SOOMA

Smart Contract Audit Report

SYSFIXED

06 Nov 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
SOOMA	SOA	Binance Smart Chain

Addresses

Contract address	0x9a30B72A793a716920cF985Fd9Ae3a3cb278C3a2
Contract deployer address	0x9C0B62E8535849857933efdAB7772e3E468357ee

Project Website

https://sooma.io/

Codebase

https://bscscan.com/address/0x9a30B72A793a716920cF985Fd9Ae3a3cb278C3a2#code

SUMMARY

SOOMA (SOA) is a multipurpose deflation token on the bsc platform. After the launch of the project, 2% of the total amount will be burned every month according to Tokenomics.

Contract Summary

Documentation Quality

SOOMA 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 SOOMA 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 629.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 25, 33, 39, 40, 47, 53, 57, 60, 63, 66, 69, 74, 80, 86, 225, 604, 605, 632, 634, 827, 945, 966, 974 and 829.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 9.
- SWC-110 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 828, 829, 947, 948, 950, 951, 1078 and 1079.

CONCLUSION

We have audited the SOOMA project which has released on November 2022 to discover issues and identify potential security vulnerabilities in SOOMA Project. This process is used to find technical issues and security loopholes that find some common issues in the code.

The security audit report produced satisfactory results with low-risk issues.

The most common issue found in writing code on contracts that do not pose a big risk, writing on contracts is close to the standard of writing contracts in general. The low-level issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set and out of bounds array access which the index access expression can cause an exception in case of 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
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
Assert Violation	SWC-110	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 Caller	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
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
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

SMART CONTRACT ANALYSIS

Started	Saturday Nov 05 2022 12:16:57 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Nov 06 2022 11:20:13 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	SOOMA.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT 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-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

- SOOMA.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 33

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
32   if (b > a) return (false, 0);
33   return (true, a - b);
34   }
35   }
36   function tryMul(uint256 a, uint256 b) internal pure returns (bool, uint256) {
37
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 39

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
38   if (a == 0) return (true, 0);
39   uint256 c = a * b;
40   if (c / a != b) return (false, 0);
41   return (true, c);
42   }
43
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 40

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
39   uint256 c = a * b;
40   if (c / a != b) return (false, 0);
41   return (true, c);
42   }
43   }
44
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 47

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
46   if (b == 0) return (false, 0);
47   return (true, a / b);
48   }
49   }
50   function tryMod(uint256 a, uint256 b) internal pure returns (bool, uint256) {
51
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 53

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
52   if (b == 0) return (false, 0);
53   return (true, a % b);
54   }
55   }
56   function add(uint256 a, uint256 b) internal pure returns (uint256) {
57
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 57

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
56   function add(uint256 a, uint256 b) internal pure returns (uint256) {
57   return a + b;
58   }
59   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
60   return a - b;
61
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 60

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
59   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
60   return a - b;
61   }
62   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
63   return a * b;
64
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 63

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
62   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
63   return a * b;
64   }
65   function div(uint256 a, uint256 b) internal pure returns (uint256) {
66   return a / b;
67
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 66

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
65   function div(uint256 a, uint256 b) internal pure returns (uint256) {
66   return a / b;
67   }
68   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
69   return a % b;
70
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 69

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
68   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
69   return a % b;
70   }
71   function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
72   unchecked {
73
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 74

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
73   require(b <= a, errorMessage);
74   return a - b;
75   }
76   }
77   function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
78
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 80

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
79   require(b > 0, errorMessage);
80   return a / b;
81   }
82   }
83   function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
84
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 86

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
85   require(b > 0, errorMessage);
86   return a % b;
87   }
88   }
89   }
90
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 225

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
224   _owner = address(0);
225   _lockTime = block.timestamp + time;
226   emit OwnershipTransferred(_owner, address(0));
227   }
228
229
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 604

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
603   uint256 private constant MAX = ~uint256(0);
604   uint256 private _tTotal = 10000000000 * 10**18;
605   uint256 private _rTotal = (MAX - (MAX % _tTotal));
606   uint256 private _tFeeTotal;
607
608
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 605

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
604   uint256 private _tTotal = 10000000000 * 10**18;
605   uint256 private _rTotal = (MAX - (MAX % _tTotal));
606   uint256 private _tFeeTotal;
607
608   string private _name = "SOOMA";
609
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 632

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
631
632   uint256 public _maxTxAmount = 1000000* 10**18;
633
634   uint256 private numTokensSellToAddToLiquidity = 1000000* 10**18;
635
636
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 634

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
633
634   uint256 private numTokensSellToAddToLiquidity = 1000000* 10**18;
635
636   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
637   event SwapAndLiquifyEnabledUpdated(bool enabled);
638
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 827

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
826   require(_isExcluded[account], "Account is already excluded");
827   for (uint256 i = 0; i < _excluded.length; i++) {
828   if (_excluded[i] == account) {
829   _excluded[i] = _excluded[_excluded.length - 1];
830   _tOwned[account] = 0;
831
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 829

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
828   if (_excluded[i] == account) {
829   _excluded[i] = _excluded[_excluded.length - 1];
830   _tOwned[account] = 0;
831   _isExcluded[account] = false;
832   _excluded.pop();
833
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 945

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
944   uint256 tSupply = _tTotal;
945   for (uint256 i = 0; i < _excluded.length; i++) {
946   if (
947   _rOwned[_excluded[i]] > rSupply ||
948   _tOwned[_excluded[i]] > tSupply
949
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 966

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
965   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
966   return _amount.mul(_taxFee).div(10**2);
967   }
968
969   function calculateLiquidityFee(uint256 _amount)
970
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 974

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
973   {
974   return _amount.mul(_liquidityFee).div(10**2);
975   }
976
977   function removeAllFee() private {
978
```

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

LINE 829

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
828   if (_excluded[i] == account) {
829   _excluded[i] = _excluded[_excluded.length - 1];
830   _tOwned[account] = 0;
831   _isExcluded[account] = false;
832   _excluded.pop();
833
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 9

low SEVERITY

The current pragma Solidity directive is ""^0.8.7"". 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

- SOOMA.sol

Locations

```
8
9   pragma solidity ^0.8.7;
10
11   interface IBEP20 {
12   function totalSupply() external view returns (uint256);
13
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 629

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
628
629   bool inSwapAndLiquify;
630   bool public swapAndLiquifyEnabled = false;
631
632   uint256 public _maxTxAmount = 1000000* 10**18;
633
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 828

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
827   for (uint256 i = 0; i < _excluded.length; i++) {
828   if (_excluded[i] == account) {
829   _excluded[i] = _excluded[_excluded.length - 1];
830   _tOwned[account] = 0;
831   _isExcluded[account] = false;
832
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 829

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
828   if (_excluded[i] == account) {
829   _excluded[i] = _excluded[_excluded.length - 1];
830   _tOwned[account] = 0;
831   _isExcluded[account] = false;
832   _excluded.pop();
833
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 947

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
946   if (
947   _rOwned[_excluded[i]] > rSupply ||
948   _tOwned[_excluded[i]] > tSupply
949   ) return (_rTotal, _tTotal);
950   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
951
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 948

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
947   _rOwned[_excluded[i]] > rSupply ||
948   _tOwned[_excluded[i]] > tSupply
949   ) return (_rTotal, _tTotal);
950   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
951   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
952
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 950

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
949   ) return (_rTotal, _tTotal);
950   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
951   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
952   }
953   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
954
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 951

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
950   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
951   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
952   }
953   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
954   return (rSupply, tSupply);
955
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1078

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
1077   address[] memory path = new address[](2);
1078   path[0] = address(this);
1079   path[1] = uniswapV2Router.WETH();
1080
1081   _approve(address(this), address(uniswapV2Router), tokenAmount);
1082
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1079

low SEVERITY

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

Source File

- SOOMA.sol

Locations

```
1078   path[0] = address(this);
1079   path[1] = uniswapV2Router.WETH();
1080
1081   _approve(address(this), address(uniswapV2Router), tokenAmount);
1082
1083
```

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.