Coinary Token

Smart Contract Audit Report

SYSFIXED

07 Sep 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Coinary Token	CYT	Binance Smart Chain

Addresses

Contract address	0xd9025e25bb6cf39f8c926a704039d2dd51088063
Contract deployer address	0xF3Ced1e4F73256C4F34530F3AFF920c79A65f65d

Project Website

https://hub.coinary.com/

Codebase

https://bscscan.com/address/0xd9025e25bb6cf39f8c926a704039d2dd51088063#code

SUMMARY

Coinary token is a token that is hyper deflationary and controls the gamenomics of Dragonary, an NFT blockchain game based on BSC.

Contract Summary

Documentation Quality

Coinary Token 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 Coinary Token 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, 88, 98, 853, 865, 878, 879, 890, 900, 914, 931, 946, 947, 965, 982, 1000, 1020, 1040, 1575, 1876, 1899, 87 and 88.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 9, 310, 503, 531, 750, 831, 1049, 1357, 1401, 1456, 1546, 1588, 1700, 1780 and 1831.
- 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 93, 96 and 138.

CONCLUSION

We have audited the Coinary Token project released on September 2021 to discover issues and identify potential security vulnerabilities in Coinary Token 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 satisfactory results with low-risk issues.

The issues found in the Coinary Token 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 and out-of-bounds array access. The index access expression can cause an exception in case of an invalid array index value.

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.	PASS
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 Sep 06 2021 05:35:19 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Sep 07 2021 21:08:43 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	CoinaryToken.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged

SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	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

- CoinaryToken.sol

Locations

```
86
87   uint256 toDeleteIndex = valueIndex - 1;
88   uint256 lastIndex = set._values.length - 1;
89
90   // When the value to delete is the last one, the swap operation is unnecessary.
However, since this occurs
91
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 88

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
87   uint256 toDeleteIndex = valueIndex - 1;
88   uint256 lastIndex = set._values.length - 1;
89
90   // When the value to delete is the last one, the swap operation is unnecessary.
However, since this occurs
91   // so rarely, we still do the swap anyway to avoid the gas cost of adding an 'if'
statement.
92
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 98

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
97   // Update the index for the moved value
98   set._indexes[lastvalue] = toDeleteIndex + 1; // All indexes are 1-based
99
100   // Delete the slot where the moved value was stored
101   set._values.pop();
102
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 853

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
852   function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
853   uint256 c = a + b;
854   if (c < a) return (false, 0);
855   return (true, c);
856   }
857
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 865

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
864   if (b > a) return (false, 0);
865   return (true, a - b);
866   }
867
868   /**
869
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 878

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
877   if (a == 0) return (true, 0);
878   uint256 c = a * b;
879   if (c / a != b) return (false, 0);
880   return (true, c);
881   }
882
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 879

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
878   uint256 c = a * b;
879   if (c / a != b) return (false, 0);
880   return (true, c);
881   }
882
883
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 890

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
889   if (b == 0) return (false, 0);
890   return (true, a / b);
891   }
892
893   /**
894
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 900

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
899   if (b == 0) return (false, 0);
900   return (true, a % b);
901   }
902
903   /**
904
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 914

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
913   function add(uint256 a, uint256 b) internal pure returns (uint256) {
914   uint256 c = a + b;
915   require(c >= a, "SafeMath: addition overflow");
916   return c;
917   }
918
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 931

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
930   require(b <= a, "SafeMath: subtraction overflow");
931   return a - b;
932   }
933
934   /**
935
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 946

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
945   if (a == 0) return 0;
946   uint256 c = a * b;
947   require(c / a == b, "SafeMath: multiplication overflow");
948   return c;
949   }
950
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 947

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
946   uint256 c = a * b;
947   require(c / a == b, "SafeMath: multiplication overflow");
948   return c;
949   }
950
951
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 965

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
964   require(b > 0, "SafeMath: division by zero");
965   return a / b;
966   }
967
968   /**
969
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 982

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
981   require(b > 0, "SafeMath: modulo by zero");
982   return a % b;
983   }
984
985   /**
986
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1000

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
999   require(b <= a, errorMessage);
1000   return a - b;
1001   }
1002
1003   /**
1004
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1020

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
1019   require(b > 0, errorMessage);
1020   return a / b;
1021   }
1022
1023   /**
1024
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1040

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
1039   require(b > 0, errorMessage);
1040   return a % b;
1041   }
1042   }
1043
1044
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1575

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
1574   // The {SafeMath} overflow check can be skipped here, see the comment at the top
1575   counter._value += 1;
1576   }
1577
1578   function decrement(Counter storage counter) internal {
1579
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1876

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
1875
1876   nextMintingDate = block.timestamp + 1 days;
1877   }
1878
1879   /**
1880
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1899

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
1898   _mint(to, amount);
1899   nextMintingDate = block.timestamp + 1 days;
1900   }
1901
1902   /**
1903
```

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

LINE 87

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
86
87   uint256 toDeleteIndex = valueIndex - 1;
88   uint256 lastIndex = set._values.length - 1;
89
90   // When the value to delete is the last one, the swap operation is unnecessary.
However, since this occurs
91
```

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

LINE 88

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
87   uint256 toDeleteIndex = valueIndex - 1;
88   uint256 lastIndex = set._values.length - 1;
89
90   // When the value to delete is the last one, the swap operation is unnecessary.
However, since this occurs
91   // so rarely, we still do the swap anyway to avoid the gas cost of adding an 'if'
statement.
92
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 9

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
8
9   pragma solidity ^0.7.0;
10
11   /**
12   * @dev Library for managing
13
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 310

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
309
310   pragma solidity ^0.7.0;
311
312   /**
313   * @dev Collection of functions related to the address type
314
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 503

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<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

- CoinaryToken.sol

Locations

```
502
503   pragma solidity >=0.6.0 <0.8.0;
504
505   /*
506   * @dev Provides information about the current execution context, including the
507
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 531

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
530
531   pragma solidity ^0.7.0;
532
533
534
535
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 750

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
749
750   pragma solidity ^0.7.0;
751
752   /**
753   * @dev Interface of the ERC20 standard as defined in the EIP.
754
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 831

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
830
831   pragma solidity ^0.7.0;
832
833   /**
834   * @dev Wrappers over Solidity's arithmetic operations with added overflow
835
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1049

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
1048
1049   pragma solidity ^0.7.0;
1050
1051
1052
1053
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1357

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
1356
1357   pragma solidity ^0.7.0;
1358
1359
1360   /**
1361
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1401

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<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

- CoinaryToken.sol

Locations

```
1400
1401   pragma solidity >=0.6.0 <0.8.0;
1402
1403   /**
1404   * @dev Interface of the ERC20 Permit extension allowing approvals to be made via
signatures, as defined in
1405
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1456

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
1455
1456   pragma solidity ^0.7.0;
1457
1458   /**
1459   * @dev Elliptic Curve Digital Signature Algorithm (ECDSA) operations.
1460
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1546

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
1545
1546   pragma solidity ^0.7.0;
1547
1548   /**
1549   * @title Counters
1550
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1588

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<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

- CoinaryToken.sol

Locations

```
1587
1588   pragma solidity >=0.6.0 <0.8.0;
1589
1590   /**
1591   * @dev https://eips.ethereum.org/EIPS/eip-712[EIP 712] is a standard for hashing
and signing of typed structured data.
1592
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1700

low SEVERITY

The current pragma Solidity directive is ""=0.6.5<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

- CoinaryToken.sol

Locations

```
1699
1700   pragma solidity >=0.6.5 <0.8.0;
1701
1702
1703
1704
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1780

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
1779
1780   pragma solidity ^0.7.0;
1781
1782   /**
1783   * @dev Extension of {ERC20} that adds a cap to the supply of tokens.
1784
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1831

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- CoinaryToken.sol

Locations

```
1830
1831   pragma solidity ^0.7.0;
1832
1833
1834
1835
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 93

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
92
93   bytes32 lastvalue = set._values[lastIndex];
94
95   // Move the last value to the index where the value to delete is
96   set._values[toDeleteIndex] = lastvalue;
97
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 96

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
95   // Move the last value to the index where the value to delete is
96   set._values[toDeleteIndex] = lastvalue;
97   // Update the index for the moved value
98   set._indexes[lastvalue] = toDeleteIndex + 1; // All indexes are 1-based
99
100
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 138

low SEVERITY

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

Source File

- CoinaryToken.sol

Locations

```
137   require(set._values.length > index, "EnumerableSet: index out of bounds");
138   return set._values[index];
139   }
140
141   // Bytes32Set
142
```

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.