Libera.Financial

Smart Contract Audit Report

SYSFIXED

08 May 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Libera.Financial	LIBERA	Binance Smart Chain

Addresses

Contract address	0x3a806a3315e35b3f5f46111adb6e2baf4b14a70d
Contract deployer address	0x17Eca3A2aFDff586e5D66c992554ea29fF9Cb11D

Project Website

https://bitgert.com/

Codebase

https://bscscan.com/address/0x3a806a3315e35b3f5f46111adb6e2baf4b14a70d#code

SUMMARY

The Bitgert team has brought together for this project a relevant number of professionals from a range of technological fields to build one of the most disruptive blockchain ecosystems to date. We have a team that is specialized in building crypto projects, which includes software engineers with years of experience in building blockchain-based products. Our team also includes crypto/blockchain researchers, marketers, and other professionals with years of experience who have been involved in some of the most successful crypto projects in the market. This explains why the project's delivery has so far been second to none in the industry.

Contract Summary

Documentation Quality

Libera.Financial 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 Libera.Financial 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 85.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 11, 17, 24, 25, 30, 34, 37, 43, 47, 48, 53, 57, 61, 65, 69, 346, 346, 346, 346, 378, 378, 378, 378, 380, 380, 381, 381, 395, 395, 454, 455, 455, 461, 461, 468, 489, 495, 548, 548, 551, 551, 565, 567, 578, 624, 624, 624, 624, 656, 790, 859, 868, 912, 926, 945, 567 and 859.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 7.
- 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 470, 473, 479, 566, 567, 567, 657, 813, 814, 815, 818, 819, 836, 837, 840, 841 and 842.

CONCLUSION

We have audited the Libera.Financial project released on May 2022 to discover issues and identify potential security vulnerabilities in Libera.Financial 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 Libera.Financial smart contract codes 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, 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 ""^0.8.4"". 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. It is best practice to set the visibility of state variables explicitly. The default visibility for "owner" is internal. Other possible visibility settings are public and private.

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	Saturday May 07 2022 12:38:49 GMT+0000 (Coordinated Universal Time)
Finished	Sunday May 08 2022 17:20:14 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	LiberaToken.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-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

SYSFIXED

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-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 11

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
10   function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
11   uint256 c = a + b;
12   if (c < a) return (false, 0);
13   return (true, c);
14   }
15
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 17

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
16   if (b > a) return (false, 0);
17   return (true, a - b);
18   }
19   function tryMul(uint256 a, uint256 b) internal pure returns (bool, uint256) {
20   // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
21
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 24

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
23   if (a == 0) return (true, 0);
24   uint256 c = a * b;
25   if (c / a != b) return (false, 0);
26   return (true, c);
27   }
28
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 25

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
24   uint256 c = a * b;
25   if (c / a != b) return (false, 0);
26   return (true, c);
27   }
28   function tryDiv(uint256 a, uint256 b) internal pure returns (bool, uint256) {
29
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 30

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
29   if (b == 0) return (false, 0);
30   return (true, a / b);
31   }
32   function tryMod(uint256 a, uint256 b) internal pure returns (bool, uint256) {
33   if (b == 0) return (false, 0);
34
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 34

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
33   if (b == 0) return (false, 0);
34   return (true, a % b);
35   }
36   function add(uint256 a, uint256 b) internal pure returns (uint256) {
37   uint256 c = a + b;
38
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 37

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
36   function add(uint256 a, uint256 b) internal pure returns (uint256) {
37   uint256 c = a + b;
38   require(c >= a, "SafeMath: addition overflow");
39   return c;
40   }
41
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 43

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
42   require(b <= a, "SafeMath: subtraction overflow");
43   return a - b;
44   }
45   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
46   if (a == 0) return 0;
47
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 47

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
46   if (a == 0) return 0;
47   uint256 c = a * b;
48   require(c / a == b, "SafeMath: multiplication overflow");
49   return c;
50   }
51
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 48

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
47   uint256 c = a * b;
48   require(c / a == b, "SafeMath: multiplication overflow");
49   return c;
50   }
51   function div(uint256 a, uint256 b) internal pure returns (uint256) {
52
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 53

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
52   require(b > 0, "SafeMath: division by zero");
53   return a / b;
54   }
55   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
56   require(b > 0, "SafeMath: modulo by zero");
57
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 57

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
56   require(b > 0, "SafeMath: modulo by zero");
57   return a % b;
58   }
59   function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
60   require(b <= a, errorMessage);
61
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 61

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
60   require(b <= a, errorMessage);
61   return a - b;
62   }
63   function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
64   require(b > 0, errorMessage);
65
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 65

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
64   require(b > 0, errorMessage);
65   return a / b;
66   }
67   function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
68   require(b > 0, errorMessage);
69
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 69

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
68   require(b > 0, errorMessage);
69   return a % b;
70   }
71   }
72
73
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 346

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
345
346   uint256 public constant MAX_SUPPLY = 50 * 10**6 * 10**18;
347   uint256 private constant MAX_TAX = 5000;
348
349   bool private swapping;
350
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 346

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
345
346   uint256 public constant MAX_SUPPLY = 50 * 10**6 * 10**18;
347   uint256 private constant MAX_TAX = 5000;
348
349   bool private swapping;
350
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 346

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
345
346   uint256 public constant MAX_SUPPLY = 50 * 10**6 * 10**18;
347   uint256 private constant MAX_TAX = 5000;
348
349   bool private swapping;
350
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 346

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
345
346   uint256 public constant MAX_SUPPLY = 50 * 10**6 * 10**18;
347   uint256 private constant MAX_TAX = 5000;
348
349   bool private swapping;
350
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 378

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
377   uint256 public rewardBuyerFee = 25;
378   uint256 public totalBuyFees = liquidityFee + busdDividendFee + marketingFee +
treasuryFee + rewardBuyerFee;
379
380   uint256 public maxSellTransactionAmount = 50000 * 10**18;
381   uint256 public swapTokensAtAmount = 2000 * 10 ** 18;
382
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 378

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
377   uint256 public rewardBuyerFee = 25;
378   uint256 public totalBuyFees = liquidityFee + busdDividendFee + marketingFee +
treasuryFee + rewardBuyerFee;
379
380   uint256 public maxSellTransactionAmount = 50000 * 10**18;
381   uint256 public swapTokensAtAmount = 2000 * 10 ** 18;
382
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 378

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
377   uint256 public rewardBuyerFee = 25;
378   uint256 public totalBuyFees = liquidityFee + busdDividendFee + marketingFee +
treasuryFee + rewardBuyerFee;
379
380   uint256 public maxSellTransactionAmount = 50000 * 10**18;
381   uint256 public swapTokensAtAmount = 2000 * 10 ** 18;
382
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 378

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
377   uint256 public rewardBuyerFee = 25;
378   uint256 public totalBuyFees = liquidityFee + busdDividendFee + marketingFee +
treasuryFee + rewardBuyerFee;
379
380   uint256 public maxSellTransactionAmount = 50000 * 10**18;
381   uint256 public swapTokensAtAmount = 2000 * 10 ** 18;
382
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 380

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
379
380   uint256 public maxSellTransactionAmount = 50000 * 10**18;
381   uint256 public swapTokensAtAmount = 2000 * 10 ** 18;
382
383   mapping (address => bool) private isExcludedFromFees;
384
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 380

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
379
380   uint256 public maxSellTransactionAmount = 50000 * 10**18;
381   uint256 public swapTokensAtAmount = 2000 * 10 ** 18;
382
383   mapping (address => bool) private isExcludedFromFees;
384
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 381

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
380   uint256 public maxSellTransactionAmount = 50000 * 10**18;
381   uint256 public swapTokensAtAmount = 2000 * 10 ** 18;
382
383   mapping (address => bool) private isExcludedFromFees;
384   mapping (address => bool) public automatedMarketMakerPairs;
385
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 381

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
380   uint256 public maxSellTransactionAmount = 50000 * 10**18;
381   uint256 public swapTokensAtAmount = 2000 * 10 ** 18;
382
383   mapping (address => bool) private isExcludedFromFees;
384   mapping (address => bool) public automatedMarketMakerPairs;
385
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 395

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
394   uint256 public nukePercentToBurn = 5000;
395   uint256 public minNukeAmount = 1000 * 10**18;
396   uint256 public totalNuked;
397   bool public autoNuke = true;
398
399
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 395

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
394   uint256 public nukePercentToBurn = 5000;
395   uint256 public minNukeAmount = 1000 * 10**18;
396   uint256 public totalNuked;
397   bool public autoNuke = true;
398
399
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 454

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
453   function getPeriod() public view returns (uint256) {
454   uint256 secondsSinceLaunch = block.timestamp - launchTime;
455   return 1 + (secondsSinceLaunch / biggestBuyerPeriod);
456   }
457
458
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 455

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
454   uint256 secondsSinceLaunch = block.timestamp - launchTime;
455   return 1 + (secondsSinceLaunch / biggestBuyerPeriod);
456   }
457
458   function manualNukeLpTokens(address _lpAddress, uint256 _percent) external
onlyOwner {
459
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 455

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
454   uint256 secondsSinceLaunch = block.timestamp - launchTime;
455   return 1 + (secondsSinceLaunch / biggestBuyerPeriod);
456   }
457
458   function manualNukeLpTokens(address _lpAddress, uint256 _percent) external
onlyOwner {
459
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 461

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
460   require(_percent <= 1000, 'Cannot burn more than 10% dex balance');
461   _nukeFromLp(_lpAddress, (balanceOf(_lpAddress) * _percent) / 10000);
462   }
463   function nukeLpTokenFromBuildup() external authorized {
464   _nukeLpTokenFromBuildup();
465
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 461

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
460   require(_percent <= 1000, 'Cannot burn more than 10% dex balance');
461   _nukeFromLp(_lpAddress, (balanceOf(_lpAddress) * _percent) / 10000);
462   }
463   function nukeLpTokenFromBuildup() external authorized {
464   _nukeLpTokenFromBuildup();
465
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 468

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
467   if(lpNukeEnabled){
468   for(uint i = 0; i < _markerPairs.length; i++){
469
470   uint256 nukeAmount = lpNukeBuildup[_markerPairs[i]];
471
472
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 489

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
488   lpNukeBuildup[lpAddress] = 0;
489   totalNuked = totalNuked + amount;
490   uint256 nukeToBurn = amount.mul(nukePercentToBurn).div(10000);
491   if (nukeToBurn>0) {
492   super._transfer(lpAddress, deadAddress, nukeToBurn);
493
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 495

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
494   if (amount > nukeToBurn) {
495   super._transfer(lpAddress, address(nukeTreasury), amount - nukeToBurn);
496   nukeTreasury.updateRewards();
497   }
498
499
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 548

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
547
548   _approve(address(this), address(dexRouter), 2**256 - 1);
549
550   //approve for owner, not quite necessary
551   approve(address(dexRouter), 2**256 - 1);
552
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 548

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
547
548   _approve(address(this), address(dexRouter), 2**256 - 1);
549
550   //approve for owner, not quite necessary
551   approve(address(dexRouter), 2**256 - 1);
552
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 551

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
550   //approve for owner, not quite necessary
551   approve(address(dexRouter), 2**256 - 1);
552
553   //liquidity making outside of contract, so this is not needed any more
554   //IERC20(busdToken).approve(address(dexRouter), 2**256 - 1);
555
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 551

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
550   //approve for owner, not quite necessary
551   approve(address(dexRouter), 2**256 - 1);
552
553   //liquidity making outside of contract, so this is not needed any more
554   //IERC20(busdToken).approve(address(dexRouter), 2**256 - 1);
555
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 565

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
564   require( _dexPair != dexPair, "Cannot remove dexPair");
565   for (uint256 i = 0; i < _markerPairs.length; i++) {
566   if (_markerPairs[i] == _dexPair) {
567   _markerPairs[i] = _markerPairs[_markerPairs.length - 1];
568   _markerPairs.pop();
569
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 567

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
566   if (_markerPairs[i] == _dexPair) {
567   _markerPairs[i] = _markerPairs[_markerPairs.length - 1];
568   _markerPairs.pop();
569   break;
570   }
571
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 578

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
577   function setMaxSell(uint256 _amount) external onlyOwner {
578   require(_amount >= 10**18,"Too small");
579   maxSellTransactionAmount = _amount;
580   }
581
582
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 624

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
623   ) external onlyOwner {
624   uint256 _totalBuyFees = _liquidityFee + _busdDividendFee + _marketingFee +
_treasuryFee + _rewardBuyerFee;
625
626   require(_totalBuyFees <= MAX_TAX, "Buy fee too high");
627   require(_totalSellFees <= MAX_TAX, "Sell fee too high");
628
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 624

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
623   ) external onlyOwner {
624   uint256 _totalBuyFees = _liquidityFee + _busdDividendFee + _marketingFee +
_treasuryFee + _rewardBuyerFee;
625
626   require(_totalBuyFees <= MAX_TAX, "Buy fee too high");
627   require(_totalSellFees <= MAX_TAX, "Sell fee too high");
628
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 624

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
623   ) external onlyOwner {
624   uint256 _totalBuyFees = _liquidityFee + _busdDividendFee + _marketingFee +
_treasuryFee + _rewardBuyerFee;
625
626   require(_totalBuyFees <= MAX_TAX, "Buy fee too high");
627   require(_totalSellFees <= MAX_TAX, "Sell fee too high");
628
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 624

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
623   ) external onlyOwner {
624   uint256 _totalBuyFees = _liquidityFee + _busdDividendFee + _marketingFee +
_treasuryFee + _rewardBuyerFee;
625
626   require(_totalBuyFees <= MAX_TAX, "Buy fee too high");
627   require(_totalSellFees <= MAX_TAX, "Sell fee too high");
628
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 656

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
655   if(!lpNukeEnabled){
656   for(uint i = 0; i < _markerPairs.length; i++){
657   lpNukeBuildup[_markerPairs[i]] = 0;
658   }
659   }
660
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 790

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
789   if (lpNukeEnabled && isSelling && from != address(this) && !excludedAccount) {
790   lpNukeBuildup[to] += amount.mul(nukePercentPerSell).div(10000);
791   }
792   if (autoNuke && !swapping && lpNukeEnabled && !isSelling && !isBuying){
793   _nukeLpTokenFromBuildup();
794
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 859

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
858   function _checkAndPayBiggestBuyer(uint256 _currentPeriod) private {
859   uint256 _prevPeriod = _currentPeriod - 1;
860   if (
861   _currentPeriod > 1 &&
862   biggestBuyerAmount[_prevPeriod] > 0 &&
863
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 868

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
867   _transferBNBToWallet(payable(biggestBuyer[_prevPeriod]), _rewardAmount);
868   totalBiggestBuyerPaid = totalBiggestBuyerPaid + _rewardAmount;
869   biggestBuyerPaid[_prevPeriod] = _rewardAmount;
870
871   emit PayBiggestBuyer(biggestBuyer[_prevPeriod], _prevPeriod, _rewardAmount);
872
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 912

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
911   if (circuitBreakerFlag == 2) {
912   if (circuitBreakerTime + breakerPeriod < block.timestamp) {
913   _deactivateCircuitBreaker();
914   }
915   }
916
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 926

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
925   if (timeDiffGlobal < breakerPeriod) {
926   _taxBreakerCheck = _taxBreakerCheck + priceChange;
927   } else {
928   _taxBreakerCheck = priceChange;
929   _timeBreakerCheck = block.timestamp;
930
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 945

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
944   uint deno = r1.add(x).mul(r1_.add(x_));
945   uint priceChange = nume / deno;
946   priceChange = (uint(10000).sub(priceChange)).div(2);
947
948   return priceChange;
949
```

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

LINE 567

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
566   if (_markerPairs[i] == _dexPair) {
567   _markerPairs[i] = _markerPairs[_markerPairs.length - 1];
568   _markerPairs.pop();
569   break;
570   }
571
```

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

LINE 859

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
858   function _checkAndPayBiggestBuyer(uint256 _currentPeriod) private {
859   uint256 _prevPeriod = _currentPeriod - 1;
860   if (
861   _currentPeriod > 1 &&
862   biggestBuyerAmount[_prevPeriod] > 0 &&
863
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

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

- LiberaToken.sol

Locations

```
6
7   pragma solidity ^0.8.4;
8
9   library SafeMath {
10   function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
11
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 85

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
84   abstract contract Auth is Context{
85   address owner;
86   mapping (address => bool) private authorizations;
87
88   constructor(address _owner) {
89
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 470

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
469
470   uint256 nukeAmount = lpNukeBuildup[_markerPairs[i]];
471
472   if(nukeAmount > minNukeAmount){
473   uint256 maxBuildUp = balanceOf(_markerPairs[i]).mul(1000).div(10000);
474
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 473

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
472   if(nukeAmount > minNukeAmount){
473   uint256 maxBuildUp = balanceOf(_markerPairs[i]).mul(1000).div(10000);
474
475   if(nukeAmount > maxBuildUp){
476   nukeAmount = maxBuildUp;
477
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 479

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
478
479   _nukeFromLp(_markerPairs[i], nukeAmount);
480   }
481   }
482   }
483
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 566

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
565   for (uint256 i = 0; i < _markerPairs.length; i++) {
566   if (_markerPairs[i] == _dexPair) {
567   _markerPairs[i] = _markerPairs[_markerPairs.length - 1];
568   _markerPairs.pop();
569   break;
570
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 567

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
566   if (_markerPairs[i] == _dexPair) {
567   _markerPairs[i] = _markerPairs[_markerPairs.length - 1];
568   _markerPairs.pop();
569   break;
570   }
571
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 567

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
566   if (_markerPairs[i] == _dexPair) {
567   _markerPairs[i] = _markerPairs[_markerPairs.length - 1];
568   _markerPairs.pop();
569   break;
570   }
571
```

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

- LiberaToken.sol

Locations

```
656   for(uint i = 0; i < _markerPairs.length; i++){
657   lpNukeBuildup[_markerPairs[i]] = 0;
658   }
659   }
660   }
661
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 813

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
812   path = new address[](3);
813   path[0] = address(this);
814   path[1] = dexToken;
815   path[2] = dexRouter.WETH();
816   } else {
817
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 814

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
813   path[0] = address(this);
814   path[1] = dexToken;
815   path[2] = dexRouter.WETH();
816   } else {
817   path = new address[](2);
818
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 815

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
814   path[1] = dexToken;
815   path[2] = dexRouter.WETH();
816   } else {
817   path = new address[](2);
818   path[0] = address(this);
819
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 818

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
817   path = new address[](2);
818   path[0] = address(this);
819   path[1] = dexRouter.WETH();
820   }
821
822
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 819

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
818   path[0] = address(this);
819   path[1] = dexRouter.WETH();
820   }
821
822   dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
823
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 836

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
835   path = new address[](2);
836   path[0] = address(this);
837   path[1] = busdToken;
838   } else {
839   path = new address[](3);
840
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 837

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
836   path[0] = address(this);
837   path[1] = busdToken;
838   } else {
839   path = new address[](3);
840   path[0] = address(this);
841
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 840

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
839   path = new address[](3);
840   path[0] = address(this);
841   path[1] = dexToken;
842   path[2] = busdToken;
843   }
844
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 841

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
840   path[0] = address(this);
841   path[1] = dexToken;
842   path[2] = busdToken;
843   }
844
845
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 842

low SEVERITY

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

Source File

- LiberaToken.sol

Locations

```
841   path[1] = dexToken;
842   path[2] = busdToken;
843   }
844
845   dexRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(
846
```

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.