany.sol#244-249) ignores return value by IERC20(_rewardTokenAddress).transfer(feeAddress,newBalance) (contracts/CryptoMarketingCompany.sol#248)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

```
INFO:Detectors:
Reentrancy in DividendPayingToken._withdrawDividendOfUser(address) (contracts/DividendPayingToken.sol#83-106):
        External calls:
        - success = IERC20(_rewardTokenAddress).transfer(user,_withdrawableDividend) (contracts/DividendPayingTo
ken.sol#93)
        State variables written after the call(s):
        - withdrawnDividends[user] = withdrawnDividends[user].sub(_withdrawableDividend) (contracts/DividendPayi
ngToken.sol#96-98)
Reentrancy in CryptoMarketingCompany.updateMontlyDividendTracker(address) (contracts/CryptoMarketingCompany.sol#
174-198):
        External calls:
        - newDividendTracker.excludeFromDividends(address(weeklyDividendTracker)) (contracts/CryptoMarketingComp
any.sol#189)
        - newDividendTracker.excludeFromDividends(address(newDividendTracker)) (contracts/CryptoMarketingCompany
.sol#190)
        - newDividendTracker.excludeFromDividends(address(this)) (contracts/CryptoMarketingCompany.sol#191)
        - newDividendTracker.excludeFromDividends(owner()) (contracts/CryptoMarketingCompany.sol#192)
        - newDividendTracker.excludeFromDividends(address(uniswapV2Router)) (contracts/CryptoMarketingCompany.so
l#193)
        State variables written after the call(s):
        - monthlyDividendTracker = newDividendTracker (contracts/CryptoMarketingCompany.sol#197)
Reentrancy in CryptoMarketingCompany.updateWeeklyDividendTracker(address) (contracts/CryptoMarketingCompany.sol#
148-172):
        External calls:
        - newDividendTracker.excludeFromDividends(address(newDividendTracker)) (contracts/CryptoMarketingCompany
.sol#163)
        - newDividendTracker.excludeFromDividends(address(monthlyDividendTracker)) (contracts/CryptoMarketingCom
pany.sol#164)
```

```
INFO:Detectors:
CryptoMarketingCompany._transfer(address,address,uint256).claims (contracts/CryptoMarketingCompany.sol#564) is a
 local variable never initialized
CryptoMarketingCompany._transfer(address,address,uint256).claims_scope_1 (contracts/CryptoMarketingCompany.sol#5
68) is a local variable never initialized
CryptoMarketingCompany._transfer(address,address,uint256).lastProcessedIndex (contracts/CryptoMarketingCompany.s
ol#564) is a local variable never initialized
CryptoMarketingCompany._transfer(address,address,uint256).lastProcessedIndex_scope_2 (contracts/CryptoMarketingC
ompany.sol#568) is a local variable never initialized
CryptoMarketingCompany._transfer(address,address,uint256).iterations (contracts/CryptoMarketingCompany.sol#564)
is a local variable never initialized
CryptoMarketingCompany._transfer(address,address,uint256).iterations_scope_0 (contracts/CryptoMarketingCompany.s
ol#568) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CryptoMarketingCompany.claim() (contracts/CryptoMarketingCompany.sol#462-465) ignores return value by weeklyDivi
dendTracker.processAccount(msg.sender,false) (contracts/CryptoMarketingCompany.sol#463)
CryptoMarketingCompany.claim() (contracts/CryptoMarketingCompany.sol#462-465) ignores return value by monthlyDiv
idendTracker.processAccount(msg.sender,false) (contracts/CryptoMarketingCompany.sol#464)
CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#496-572) ignores
 return value by weeklyDividendTracker.process(gas) (contracts/CryptoMarketingCompany.sol#564-566)
CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#496-572) ignores
 return value by monthlyDividendTracker.process(gas) (contracts/CryptoMarketingCompany.sol#568-570)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
INFO:Detectors:
DividendPayingToken.constructor(string,string)._name (contracts/DividendPayingToken.sol#57) shadows:
        - ERC20._name (contracts/ERC20.sol#48) (state variable)
DividendPayingToken.constructor(string,string)._symbol (contracts/DividendPayingToken.sol#57) shadows:
        - ERC20._symbol (contracts/ERC20.sol#49) (state variable)
DividendPayingToken.dividendOf(address)._owner (contracts/DividendPayingToken.sol#111) shadows:
        - Ownable._owner (contracts/Ownable.sol#7) (state variable)
DividendPayingToken.withdrawableDividendOf(address)._owner (contracts/DividendPayingToken.sol#118) shadows:
        - Ownable._owner (contracts/Ownable.sol#7) (state variable)
DividendPayingToken.withdrawnDividendOf(address)._owner (contracts/DividendPayingToken.sol#130) shadows:
        - Ownable._owner (contracts/Ownable.sol#7) (state variable)
DividendPayingToken.accumulativeDividendOf(address)._owner (contracts/DividendPayingToken.sol#144) shadows:
        - Ownable._owner (contracts/Ownable.sol#7) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
DividendPayingToken.switchRewardTokenAddress(address).rewardTokenAddress (contracts/DividendPayingToken.sol#218)
 lacks a zero-check on :
                - _rewardTokenAddress = rewardTokenAddress (contracts/DividendPayingToken.sol#219)
Ownable.constructor().msgSender (contracts/Ownable.sol#18) lacks a zero-check on :
                - _owner = msgSender (contracts/Ownable.sol#19)
CryptoMarketingCompany.updateUniswapV2Router(address)._uniswapV2Pair (contracts/CryptoMarketingCompany.sol#207-2
08) lacks a zero-check on :
                - uniswapV2Pair = _uniswapV2Pair (contracts/CryptoMarketingCompany.sol#209)
```

```
INFO:Detectors:
DividendPayingToken.switchRewardTokenAddress(address).rewardTokenAddress (contracts/DividendPayingToken.sol#218)
 lacks a zero-check on :
                - _rewardTokenAddress = rewardTokenAddress (contracts/DividendPayingToken.sol#219)
Ownable.constructor().msgSender (contracts/Ownable.sol#18) lacks a zero-check on :
                - _owner = msgSender (contracts/Ownable.sol#19)
CryptoMarketingCompany.updateUniswapV2Router(address)._uniswapV2Pair (contracts/CryptoMarketingCompany.sol#207-2
08) lacks a zero-check on :
                - uniswapV2Pair = _uniswapV2Pair (contracts/CryptoMarketingCompany.sol#209)
CryptoMarketingCompany.setMarketingWallet(address).wallet (contracts/CryptoMarketingCompany.sol#233) lacks a zer
o-check on :
                - _marketingWalletAddress = wallet (contracts/CryptoMarketingCompany.sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
Reentrancy in CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#49
6-572):
        External calls:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(
this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
        - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketing
Company.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,addre
ss(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingComp
any.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCom
pany.sol#656)
        External calls sending eth:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - address(_marketingWalletAddress).transfer(swapTokensForEth(tokens)) (contracts/CryptoMarketing
Company.sol#575)
        State variables written after the call(s):
        - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
                - _allowances[owner][spender] = amount (contracts/ERC20.sol#346)
Reentrancy in CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#49
6-572):
        External calls:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(
this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
```



```
        External calls:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(
this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
        - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketing
Company.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,addre
ss(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingComp
any.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCom
pany.sol#656)
        - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketing
Company.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,addre
ss(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingComp
any.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCom
pany.sol#656)
        External calls sending eth:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - address(_marketingWalletAddress).transfer(swapTokensForEth(tokens)) (contracts/CryptoMarketing
Company.sol#575)
```

```
        State variables written after the call(s):
        - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
                - _allowances[owner][spender] = amount (contracts/ERC20.sol#346)
Reentrancy in CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#496-572):
        External calls:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
        - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketingCompany.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#656)
        - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketingCompany.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#656)
```

```
        - swapAndLiquify(liquidityTokens) (contracts/CryptoMarketingCompany.sol#536)
                - (ethFromLiquidity) = uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/CryptoMarketingCompany.sol#634-641)
                - (success) = recipient.call{value: amount}() (contracts/Address.sol#58)
                - address(_marketingWalletAddress).sendValue(ethAmount - ethFromLiquidity) (contracts/CryptoMarketingCompany.sol#643)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
        External calls sending eth:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - address(_marketingWalletAddress).transfer(swapTokensForEth(tokens)) (contracts/CryptoMarketingCompany.sol#575)
        - swapAndLiquify(liquidityTokens) (contracts/CryptoMarketingCompany.sol#536)
                - (ethFromLiquidity) = uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (contracts/CryptoMarketingCompany.sol#634-641)
                - (success) = recipient.call{value: amount}() (contracts/Address.sol#58)
        State variables written after the call(s):
        - swapAndLiquify(liquidityTokens) (contracts/CryptoMarketingCompany.sol#536)
                - _allowances[owner][spender] = amount (contracts/ERC20.sol#346)
Reentrancy in CryptoMarketingCompany.constructor() (contracts/CryptoMarketingCompany.sol#104-144):
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (contracts/CryptoMarketingCompany.sol#112-113)
        State variables written after the call(s):
        - uniswapV2Pair = _uniswapV2Pair (contracts/CryptoMarketingCompany.sol#116)
        - uniswapV2Router = _uniswapV2Router (contracts/CryptoMarketingCompany.sol#115)
Reentrancy in CryptoMarketingCompany.constructor() (contracts/CryptoMarketingCompany.sol#104-144):
```

```
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (contracts/CryptoMarketingCompany.sol#112-113)
        State variables written after the call(s):
        - uniswapV2Pair = _uniswapV2Pair (contracts/CryptoMarketingCompany.sol#116)
        - uniswapV2Router = _uniswapV2Router (contracts/CryptoMarketingCompany.sol#115)
Reentrancy in CryptoMarketingCompany.constructor() (contracts/CryptoMarketingCompany.sol#104-144):
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (contracts/CryptoMarketingCompany.sol#112-113)
        - _setAutomatedMarketMakerPair(_uniswapV2Pair,true) (contracts/CryptoMarketingCompany.sol#118)
                - weeklyDividendTracker.excludeFromDividends(pair) (contracts/CryptoMarketingCompany.sol#295)
                - monthlyDividendTracker.excludeFromDividends(pair) (contracts/CryptoMarketingCompany.sol#296)
        - weeklyDividendTracker.excludeFromDividends(address(weeklyDividendTracker)) (contracts/CryptoMarketingCompany.sol#121)
        - weeklyDividendTracker.excludeFromDividends(address(monthlyDividendTracker)) (contracts/CryptoMarketingCompany.sol#122)
        - weeklyDividendTracker.excludeFromDividends(address(this)) (contracts/CryptoMarketingCompany.sol#123)
        - weeklyDividendTracker.excludeFromDividends(owner()) (contracts/CryptoMarketingCompany.sol#124)
        - weeklyDividendTracker.excludeFromDividends(deadWallet) (contracts/CryptoMarketingCompany.sol#125)
        - weeklyDividendTracker.excludeFromDividends(address(_uniswapV2Router)) (contracts/CryptoMarketingCompany.sol#126)
        - monthlyDividendTracker.excludeFromDividends(address(weeklyDividendTracker)) (contracts/CryptoMarketingCompany.sol#127)
        - monthlyDividendTracker.excludeFromDividends(address(monthlyDividendTracker)) (contracts/CryptoMarketingCompany.sol#128)
```

```
        - monthlyDividendTracker.excludeFromDividends(address(this)) (contracts/CryptoMarketingCompany.sol#129)
        - monthlyDividendTracker.excludeFromDividends(owner()) (contracts/CryptoMarketingCompany.sol#130)
        - monthlyDividendTracker.excludeFromDividends(deadWallet) (contracts/CryptoMarketingCompany.sol#131)
        - monthlyDividendTracker.excludeFromDividends(address(_uniswapV2Router)) (contracts/CryptoMarketingCompa
ny.sol#132)
        State variables written after the call(s):
        - _mint(owner(),50_000_000 * (10 ** 18)) (contracts/CryptoMarketingCompany.sol#143)
                - _balances[account] = _balances[account].add(amount) (contracts/ERC20.sol#297)
        - excludeFromFees(owner(),true) (contracts/CryptoMarketingCompany.sol#135)
                - _isExcludedFromFees[account] = excluded (contracts/CryptoMarketingCompany.sol#217)
        - excludeFromFees(_marketingWalletAddress,true) (contracts/CryptoMarketingCompany.sol#136)
                - _isExcludedFromFees[account] = excluded (contracts/CryptoMarketingCompany.sol#217)
        - excludeFromFees(address(this),true) (contracts/CryptoMarketingCompany.sol#137)
                - _isExcludedFromFees[account] = excluded (contracts/CryptoMarketingCompany.sol#217)
        - _mint(owner(),50_000_000 * (10 ** 18)) (contracts/CryptoMarketingCompany.sol#143)
                - _totalSupply = _totalSupply.add(amount) (contracts/ERC20.sol#296)
Reentrancy in CMCDividendTracker.processAccount(address,bool) (contracts/CMCDividendTracker.sol#257-271):
        External calls:
        - amount = _withdrawDividendOfUser(account) (contracts/CMCDividendTracker.sol#262)
                - success = IERC20(_rewardTokenAddress).transfer(user,_withdrawableDividend) (contracts/Dividend
PayingToken.sol#93)
        State variables written after the call(s):
        - lastClaimTimes[account] = block.timestamp (contracts/CMCDividendTracker.sol#265)
Reentrancy in CryptoMarketingCompany.swapAndLiquify(uint256) (contracts/CryptoMarketingCompany.sol#578-589):
        External calls:
        - newBalance = swapTokensForEth(half) (contracts/CryptoMarketingCompany.sol#583)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(
this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
```

```
        External calls:
        - weeklyDividendTracker.excludeFromDividends(pair) (contracts/CryptoMarketingCompany.sol#295)
        - monthlyDividendTracker.excludeFromDividends(pair) (contracts/CryptoMarketingCompany.sol#296)
        Event emitted after the call(s):
        - SetAutomatedMarketMakerPair(pair,value) (contracts/CryptoMarketingCompany.sol#299)
Reentrancy in CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#49
6-572):
        External calls:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(
this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
        - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketing
Company.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,addre
ss(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingComp
any.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCom
pany.sol#656)
        External calls sending eth:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - address(_marketingWalletAddress).transfer(swapTokensForEth(tokens)) (contracts/CryptoMarketing
Company.sol#575)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/ERC20.sol#347)
                - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
        - SendDividends(tokens,dividends) (contracts/CryptoMarketingCompany.sol#658)
                - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
```

```
        - addLiquidity(otherHalf,newBalance) (contracts/CryptoMarketingCompany.sol#586)
                - (ethFromLiquidity) = uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmou
nt,0,0,address(0),block.timestamp) (contracts/CryptoMarketingCompany.sol#634-641)
                - (success) = recipient.call{value: amount}() (contracts/Address.sol#58)
                - address(_marketingWalletAddress).sendValue(ethAmount - ethFromLiquidity) (contracts/CryptoMark
etingCompany.sol#643)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (contracts/CryptoMarketingCompany.sol#586)
                - (ethFromLiquidity) = uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmou
nt,0,0,address(0),block.timestamp) (contracts/CryptoMarketingCompany.sol#634-641)
                - (success) = recipient.call{value: amount}() (contracts/Address.sol#58)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (contracts/CryptoMarketingCompany.sol#586)
                - _allowances[owner][spender] = amount (contracts/ERC20.sol#346)
Reentrancy in CryptoMarketingCompany.updateUniswapV2Router(address) (contracts/CryptoMarketingCompany.sol#200-21
0):
        External calls:
        - _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(address(this),uniswapV2Router
.WETH()) (contracts/CryptoMarketingCompany.sol#207-208)
        State variables written after the call(s):
        - uniswapV2Pair = _uniswapV2Pair (contracts/CryptoMarketingCompany.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CryptoMarketingCompany._setAutomatedMarketMakerPair(address,bool) (contracts/CryptoMarketingCompan
y.sol#287-300):
```

```
Reentrancy in CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#496-572):
        External calls:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
        - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketingCompany.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#656)
        - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketingCompany.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#656)
        External calls sending eth:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - address(_marketingWalletAddress).transfer(swapTokensForEth(tokens)) (contracts/CryptoMarketingCompany.sol#575)
```

```
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/ERC20.sol#347)
                - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
        - SendDividends(tokens,dividends) (contracts/CryptoMarketingCompany.sol#658)
                - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
Reentrancy in CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#496-572):
        External calls:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#601-607)
        - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketingCompany.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#656)
        - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
                - success = IERC20(_rewardTokenAddress).transfer(txAddress,dividends) (contracts/CryptoMarketingCompany.sol#650)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/CryptoMarketingCompany.sol#620-626)
                - weeklyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#654)
                - monthlyDividendTracker.distributeRewardTokenDividends(dividends) (contracts/CryptoMarketingCompany.sol#656)
```

```
INFO:Detectors:
Reentrancy in CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#49
6-572):
        External calls:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - address(_marketingWalletAddress).transfer(swapTokensForEth(tokens)) (contracts/CryptoMarketing
Company.sol#575)
        State variables written after the call(s):
        - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
                - _allowances[owner][spender] = amount (contracts/ERC20.sol#346)
        - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
                - _allowances[owner][spender] = amount (contracts/ERC20.sol#346)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/ERC20.sol#347)
                - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
        - Approval(owner,spender,amount) (contracts/ERC20.sol#347)
                - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
        - SendDividends(tokens,dividends) (contracts/CryptoMarketingCompany.sol#658)
                - swapAndSendDividends(weeklyTokens,true) (contracts/CryptoMarketingCompany.sol#532)
        - SendDividends(tokens,dividends) (contracts/CryptoMarketingCompany.sol#658)
                - swapAndSendDividends(monthlyTokens,false) (contracts/CryptoMarketingCompany.sol#533)
Reentrancy in CryptoMarketingCompany._transfer(address,address,uint256) (contracts/CryptoMarketingCompany.sol#49
6-572):
        External calls:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - address(_marketingWalletAddress).transfer(swapTokensForEth(tokens)) (contracts/CryptoMarketing
Company.sol#575)
        External calls sending eth:
        - swapAndSendToFee(marketingTokens) (contracts/CryptoMarketingCompany.sol#531)
                - address(_marketingWalletAddress).transfer(swapTokensForEth(tokens)) (contracts/CryptoMarketing
```

Mythril

Mythril is a security analysis tool for EVM bytecode. It detects security vulnerabilities in smart contracts built for Ethereum, Hedera, Quorum, Vechain, Roostock, Tron, and other EVM-compatible blockchains. It uses symbolic execution, SMT solving, and taint analysis to detect a variety of security vulnerabilities.

```
..ocuments/Audit/standardToken) myth analyze contracts/CryptoMarketingCompany.so
l
The analysis was completed successfully. No issues were detected.
```

Limitations on Disclosure and Use of this Report

This report contains information concerning potential details of CMC Coin and methods for exploiting them. Antier Solutions recommends that precautions should be taken to protect the confidentiality of this document and the information contained herein.

Security Assessment is an uncertain process based on experiences, currently available information, and known threats. All information security systems, which by their nature are dependent on human beings, are vulnerable to some degree. Therefore, although Antier Solutions has identified major security vulnerabilities of the analysed system, there can be no assurance that any exercise of this nature will identify all possible vulnerabilities or propose exhaustive and operationally viable recommendations to mitigate those exposures.

As technologies and risks change over time, the vulnerabilities associated with the operation of the CMC Coin Smart Contract described in this report, as well as the actions necessary to reduce the exposure to such vulnerabilities will also change. Antier Solutions makes no undertaking to supplement or update this report based on the changed circumstances or facts of which Antier Solutions becomes aware after the date hereof.

This report may recommend that Antier Solutions use certain software or hardware products manufactured or maintained by other vendors. Antier Solutions bases these recommendations on its prior experience with the capabilities of those products. Nonetheless, Antier Solutions does not and cannot warrant that a particular product will work as advertised by the vendor, nor that it will operate in the manner intended.

The Non-Disclosure Agreement (NDA) in effect between Antier Solutions and CMC Coin Ltd.governs the disclosure of this report to all other parties, including product vendors and suppliers.