Adamant Metanetwork

Smart Contract Audit Report

SYSFIXED

24 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Adamant Metanetwork	ADMN	Binance Smart Chain

Addresses

Contract address	0xc40657ee2972a9c5b62DD51443c1Bc0FB45e49EA
Contract deployer address	0xcFD5E672c1E1C5C2B1D451b27638493aC437Df5d

Project Website

https://adamantmetanetwork.com/

Codebase

https://bscscan.com/address/0xc40657ee2972a9c5b62DD51443c1Bc0FB45e49EA#code

SUMMARY

The Governance and yield generating token of the Adamant Metanetwork. Have the opportunity to earn BNB dividends, vote on governance decisions, and more within the Adamant Metanetwork!

Contract Summary

Documentation Quality

Adamant Metanetwork provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by Adamant Metanetwork with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 87, 99, 112, 113, 124, 136, 148, 152, 164, 171, 180, 800, 830, 893, 1123, 1133, 1137, 1202, 1312, 1312, 1313, 1372, 1568, 1570, 1572, 1578, 1580, 1582, 1613, 1631, 1703 and 1202.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 1129, 1175, 1203, 1208, 1296, 1297, 1299, 1300, 1301, 1302, 1304, 1305, 1306, 1307, 1318, 1322, 1373, 1638, 1639, 1658, 1659, 1671, 1672 and 1673.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 1471 and 1600.

CONCLUSION

We have audited the Adamant Metanetwork project released on January 2023 to discover issues and identify potential security vulnerabilities in Adamant Metanetwork Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides a satisfactory result with some low-risk issues.

The issues found in the Adamant Metanetwork smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, tx.origin as a part of authorization control and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value. We recommend avoiding "tx.origin" using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	PASS
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	ISSUE FOUND
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Monday Jan 23 2023 14:18:35 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Jan 24 2023 02:15:47 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	CoinToken.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 87

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
86   function add(uint256 a, uint256 b) internal pure returns (uint256) {
87   uint256 c = a + b;
88   require(c >= a, "SafeMath: addition overflow");
89
90   return c;
91
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 99

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
98   require(b <= a, errorMessage);
99   uint256 c = a - b;
100
101   return c;
102   }
103
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 112

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
111
112   uint256 c = a * b;
113   require(c / a == b, "SafeMath: multiplication overflow");
114
115   return c;
116
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 113

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
112   uint256 c = a * b;
113   require(c / a == b, "SafeMath: multiplication overflow");
114
115   return c;
116   }
117
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 124

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
123   require(b > 0, errorMessage);
124   uint256 c = a / b;
125   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
126
127   return c;
128
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 136

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
135   require(b != 0, errorMessage);
136   return a % b;
137   }
138   }
139
140
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 148

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
147   function mul(int256 a, int256 b) internal pure returns (int256) {
148   int256 c = a * b;
149
150   // Detect overflow when multiplying MIN_INT256 with -1
151   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
152
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 152

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
151   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
152   require((b == 0) || (c / b == a));
153   return c;
154   }
155
156
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 164

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
163   // Solidity already throws when dividing by 0.
164   return a / b;
165   }
166
167   /**
168
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 171

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
170   function sub(int256 a, int256 b) internal pure returns (int256) {
171   int256 c = a - b;
172   require((b >= 0 && c <= a) || (b < 0 && c > a));
173   return c;
174   }
175
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 180

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
179   function add(int256 a, int256 b) internal pure returns (int256) {
180   int256 c = a + b;
181   require((b >= 0 && c >= a) || (b < 0 && c < a));
182   return c;
183   }
184
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 800

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
799   //  see https://github.com/ethereum/EIPs/issues/1726#issuecomment-472352728
800   uint256 constant internal magnitude = 2**128;
801
802   uint256 internal magnifiedDividendPerShare;
803
804
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 830

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
829   magnifiedDividendPerShare = magnifiedDividendPerShare.add(
830   (amount).mul(magnitude) / totalSupply()
831   );
832   emit DividendsDistributed(msg.sender, amount);
833
834
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 893

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
892   function accumulativeDividendOf(address _owner) public view override
returns(uint256) {
893   return magnifiedDividendPerShare.mul(balanceOf(_owner)).toInt256Safe()
894   .add(magnifiedDividendCorrections[_owner]).toUint256Safe() / magnitude;
895   }
896
897
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1123

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1122   while(gasUsed < gas && iterations < numberOfTokenHolders) {
1123   _lastProcessedIndex++;
1124
1125   if(_lastProcessedIndex >= tokenHoldersMap.keys.length) {
1126   _lastProcessedIndex = 0;
1127
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1133

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1132   if(processAccount(payable(account), true)) {
1133   claims++;
1134   }
1135   }
1136
1137
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1137

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1136
1137   iterations++;
1138
1139   uint256 newGasLeft = gasleft();
1140
1141
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1202

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1201   uint index = tokenHoldersMap.indexOf[key];
1202   uint lastIndex = tokenHoldersMap.keys.length - 1;
1203   address lastKey = tokenHoldersMap.keys[lastIndex];
1204
1205   tokenHoldersMap.indexOf[lastKey] = index;
1206
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1312

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1311
1312   uint256 totalSupply = totalSupply_ * (10**18);
1313   swapTokensAtAmount = totalSupply.mul(2).div(10**6); // 0.002%
1314
1315   // use by default 300,000 gas to process auto-claiming dividends
1316
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1312

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1311
1312   uint256 totalSupply = totalSupply_ * (10**18);
1313   swapTokensAtAmount = totalSupply.mul(2).div(10**6); // 0.002%
1314
1315   // use by default 300,000 gas to process auto-claiming dividends
1316
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1313

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1312   uint256 totalSupply = totalSupply_ * (10**18);
1313   swapTokensAtAmount = totalSupply.mul(2).div(10**6); // 0.002%
1314
1315   // use by default 300,000 gas to process auto-claiming dividends
1316   gasForProcessing = 300000;
1317
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1372

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1371   function excludeMultipleAccountsFromFees(address[] calldata accounts, bool
excluded) public onlyOwner {
1372   for(uint256 i = 0; i < accounts.length; i++) {
1373   _isExcludedFromFees[accounts[i]] = excluded;
1374   }
1375
1376
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1568

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1567   LFee = amount.mul(buyLiquidityFee).div(100);
1568   AmountLiquidityFee += LFee;
1569   RFee = amount.mul(buyTokenRewardsFee).div(100);
1570   AmountTokenRewardsFee += RFee;
1571   MFee = amount.mul(buyMarketingFee).div(100);
1572
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1570

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1569   RFee = amount.mul(buyTokenRewardsFee).div(100);
1570   AmountTokenRewardsFee += RFee;
1571   MFee = amount.mul(buyMarketingFee).div(100);
1572   AmountMarketingFee += MFee;
1573   DFee = amount.mul(buyDeadFee).div(100);
1574
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1572

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1571   MFee = amount.mul(buyMarketingFee).div(100);
1572   AmountMarketingFee += MFee;
1573   DFee = amount.mul(buyDeadFee).div(100);
1574   fees = LFee.add(RFee).add(MFee).add(DFee);
1575   }
1576
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1578

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1577   LFee = amount.mul(sellLiquidityFee).div(100);
1578   AmountLiquidityFee += LFee;
1579   RFee = amount.mul(sellTokenRewardsFee).div(100);
1580   AmountTokenRewardsFee += RFee;
1581   MFee = amount.mul(sellMarketingFee).div(100);
1582
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1580

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1579   RFee = amount.mul(sellTokenRewardsFee).div(100);
1580   AmountTokenRewardsFee += RFee;
1581   MFee = amount.mul(sellMarketingFee).div(100);
1582   AmountMarketingFee += MFee;
1583   DFee = amount.mul(sellDeadFee).div(100);
1584
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1582

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1581   MFee = amount.mul(sellMarketingFee).div(100);
1582   AmountMarketingFee += MFee;
1583   DFee = amount.mul(sellDeadFee).div(100);
1584   fees = LFee.add(RFee).add(MFee).add(DFee);
1585   }
1586
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1613

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1612   IERC20(rewardToken).transfer(_marketingWalletAddress, newBalance);
1613   AmountMarketingFee = AmountMarketingFee - tokens;
1614   }
1615
1616   function swapAndLiquify(uint256 tokens) private {
1617
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1631

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1630   addLiquidity(otherHalf, newBalance);
1631   AmountLiquidityFee = AmountLiquidityFee - tokens;
1632   emit SwapAndLiquify(half, newBalance, otherHalf);
1633   }
1634
1635
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1703

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1702   swapTokensForToken(tokens);
1703   AmountTokenRewardsFee = AmountTokenRewardsFee - tokens;
1704   uint256 dividends = IERC20(rewardToken).balanceOf(address(this));
1705   bool success = IERC20(rewardToken).transfer(address(dividendTracker), dividends);
1706   if (success) {
1707
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1202

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- CoinToken.sol

Locations

```
1201   uint index = tokenHoldersMap.indexOf[key];
1202   uint lastIndex = tokenHoldersMap.keys.length - 1;
1203   address lastKey = tokenHoldersMap.keys[lastIndex];
1204
1205   tokenHoldersMap.indexOf[lastKey] = index;
1206
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- CoinToken.sol

Locations

```
5   // SPDX-License-Identifier: MIT
6   pragma solidity ^0.8.0;
7
8   abstract contract Context {
9   function _msgSender() internal view virtual returns (address) {
10
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 1471

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- CoinToken.sol

Locations

```
1470   (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) =
dividendTracker.process(gas);
1471   emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, false, gas,
tx.origin);
1472   }
1473
1474   function claim() external {
1475
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 1600

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- CoinToken.sol

Locations

```
1599   try dividendTracker.process(gas) returns (uint256 iterations, uint256 claims,
uint256 lastProcessedIndex) {
1600   emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas,
tx.origin);
1601   }
1602   catch {
1603
1604
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1129

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1128
1129   address account = tokenHoldersMap.keys[_lastProcessedIndex];
1130
1131   if(canAutoClaim(lastClaimTimes[account])) {
1132   if(processAccount(payable(account), true)) {
1133
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1175

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1174   function MAPGetKeyAtIndex(uint index) public view returns (address) {
1175   return tokenHoldersMap.keys[index];
1176   }
1177
1178   function MAPSize() public view returns (uint) {
1179
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1203

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1202   uint lastIndex = tokenHoldersMap.keys.length - 1;
1203   address lastKey = tokenHoldersMap.keys[lastIndex];
1204
1205   tokenHoldersMap.indexOf[lastKey] = index;
1206   delete tokenHoldersMap.indexOf[key];
1207
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1208

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1207
1208   tokenHoldersMap.keys[index] = lastKey;
1209   tokenHoldersMap.keys.pop();
1210   }
1211   }
1212
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1296

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1295   ) payable ERC20(name_, symbol_)  {
1296   rewardToken = addrs[0];
1297   _marketingWalletAddress = addrs[2];
1298
1299   buyTokenRewardsFee = buyFeeSetting_[0];
1300
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1297

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1296   rewardToken = addrs[0];
1297   _marketingWalletAddress = addrs[2];
1298
1299   buyTokenRewardsFee = buyFeeSetting_[0];
1300   buyLiquidityFee = buyFeeSetting_[1];
1301
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1299

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1298
1299   buyTokenRewardsFee = buyFeeSetting_[0];
1300   buyLiquidityFee = buyFeeSetting_[1];
1301   buyMarketingFee = buyFeeSetting_[2];
1302   buyDeadFee = buyFeeSetting_[3];
1303
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1300

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1299   buyTokenRewardsFee = buyFeeSetting_[0];
1300   buyLiquidityFee = buyFeeSetting_[1];
1301   buyMarketingFee = buyFeeSetting_[2];
1302   buyDeadFee = buyFeeSetting_[3];
1303
1304
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1301

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1300   buyLiquidityFee = buyFeeSetting_[1];
1301   buyMarketingFee = buyFeeSetting_[2];
1302   buyDeadFee = buyFeeSetting_[3];
1303
1304   sellTokenRewardsFee = sellFeeSetting_[0];
1305
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1302

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1301   buyMarketingFee = buyFeeSetting_[2];
1302   buyDeadFee = buyFeeSetting_[3];
1303
1304   sellTokenRewardsFee = sellFeeSetting_[0];
1305   sellLiquidityFee = sellFeeSetting_[1];
1306
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1304

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1303
1304   sellTokenRewardsFee = sellFeeSetting_[0];
1305   sellLiquidityFee = sellFeeSetting_[1];
1306   sellMarketingFee = sellFeeSetting_[2];
1307   sellDeadFee = sellFeeSetting_[3];
1308
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1305

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1304   sellTokenRewardsFee = sellFeeSetting_[0];
1305   sellLiquidityFee = sellFeeSetting_[1];
1306   sellMarketingFee = sellFeeSetting_[2];
1307   sellDeadFee = sellFeeSetting_[3];
1308
1309
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1306

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1305   sellLiquidityFee = sellFeeSetting_[1];
1306   sellMarketingFee = sellFeeSetting_[2];
1307   sellDeadFee = sellFeeSetting_[3];
1308
1309
require(buyTokenRewardsFee.add(buyLiquidityFee).add(buyMarketingFee).add(buyDeadFee) <=
25, "Total buy fee is over 25%");
1310
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1307

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1306   sellMarketingFee = sellFeeSetting_[2];
1307   sellDeadFee = sellFeeSetting_[3];
1308
1309
require(buyTokenRewardsFee.add(buyLiquidityFee).add(buyMarketingFee).add(buyDeadFee) <=
25, "Total buy fee is over 25%");
1310
require(sellTokenRewardsFee.add(sellLiquidityFee).add(sellMarketingFee).add(sellDeadFee)
<= 25, "Total sell fee is over 25%");
1311
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1318

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1317
1318   _node = addrs[3];
1319   dividendTracker = new TokenDividendTracker(rewardToken, tokenBalanceForReward_);
1320
1321
1322
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1322

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1321
1322   IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(addrs[1]);
1323   address _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
1324   .createPair(address(this), _uniswapV2Router.WETH());
1325
1326
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1373

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1372   for(uint256 i = 0; i < accounts.length; i++) {
1373   _isExcludedFromFees[accounts[i]] = excluded;
1374   }
1375
1376   emit ExcludeMultipleAccountsFromFees(accounts, excluded);
1377
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1638

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1637   address[] memory path = new address[](2);
1638   path[0] = address(this);
1639   path[1] = uniswapV2Router.WETH();
1640
1641   _approve(address(this), address(uniswapV2Router), tokenAmount);
1642
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1639

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1638   path[0] = address(this);
1639   path[1] = uniswapV2Router.WETH();
1640
1641   _approve(address(this), address(uniswapV2Router), tokenAmount);
1642
1643
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1658

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1657   address[] memory path = new address[](2);
1658   path[0] = address(this);
1659   path[1] = rewardToken;
1660   _approve(address(this), address(uniswapV2Router), tokenAmount);
1661   // make the swap
1662
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1659

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1658   path[0] = address(this);
1659   path[1] = rewardToken;
1660   _approve(address(this), address(uniswapV2Router), tokenAmount);
1661   // make the swap
1662   uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(
1663
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1671

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1670   address[] memory path = new address[](3);
1671   path[0] = address(this);
1672   path[1] = uniswapV2Router.WETH();
1673   path[2] = rewardToken;
1674   _approve(address(this), address(uniswapV2Router), tokenAmount);
1675
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1672

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1671   path[0] = address(this);
1672   path[1] = uniswapV2Router.WETH();
1673   path[2] = rewardToken;
1674   _approve(address(this), address(uniswapV2Router), tokenAmount);
1675   // make the swap
1676
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1673

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- CoinToken.sol

Locations

```
1672   path[1] = uniswapV2Router.WETH();
1673   path[2] = rewardToken;
1674   _approve(address(this), address(uniswapV2Router), tokenAmount);
1675   // make the swap
1676   uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(
1677
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. 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 Sysfixed's prior written consent in each instance.

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 considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report 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 is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.