LaunchVerse

Smart Contract Audit Report

SYSFIXED

04 May 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
LaunchVerse	XLV	BSC

Addresses

Contract address	0x2304AE9aF71a5AE1b92f0091aC3cafF105C67766
Contract deployer address	0x7b7943394B172Be0Af3961E01c33AB4bcF195d77

Project Website

https://launchverse.space/

Codebase

https://bscscan.com/address/0x2304AE9aF71a5AE1b92f0091aC3cafF105C67766#code

SUMMARY

Launchverse is a Data Aggregator - a Crypto Investing Platform. Web3 investing solutions with multi earning ways. XLV is the first cryptocurrency that reflects promising tokens at their IDO stages. Hold XLV & Earn many other tokens from best performing Launchpads. Unique multiplied version of passive income. Launchverse has Data Aggregation, NFT Exhibition, Staking Seed Pad, and Dapps Ready. Launchverse is no private sale and no unlocked token. Launchverse also has CMC listed, CG fast track, LV tracking tool, huge marketing, partnerships, and CEX listing.

Contract Summary

Documentation Quality

LaunchVerse provides a document with a good standard of solidity base code.

- The technical description is provided clearly and structured.

Code Quality

The Overall quality of the basecode is GOOD with only 2 low-risk issues

- Standart solidity basecode and rules are already followed with LaunchVerse Project .

Test Coverage

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

Audit Findings Summary

- SWC-101 | Arithmetic operation Issues discovered on lines 61, 79, 98, 99, 116, 132, 147, 161, 175, 189, 205, 228, 255, 281, 294, 298, 310, 317, 326, 754, 755, 755, 771, 999, 1001, 1130, 1211, 1001.
- SWC-103 | A floating pragma is set on lines 6.
- SWC-108 | State variable visibility is not set on lines 776. It is best practice to set the visibility of state variables explicitly. The default visibility for "protections" is internal. Other possible visibility settings are public and private.
- SWC-110 | Out of bounds array access on lines 1000, 1001, 1132, 1133, 1135, 1136, 1301, 1302.

CONCLUSION

We have audited the LaunchVerse Coin which has been released to discover issues and identify potential security vulnerabilities in LaunchVerseProject. This process is used to find bugs, technical issues, and security loopholes that find some common issues in the code.

The security audit report produced satisfactory results with a low-risk issue on the contract project.

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. Some of the low issues that were found were asserted violation and a floating pragma set.

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
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external 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	Tue May 3 2023 07:31:58 GMT+0000 (Coordinated Universal Time)
Finished	Wed May 4 2023 08:31:58 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	XLV.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	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 61

low SEVERITY

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

Source File

- XLV.sol

Locations

```
60   unchecked {
61   uint256 c = a + b;
62   if (c < a) return (false, 0);
63   return (true, c);
64   }
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 79

low SEVERITY

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

Source File

- XLV.sol

Locations

```
78   if (b > a) return (false, 0);
79   return (true, a - b);
80   }
81   }
82
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 98

low SEVERITY

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

Source File

- XLV.sol

Locations

```
97   if (a == 0) return (true, 0);
98   uint256 c = a * b;
99   if (c / a != b) return (false, 0);
100   return (true, c);
101   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 99

low SEVERITY

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

Source File

- XLV.sol

Locations

```
98   uint256 c = a * b;
99   if (c / a != b) return (false, 0);
100   return (true, c);
101   }
102   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 116

low SEVERITY

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

Source File

- XLV.sol

Locations

```
115   if (b == 0) return (false, 0);
116   return (true, a / b);
117   }
118   }
119
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 132

low SEVERITY

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

Source File

- XLV.sol

Locations

```
if (b == 0) return (false, 0);
return (true, a % b);
}
}

```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 147

low SEVERITY

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

Source File

- XLV.sol

Locations

```
146   function add(uint256 a, uint256 b) internal pure returns (uint256) {
147   return a + b;
148   }
149
150   /**
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 161

low SEVERITY

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

Source File

- XLV.sol

Locations

```
160   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
161   return a - b;
162   }
163
164   /**
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 175

low SEVERITY

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

Source File

- XLV.sol

Locations

```
174   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
175   return a * b;
176   }
177
178   /**
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 189

low SEVERITY

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

Source File

- XLV.sol

Locations

```
188   function div(uint256 a, uint256 b) internal pure returns (uint256) {
189   return a / b;
190   }
191
192   /**
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 205

low SEVERITY

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

Source File

- XLV.sol

Locations

```
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
return a % b;
}

/**
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 228

low SEVERITY

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

Source File

- XLV.sol

Locations

```
227   require(b <= a, errorMessage);
228   return a - b;
229   }
230   }
231
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 255

low SEVERITY

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

Source File

- XLV.sol

Locations

```
254   require(b > 0, errorMessage);
255   return a / b;
256   }
257   }
258
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 281

low SEVERITY

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

Source File

- XLV.sol

Locations

```
280   require(b > 0, errorMessage);
281   return a % b;
282   }
283   }
284   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 294

low SEVERITY

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

Source File

- XLV.sol

Locations

```
293   function mul(int256 a, int256 b) internal pure returns (int256) {
294   int256 c = a * b;
295
296   // Detect overflow when multiplying MIN_INT256 with -1
297   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 298

low SEVERITY

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

Source File

- XLV.sol

Locations

```
297   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
298   require((b == 0) || (c / b == a));
299   return c;
300   }
301
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 310

low SEVERITY

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

Source File

- XLV.sol

Locations

```
309   // Solidity already throws when dividing by 0.
310   return a / b;
311   }
312
313   /**
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 317

low SEVERITY

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

Source File

- XLV.sol

Locations

```
316   function sub(int256 a, int256 b) internal pure returns (int256) {
317   int256 c = a - b;
318   require((b >= 0 && c <= a) || (b < 0 && c > a));
319   return c;
320   }
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 326

low SEVERITY

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

Source File

- XLV.sol

Locations

```
325   function add(int256 a, int256 b) internal pure returns (int256) {
326   int256 c = a + b;
327   require((b >= 0 && c >= a) || (b < 0 && c < a));
328   return c;
329   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 754

low SEVERITY

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

Source File

- XLV.sol

Locations

```
753   uint256 private constant MAX = ~uint256(0);
754   uint256 private _tTotal = 1000 * 10**6 * 10**9;
755   uint256 private _rTotal = (MAX - (MAX % _tTotal));
756   uint256 private _tFeeTotal;
757
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 755

low SEVERITY

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

Source File

- XLV.sol

Locations

```
754   uint256 private _tTotal = 1000 * 10**6 * 10**9;
755   uint256 private _rTotal = (MAX - (MAX % _tTotal));
756   uint256 private _tFeeTotal;
757
758   string private _name = "LaunchVerse";
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 755

low SEVERITY

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

Source File

- XLV.sol

Locations

```
754   uint256 private _tTotal = 1000 * 10**6 * 10**9;
755   uint256 private _rTotal = (MAX - (MAX % _tTotal));
756   uint256 private _tFeeTotal;
757
758   string private _name = "LaunchVerse";
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 771

low SEVERITY

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

Source File

- XLV.sol

Locations

```
770   uint256 public _maxWalletToken = _tTotal.mul(2).div(100);
771   uint256 public _swapTokensAt = 100 * 10**5 * 10**9;
772
773   bool public tradeEnable = false;
774   // auto liquidity
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 999

low SEVERITY

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

Source File

- XLV.sol

Locations

```
998
999   for (uint256 i = 0; i < _excluded.length; i++) {
1000   if (_excluded[i] == account) {
1001   _excluded[i] = _excluded[_excluded.length - 1];
1002   _tOwned[account] = 0;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1001

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1000   if (_excluded[i] == account) {
1001   _excluded[i] = _excluded[_excluded.length - 1];
1002   _tOwned[account] = 0;
1003   _isExcluded[account] = false;
1004   _excluded.pop();
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1130

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1129   uint256 tSupply = _tTotal;
1130   for (uint256 i = 0; i < _excluded.length; i++) {
1131   if (
1132   _rOwned[_excluded[i]] > rSupply ||
1133   _tOwned[_excluded[i]] > tSupply
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1211

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1210   require(
1211   reciverTokenBalance + amount <= _maxWalletToken,
1212   "Max Wallet Token Exceeded."
1213   );
1214   }
```

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

LINE 1001

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1000   if (_excluded[i] == account) {
1001   _excluded[i] = _excluded[_excluded.length - 1];
1002   _tOwned[account] = 0;
1003   _isExcluded[account] = false;
1004   _excluded.pop();
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

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

- XLV.sol

Locations

```
5   // SPDX-License-Identifier: Unlicensed
6   pragma solidity ^0.8.4;
7
8   abstract contract Context {
9   function _msgSender() internal view virtual returns (address) {
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 776

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

- XLV.sol

Locations

```
775   bool public _swapAndLiquifyEnabled = true;
776   bool _inSwapAndLiquify;
777   IUniswapV2Router02 public _uniswapV2Router;
778   address public _uniswapV2Pair;
779   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1000

low SEVERITY

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

Source File

- XLV.sol

Locations

```
999   for (uint256 i = 0; i < _excluded.length; i++) {
1000   if (_excluded[i] == account) {
1001   _excluded[i] = _excluded[_excluded.length - 1];
1002   _tOwned[account] = 0;
1003   _isExcluded[account] = false;
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1001

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1000   if (_excluded[i] == account) {
1001   _excluded[i] = _excluded[_excluded.length - 1];
1002   _tOwned[account] = 0;
1003   _isExcluded[account] = false;
1004   _excluded.pop();
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1132

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1131   if (
1132   _rOwned[_excluded[i]] > rSupply ||
1133   _tOwned[_excluded[i]] > tSupply
1134   ) return (_rTotal, _tTotal);
1135   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1133

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1132   _rOwned[_excluded[i]] > rSupply ||
1133   _tOwned[_excluded[i]] > tSupply
1134   ) return (_rTotal, _tTotal);
1135   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1136   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1135

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1134   ) return (_rTotal, _tTotal);
1135   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1136   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1137   }
1138   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1136

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1135    rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1136    tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1137    }
1138    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1139    return (rSupply, tSupply);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1301

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1300   address[] memory path = new address[](2);
1301   path[0] = address(this);
1302   path[1] = _uniswapV2Router.WETH();
1303
1304   _approve(address(this), address(_uniswapV2Router), tokenAmount);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1302

low SEVERITY

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

Source File

- XLV.sol

Locations

```
1301   path[0] = address(this);
1302   path[1] = _uniswapV2Router.WETH();
1303
1304   _approve(address(this), address(_uniswapV2Router), tokenAmount);
1305
```

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.