LoopNetwork

Smart Contract Audit Report

SYSFIXED

01 Feb 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
LoopNetwork	LOOP	Binance Smart Chain

Addresses

Contract address	0xce186ad6430e2fe494a22c9edbd4c68794a28b35
Contract deployer address	0x954137f063c821cd8247Ab2E1235b4548B8ac8D5

Project Website

https://www.getloop.network/

Codebase

https://bscscan.com/address/0xce186ad6430e2fe494a22c9edbd4c68794a28b35#code

SUMMARY

A cryptocurrency system that supports smart contracts without the scalability and privacy limitations of earlier systems like Ethereum. Loop network, like Ethereum, allows parties to create smart contracts using code to specify the behavior of the virtual machine (VM) that executes the contract's function.Loop Network strives to solve scalability and usability issues, without compromising decentralization, and leverages the existing developer community and ecosystem. It is an off-chain/external scaling solution for existing platforms to provide scalability and superior user experience for DApps/user features.

Contract Summary

Documentation Quality

LoopNetwork 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 LoopNetwork 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 36, 40, 44, 48, 54, 61, 413, 413, 598, 653, 658, 671, 676, 733, 733, 738, 738, 798, 900, 900, 974, 994 and 994.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 17.
- 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 656, 657, 674, 675, 909 and 910.
- SWC-131 SWC-135 | It is recommended to remove all unused variables from the code base on lines 898.

CONCLUSION

We have audited the LoopNetwork project released on January 2022 to discover issues and identify potential security vulnerabilities in LoopNetwork 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 LoopNetwork 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, usage of equality comparison instead of assignment, 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. The current pragma Solidity directive is "^0.8.7". Specifying a fixed compiler version is recommended 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. Using equality comparison instead of assignment, this equality comparison has no effect.

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 Jan 31 2022 08:14:25 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Feb 01 2022 16:26:47 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	LoopNetwork.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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-135	USAGE OF EQUALITY COMPARISON INSTEAD OF ASSIGNMENT	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 36

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
35   function add(uint256 a, uint256 b) internal pure returns (uint256) {
36   return a + b;
37   }
38
39   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
40
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 40

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
39   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
40   return a - b;
41   }
42
43   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
44
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 44

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
43   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
44   return a * b;
45   }
46
47   function div(uint256 a, uint256 b) internal pure returns (uint256) {
48
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 48

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
47   function div(uint256 a, uint256 b) internal pure returns (uint256) {
48   return a / b;
49   }
50
51   function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
52
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 54

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
53   require(b <= a, errorMessage);
54   return a - b;
55   }
56   }
57
58
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 61

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
60   require(b > 0, errorMessage);
61   return a / b;
62   }
63   }
64
65
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 413

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
412   uint8 private _decimals = 18;
413   uint256 private _tTotal = 200000000 * 10**18;
414   uint256 private _tFeeTotal;
415
416   // Counter for liquify trigger
417
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 413

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
412   uint8 private _decimals = 18;
413   uint256 private _tTotal = 200000000 * 10**18;
414   uint256 private _tFeeTotal;
415
416   // Counter for liquify trigger
417
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 598

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
597
598   require((Buy_Fee + Sell_Fee) <= maxPossibleFee, "Fee is too high!");
599   _sellFee = Sell_Fee;
600   _buyFee = Buy_Fee;
601
602
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 653

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
652
653   for (uint256 i; i < addresses.length; ++i) {
654   if(gasUsed < gasleft()) {
655   startGas = gasleft();
656   if(!_isBlacklisted[addresses[i]]){
657
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 658

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
657   _isBlacklisted[addresses[i]] = true;}
658   gasUsed = startGas - gasleft();
659   }
660   }
661   }
662
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 671

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
670
671   for (uint256 i; i < addresses.length; ++i) {
672   if(gasUsed < gasleft()) {
673   startGas = gasleft();
674   if(_isBlacklisted[addresses[i]]){
675
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 676

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
675   _isBlacklisted[addresses[i]] = false;}
676   gasUsed = startGas - gasleft();
677   }
678   }
679   }
680
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 733

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
732   function set_Max_Transaction_Percent(uint256 maxTxPercent_x100) external
onlyOwner() {
733   _maxTxAmount = _tTotal*maxTxPercent_x100/10000;
734   }
735
736   // Set the maximum wallet holding (percent of total supply)
737
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 733

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
732   function set_Max_Transaction_Percent(uint256 maxTxPercent_x100) external
onlyOwner() {
733   _maxTxAmount = _tTotal*maxTxPercent_x100/10000;
734   }
735
736   // Set the maximum wallet holding (percent of total supply)
737
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 738

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
737   function set_Max_Wallet_Percent(uint256 maxWallPercent_x100) external onlyOwner() {
738   _maxWalletToken = _tTotal*maxWallPercent_x100/10000;
739   }
740
741
742
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 738

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
737   function set_Max_Wallet_Percent(uint256 maxWallPercent_x100) external onlyOwner() {
738   _maxWalletToken = _tTotal*maxWallPercent_x100/10000;
739   }
740
741
742
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 798

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
797   uint256 heldTokens = balanceOf(to);
798   require((heldTokens + amount) <= _maxWalletToken,"You are trying to buy too many
tokens. You have reached the limit for one wallet.");}
799
800
801   // Limit the maximum number of tokens that can be bought or sold in one transaction
802
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 900

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
899   uint256 tokensOnContract = balanceOf(address(this));
900   uint256 sendTokens = tokensOnContract*percent_Of_Tokens_To_Process/100;
901   swapAndLiquify(sendTokens);
902   }
903
904
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 900

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
899   uint256 tokensOnContract = balanceOf(address(this));
900   uint256 sendTokens = tokensOnContract*percent_Of_Tokens_To_Process/100;
901   swapAndLiquify(sendTokens);
902   }
903
904
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 974

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
973   } else {
974   txCount++;
975   }
976   _transferTokens(sender, recipient, amount);
977
978
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 994

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
993   function _getValues(uint256 tAmount) private view returns (uint256, uint256) {
994   uint256 tDev = tAmount*_TotalFee/100;
995   uint256 tTransferAmount = tAmount.sub(tDev);
996   return (tTransferAmount, tDev);
997   }
998
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 994

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
993   function _getValues(uint256 tAmount) private view returns (uint256, uint256) {
994   uint256 tDev = tAmount*_TotalFee/100;
995   uint256 tTransferAmount = tAmount.sub(tDev);
996   return (tTransferAmount, tDev);
997   }
998
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 17

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

- LoopNetwork.sol

Locations

```
16
17   pragma solidity ^0.8.7;
18
19
20   interface IERC20 {
21
```

SWC-135 | USAGE OF EQUALITY COMPARISON INSTEAD OF ASSIGNMENT

LINE 898

low SEVERITY

This equality comparison doesn't have any effect. Did you mean to do assignment instead?

Source File

- LoopNetwork.sol

Locations

```
897   require(!inSwapAndLiquify, "Currently processing, try later.");
898   if (percent_Of_Tokens_To_Process > 100){percent_Of_Tokens_To_Process == 100;}
899   uint256 tokensOnContract = balanceOf(address(this));
900   uint256 sendTokens = tokensOnContract*percent_Of_Tokens_To_Process/100;
901   swapAndLiquify(sendTokens);
902
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 656

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
655   startGas = gasleft();
656   if(!_isBlacklisted[addresses[i]]){
657   _isBlacklisted[addresses[i]] = true;}
658   gasUsed = startGas - gasleft();
659   }
660
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 657

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
656   if(!_isBlacklisted[addresses[i]]){
657   _isBlacklisted[addresses[i]] = true;}
658   gasUsed = startGas - gasleft();
659   }
660   }
661
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 674

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
673   startGas = gasleft();
674   if(_isBlacklisted[addresses[i]]){
675   _isBlacklisted[addresses[i]] = false;}
676   gasUsed = startGas - gasleft();
677   }
678
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 675

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
674   if(_isBlacklisted[addresses[i]]){
675   _isBlacklisted[addresses[i]] = false;}
676   gasUsed = startGas - gasleft();
677   }
678   }
679
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 909

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
908   address[] memory path = new address[](2);
909   path[0] = address(this);
910   path[1] = uniswapV2Router.WETH();
911   _approve(address(this), address(uniswapV2Router), tokenAmount);
912   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
913
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 910

low SEVERITY

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

Source File

- LoopNetwork.sol

Locations

```
909   path[0] = address(this);
910   path[1] = uniswapV2Router.WETH();
911   _approve(address(this), address(uniswapV2Router), tokenAmount);
912   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
913   tokenAmount,
914
```

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.