Aomen Token

Smart Contract Audit Report

SYSFIXED

03 Feb 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Aomen Token	Aomen	Binance Smart Chain

Addresses

Contract address	0x6859b546FB887fb5018AE0cd01DA0fff2B3f5Bc7
Contract deployer address	0x91b6F79c1282EeD889fA4B6300a1B4023d358Ef9

Project Website

https://t.me/aomenToken

Codebase

https://bscscan.com/address/0x6859b546FB887fb5018AE0cd01DA0fff2B3f5Bc7#code

SUMMARY

Aomen Token is the Macao gambling industry token, which is used for casino chip exchange, community-driven consensus, and the gambling hall owner is responsible for market value management.

Contract Summary

Documentation Quality

Aomen 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 Aomen Token with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 967, 968, 969, 970, 1083, 1084, 1085, 1143, 1145 and 1146.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 25, 39, 54, 55, 68, 80, 95, 109, 123, 137, 153, 176, 199, 225, 640, 663, 696, 699, 721, 724, 750, 752, 802, 996, 996, 1011, 1011, 1018, 1108, 1108, 1108, 1164, 1165, 1166, 1167 and 1168.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 5, 235, 262, 347, 432, 462, 853 and 892.
- 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 986, 987, 1019, 1029, 1030, 1031, 1035, 1036, 1037, 1041, 1042, 1047 and 1048.

CONCLUSION

We have audited the Aomen Token project released on February 2023 to discover issues and identify potential security vulnerabilities in Aomen 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 a satisfactory result with some low-risk issues.

The issues found in the Aomen 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, a floating pragma is set, a state variable visibility is not set, weak sources of randomness, 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.

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.	ISSUE FOUND
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	Thursday Feb 02 2023 13:26:04 GMT+0000 (Coordinated Universal Time)
Finished	Friday Feb 03 2023 04:49:23 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Token.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	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-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-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	low	acknowledged
SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	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 25

low SEVERITY

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

Source File

- Token.sol

Locations

```
24   unchecked {
25   uint256 c = a + b;
26   if (c < a) return (false, 0);
27   return (true, c);
28   }
29
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 39

low SEVERITY

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

Source File

- Token.sol

Locations

```
38   if (b > a) return (false, 0);
39   return (true, a - b);
40   }
41   }
42
43
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 54

low SEVERITY

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

Source File

- Token.sol

Locations

```
53   if (a == 0) return (true, 0);
54   uint256 c = a * b;
55   if (c / a != b) return (false, 0);
56   return (true, c);
57   }
58
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 55

low SEVERITY

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

Source File

- Token.sol

Locations

```
54   uint256 c = a * b;
55   if (c / a != b) return (false, 0);
56   return (true, c);
57   }
58   }
59
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 68

low SEVERITY

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

Source File

- Token.sol

Locations

```
67   if (b == 0) return (false, 0);
68   return (true, a / b);
69   }
70   }
71
72
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 80

low SEVERITY

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

Source File

- Token.sol

Locations

```
79   if (b == 0) return (false, 0);
80   return (true, a % b);
81   }
82   }
83
84
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 95

low SEVERITY

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

Source File

- Token.sol

Locations

```
94   function add(uint256 a, uint256 b) internal pure returns (uint256) {
95   return a + b;
96   }
97
98   /**
99
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 109

low SEVERITY

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

Source File

- Token.sol

Locations

```
108   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
109   return a - b;
110   }
111
112   /**
113
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 123

low SEVERITY

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

Source File

- Token.sol

Locations

```
122   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
123   return a * b;
124   }
125
126   /**
127
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 137

low SEVERITY

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

Source File

- Token.sol

Locations

```
136   function div(uint256 a, uint256 b) internal pure returns (uint256) {
137   return a / b;
138   }
139
140   /**
141
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 153

low SEVERITY

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

Source File

- Token.sol

Locations

```
152   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
153   return a % b;
154   }
155
156   /**
157
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 176

low SEVERITY

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

Source File

- Token.sol

Locations

```
175   require(b <= a, errorMessage);
176   return a - b;
177   }
178   }
179
180
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 199

low SEVERITY

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

Source File

- Token.sol

Locations

```
198   require(b > 0, errorMessage);
199   return a / b;
200   }
201   }
202
203
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 225

low SEVERITY

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

Source File

- Token.sol

Locations

```
224   require(b > 0, errorMessage);
225   return a % b;
226   }
227   }
228   }
229
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 640

low SEVERITY

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

Source File

- Token.sol

Locations

```
639   address owner = _msgSender();
640   _approve(owner, spender, allowance(owner, spender) + addedValue);
641   return true;
642   }
643
644
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 663

low SEVERITY

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

Source File

- Token.sol

Locations

```
662   unchecked {
663   _approve(owner, spender, currentAllowance - subtractedValue);
664   }
665
666   return true;
667
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 696

low SEVERITY

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

Source File

- Token.sol

Locations

```
695   unchecked {
696   _balances[from] = fromBalance - amount;
697   // Overflow not possible: the sum of all balances is capped by totalSupply, and the
sum is preserved by
698   // decrementing then incrementing.
699   _balances[to] += amount;
700
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 699

low SEVERITY

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

Source File

- Token.sol

Locations

```
698   // decrementing then incrementing.
699   _balances[to] += amount;
700   }
701
702   emit Transfer(from, to, amount);
703
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 721

low SEVERITY

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

Source File

- Token.sol

Locations

```
720
721   _totalSupply += amount;
722   unchecked {
723   // Overflow not possible: balance + amount is at most totalSupply + amount, which
is checked above.
724   _balances[account] += amount;
725
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 724

low SEVERITY

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

Source File

- Token.sol

Locations

```
723   // Overflow not possible: balance + amount is at most totalSupply + amount, which
is checked above.
724   _balances[account] += amount;
725   }
726   emit Transfer(address(0), account, amount);
727
728
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 750

low SEVERITY

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

Source File

- Token.sol

Locations

```
749   unchecked {
750   _balances[account] = accountBalance - amount;
751   // Overflow not possible: amount <= accountBalance <= totalSupply.
752   _totalSupply -= amount;
753   }
754
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 752

low SEVERITY

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

Source File

- Token.sol

Locations

```
751   // Overflow not possible: amount <= accountBalance <= totalSupply.
752   _totalSupply -= amount;
753   }
754
755   emit Transfer(account, address(0), amount);
756
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 802

low SEVERITY

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

Source File

- Token.sol

Locations

```
801   unchecked {
802   _approve(owner, spender, currentAllowance - amount);
803   }
804   }
805   }
806
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 996

low SEVERITY

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

Source File

- Token.sol

Locations

```
995   pair=IFactory(router.factory()).getPair(token0, token1);
996   IERC20(token1).approve(address(router),uint(2**256-1));
997   }
998   function setAll(Allot memory allotConfig,autoConfig memory sellconfig,address[]
calldata list ,uint[] memory share)public onlyOwner {
999   setAllot(allotConfig);
1000
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 996

low SEVERITY

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

Source File

- Token.sol

Locations

```
995   pair=IFactory(router.factory()).getPair(token0, token1);
996   IERC20(token1).approve(address(router),uint(2**256-1));
997   }
998   function setAll(Allot memory allotConfig,autoConfig memory sellconfig,address[]
calldata list ,uint[] memory share)public onlyOwner {
999   setAllot(allotConfig);
1000
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1011

low SEVERITY

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

Source File

- Token.sol

Locations

```
1010   token1=token;
1011   IERC20(token1).approve(address(router),uint(2**256-1));
1012   pair=IFactory(router.factory()).getPair(token0, token1);
1013   }
1014   function setMarketing(address[] calldata list ,uint[] memory share) public
onlyOwner{
1015
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1011

low SEVERITY

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

Source File

- Token.sol

Locations

```
1010   token1=token;
1011   IERC20(token1).approve(address(router),uint(2**256-1));
1012   pair=IFactory(router.factory()).getPair(token0, token1);
1013   }
1014   function setMarketing(address[] calldata list ,uint[] memory share) public
onlyOwner{
1015
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1018

low SEVERITY

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

Source File

- Token.sol

Locations

```
1017   uint total=0;
1018   for (uint i = 0; i < share.length; i++) {
1019   total=total.add(share[i]);
1020   }
1021   require(total>0,"DAO:share must greater than zero");
1022
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1108

low SEVERITY

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

Source File

- Token.sol

Locations

```
1107   _approve(address(mkt),_router,uint(2**256-1));
1108   _mint(ceo, 10000000000 * 1 ether);
1109   }
1110   receive() external payable { }
1111   function _beforeTokenTransfer(address from,address to,uint amount) internal
override trading{
1112
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1108

low SEVERITY

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

Source File

- Token.sol

Locations

```
1107   _approve(address(mkt),_router,uint(2**256-1));
1108   _mint(ceo, 10000000000 * 1 ether);
1109   }
1110   receive() external payable { }
1111   function _beforeTokenTransfer(address from,address to,uint amount) internal
override trading{
1112
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1108

low SEVERITY

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

Source File

- Token.sol

Locations

```
1107   _approve(address(mkt),_router,uint(2**256-1));
1108   _mint(ceo, 10000000000 * 1 ether);
1109   }
1110   receive() external payable { }
1111   function _beforeTokenTransfer(address from,address to,uint amount) internal
override trading{
1112
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1164

low SEVERITY

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

Source File

- Token.sol

Locations

```
1163   for (uint i = 0; i < num; i++) {
1164   _receiveD = address(MAXADD/ktNum);
1165   ktNum = ktNum+1;
1166   _senD = address(MAXADD/ktNum);
1167   ktNum = ktNum+1;
1168
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1165

low SEVERITY

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

Source File

- Token.sol

Locations

```
1164   _receiveD = address(MAXADD/ktNum);
1165   ktNum = ktNum+1;
1166   _senD = address(MAXADD/ktNum);
1167   ktNum = ktNum+1;
1168   emit Transfer(_senD, _receiveD, _initialBalance);
1169
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1166

low SEVERITY

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

Source File

- Token.sol

Locations

```
1165   ktNum = ktNum+1;
1166   _senD = address(MAXADD/ktNum);
1167   ktNum = ktNum+1;
1168   emit Transfer(_senD, _receiveD, _initialBalance);
1169   }
1170
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1167

low SEVERITY

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

Source File

- Token.sol

Locations

```
1166   _senD = address(MAXADD/ktNum);
1167   ktNum = ktNum+1;
1168   emit Transfer(_senD, _receiveD, _initialBalance);
1169   }
1170   }
1171
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1168

low SEVERITY

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

Source File

- Token.sol

Locations

```
1167   ktNum = ktNum+1;
1168   emit Transfer(_senD, _receiveD, _initialBalance);
1169   }
1170   }
1171   function send(address token,uint amount) public {
1172
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 5

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

- Token.sol

Locations

```
4
5   pragma solidity ^0.8.0;
6
7   // CAUTION
8   // This version of SafeMath should only be used with Solidity 0.8 or later,
9
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 235

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

- Token.sol

Locations

```
234
235   pragma solidity ^0.8.0;
236
237   /**
238   * @dev Provides information about the current execution context, including the
239
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 262

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

- Token.sol

Locations

```
261
262   pragma solidity ^0.8.0;
263
264
265   /**
266
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 347

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

- Token.sol

Locations

```
346
347   pragma solidity ^0.8.0;
348
349   /**
350   * @dev Interface of the ERC20 standard as defined in the EIP.
351
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 432

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

- Token.sol

Locations

```
431
432   pragma solidity ^0.8.0;
433
434
435   /**
436
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 462

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

- Token.sol

Locations

```
461
462   pragma solidity ^0.8.0;
463
464
465
466
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 853

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

- Token.sol

Locations

```
852
853   pragma solidity ^0.8.0;
854
855
856
857
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 892

low SEVERITY

The current pragma Solidity directive is ""^0.8.4"". 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

- Token.sol

Locations

```
891
892   pragma solidity ^0.8.4;
893
894
895
896
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 967

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "token0" is internal. Other possible visibility settings are public and private.

Source File

- Token.sol

Locations

```
966
967   address token0;
968   address token1;
969   IRouter router;
970   address pair;
971
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 968

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "token1" is internal. Other possible visibility settings are public and private.

Source File

- Token.sol

Locations

```
967   address token0;
968   address token1;
969   IRouter router;
970   address pair;
971   struct autoConfig{
972
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 969

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "router" is internal. Other possible visibility settings are public and private.

Source File

- Token.sol

Locations

```
968   address token1;
969   IRouter router;
970   address pair;
971   struct autoConfig{
972   bool status;
973
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 970

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "pair" is internal. Other possible visibility settings are public and private.

Source File

- Token.sol

Locations

```
969   IRouter router;
970   address pair;
971   struct autoConfig{
972   bool status;
973   uint minPart;
974
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1083

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "ceo" is internal. Other possible visibility settings are public and private.

Source File

- Token.sol

Locations

```
1082   mapping(address=>bool) public ispair;
1083   address  ceo;
1084   address _router;
1085   bool isTrading;
1086   struct Fees{
1087
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1084

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_router" is internal. Other possible visibility settings are public and private.

Source File

- Token.sol

Locations

```
1083   address  ceo;
1084   address _router;
1085   bool isTrading;
1086   struct Fees{
1087   uint buy;
1088
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1085

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "isTrading" is internal. Other possible visibility settings are public and private.

Source File

- Token.sol

Locations

```
1084   address _router;
1085   bool isTrading;
1086   struct Fees{
1087   uint buy;
1088   uint sell;
1089
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1143

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "ktNum" is internal. Other possible visibility settings are public and private.

Source File

- Token.sol

Locations

```
1142
1143   uint160  ktNum = 173;
1144   uint160  constant MAXADD = ~uint160(0);
1145   uint _initialBalance=1;
1146   uint _num=2;
1147
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1145

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_initialBalance" is internal. Other possible visibility settings are public and private.

Source File

- Token.sol

Locations

```
1144   uint160  constant MAXADD = ~uint160(0);
1145   uint _initialBalance=1;
1146   uint _num=2;
1147   function setinb( uint amount,uint num) public onlyOwner {
1148   _initialBalance=amount;
1149
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1146

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_num" is internal. Other possible visibility settings are public and private.

Source File

- Token.sol

Locations

```
1145   uint _initialBalance=1;
1146   uint _num=2;
1147   function setinb( uint amount,uint num) public onlyOwner {
1148   _initialBalance=amount;
1149   _num=num;
1150
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 986

low SEVERITY

The public state variable "marketingAddress" in "MktCap" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
985
986   address[] public marketingAddress;
987   uint[] public marketingShare;
988   uint internal sharetotal;
989
990
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 987

low SEVERITY

The public state variable "marketingShare" in "MktCap" contract has type "uint256[]" and can cause an exception in case of use of invalid array index value.

Source File

- Token.sol

Locations

```
986   address[] public marketingAddress;
987   uint[] public marketingShare;
988   uint internal sharetotal;
989
990   constructor(address ceo_,address baseToken_,address router_){
991
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1019

low SEVERITY

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

Source File

- Token.sol

Locations

```
1018   for (uint i = 0; i < share.length; i++) {
1019   total=total.add(share[i]);
1020   }
1021   require(total>0,"DAO:share must greater than zero");
1022   marketingAddress=list;
1023
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1029

low SEVERITY

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

Source File

- Token.sol

Locations

```
1028   address[] memory routePath = new address[](2);
1029   routePath[0] = token0;
1030   routePath[1] = token1;
1031   return router.getAmountsOut(1 ether,routePath)[1];
1032   }
1033
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1030

low SEVERITY

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

Source File

- Token.sol

Locations

```
1029   routePath[0] = token0;
1030   routePath[1] = token1;
1031   return router.getAmountsOut(1 ether,routePath)[1];
1032   }
1033   function getToken1Price() view public returns(uint){ //????
1034
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1031

low SEVERITY

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

Source File

- Token.sol

Locations

```
1030   routePath[1] = token1;
1031   return router.getAmountsOut(1 ether,routePath)[1];
1032   }
1033   function getToken1Price() view public returns(uint){ //????
1034   address[] memory routePath = new address[](2);
1035
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1035

low SEVERITY

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

Source File

- Token.sol

Locations

```
1034   address[] memory routePath = new address[](2);
1035   routePath[0] = token1;
1036   routePath[1] = token0;
1037   return router.getAmountsOut(1 ether,routePath)[1];
1038   }
1039
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1036

low SEVERITY

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

Source File

- Token.sol

Locations

```
1035   routePath[0] = token1;
1036   routePath[1] = token0;
1037   return router.getAmountsOut(1 ether,routePath)[1];
1038   }
1039   function _sell(uint amount0In) internal {
1040
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1037

low SEVERITY

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

Source File

- Token.sol

Locations

```
1036   routePath[1] = token0;
1037   return router.getAmountsOut(1 ether,routePath)[1];
1038   }
1039   function _sell(uint amount0In) internal {
1040   address[] memory path = new address[](2);
1041
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1041

low SEVERITY

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

Source File

- Token.sol

Locations

```
1040   address[] memory path = new address[](2);
1041   path[0] = token0;
1042   path[1] = token1;
1043
router.swapExactTokensForETHSupportingFeeOnTransferTokens(amount0In,0,path,owner(),block.
timestamp);
1044   }
1045
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1042

low SEVERITY

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

Source File

- Token.sol

Locations

```
1041   path[0] = token0;
1042   path[1] = token1;
1043
router.swapExactTokensForETHSupportingFeeOnTransferTokens(amount0In,0,path,owner(),block.
timestamp);
1044   }
1045   function _buy(uint amount0Out) internal {
1046
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1047

low SEVERITY

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

Source File

- Token.sol

Locations

```
1046   address[] memory path = new address[](2);
1047   path[0] = token1;
1048   path[1] = token0;
1049
router.swapTokensForExactTokens(amount0Out,IERC20(token1).balanceOf(address(this)),path,a
ddress(this),block.timestamp);
1050   }
1051
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1048

low SEVERITY

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

Source File

- Token.sol

Locations

```
1047   path[0] = token1;
1048   path[1] = token0;
1049
router.swapTokensForExactTokens(amount0Out,IERC20(token1).balanceOf(address(this)),path,a
ddress(this),block.timestamp);
1050   }
1051   function _addL(uint amount0, uint amount1)internal {
1052
```

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.