Aspirant

Smart Contract Audit Report

SYSFIXED

07 May 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Aspirant	ASPIRANT	BSC

Addresses

Contract address	0xCCD9748c5204927AC7fC9ECc47bA4Df6Aaa29399
Contract deployer address	0xe716d9Fd11276B03056892F29ad72c75693AeE64

Project Website

https://www.aspirant.app/

Codebase

https://bscscan.com/address/0xCCD9748c5204927AC7fC9ECc47bA4Df6Aaa29399#code

SUMMARY

Aspirant is a secure marketplace for contractors and customers that enables you to find trusted professionals for the services you require, such as car services, home services, construction services, and much more. Whether you are a customer looking for a secure home services online platform to hire local experts or a service provider willing to offer their services to local customers, we've got you covered.

Contract Summary

Documentation Quality

This project has a standard of documentation.

- Technical description provided.

Code Quality

The quality of the code in this project is up to standard.

- The official Solidity style guide is followed.

Test Scope

Project test coverage is 100% (Via Codebase).

Audit Findings Summary

Issues Found

- SWC-101 | Arithmetic operation issues discovered on lines 104, 136, 159, 160, 195, 231, 758, 758, 759, 783, 784, 915, 917, 940, 995, 1030, 1036, 1042, .
- SWC-101 | Compiler-rewritable " - 1" discovered on line 917
- SWC-108 | State variable visibility is not set on line 780. It is best practice to set the visibility of state variables explicitly to public or private.
- SWC-110 | Out of bounds array access issues discovered on lines 916, 917, 996, 997, 998, 1152, and 1153.

CONCLUSION

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

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

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

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	PASS
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	Fri May 05 2022 23:17:20 GMT+0000 (Coordinated Universal Time)
Finished	Sat May 06 2022 02:02:50 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	aspirant.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 104

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
103   function add(uint256 a, uint256 b) internal pure returns (uint256) {
104   uint256 c = a + b;
105   require(c >= a, "SafeMath: addition overflow");
106
107   return c;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 136

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
135   require(b <= a, errorMessage);
136   uint256 c = a - b;
137
138   return c;
139   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 159

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
158
159   uint256 c = a * b;
160   require(c / a == b, "SafeMath: multiplication overflow");
161
162   return c;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 160

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
159   uint256 c = a * b;
160   require(c / a == b, "SafeMath: multiplication overflow");
161
162   return c;
163   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 195

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
194   require(b > 0, errorMessage);
195   uint256 c = a / b;
196   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
197
198   return c;
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 231

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
230   require(b != 0, errorMessage);
231   return a % b;
232   }
233   }
234
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 758

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
757   uint256 private constant MAX = ~uint256(0);
758   uint256 private _tTotal = 986_000_000_000 * 10 ** 18;
759   uint256 private _rTotal = (MAX - (MAX % _tTotal));
760   uint256 private _tFeeTotal;
761
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 759

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
758   uint256 private _tTotal = 986_000_000_000 * 10 ** 18;
759   uint256 private _rTotal = (MAX - (MAX % _tTotal));
760   uint256 private _tFeeTotal;
761
762   string private _name = "Aspirant";
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 783

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
782
783   uint256 public _maxTxAmount = 4_930_000_000 * 10 ** 18;
784   uint256 private numTokensSellToAddToLiquidity = 493_000_000 * 10 ** 18;
785
786   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 784

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
783   uint256 public _maxTxAmount = 4_930_000_000 * 10 ** 18;
784   uint256 private numTokensSellToAddToLiquidity = 493_000_000 * 10 ** 18;
785
786   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
787   event SwapAndLiquifyEnabledUpdated(bool enabled);
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 915

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
914   require(_isExcluded[account], "Account is already excluded");
915   for (uint256 i = 0; i < _excluded.length; i++) {
916   if (_excluded[i] == account) {
917   _excluded[i] = _excluded[_excluded.length - 1];
918   _tOwned[account] = 0;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 917

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
916   if (_excluded[i] == account) {
917   _excluded[i] = _excluded[_excluded.length - 1];
918   _tOwned[account] = 0;
919   _isExcluded[account] = false;
920   _excluded.pop();
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 940

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
939   _maxTxAmount = _tTotal.mul(maxTxPercent).div(
940   10**2
941   );
942   }
943
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 995

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
994   uint256 tSupply = _tTotal;
995   for (uint256 i = 0; i < _excluded.length; i++) {
996   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
997   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
998   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1030

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
1029   return _amount.mul(_taxFee).div(
1030   10**2
1031   );
1032   }
1033
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1036

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
1035   return _amount.mul(_liquidityFee).div(
1036   10**2
1037   );
1038   }
1039
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1042

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
1041   return _amount.mul(_marketingFee).div(
1042   10**2
1043   );
1044   }
1045
```

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

LINE 917

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
916   if (_excluded[i] == account) {
917   _excluded[i] = _excluded[_excluded.length - 1];
918   _tOwned[account] = 0;
919   _isExcluded[account] = false;
920   _excluded.pop();
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 780

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

- aspirant.sol

Locations

```
779
780   bool inSwapAndLiquify;
781   bool public swapAndLiquifyEnabled = false;
782
783   uint256 public _maxTxAmount = 4_930_000_000 * 10 ** 18;
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 916

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
915   for (uint256 i = 0; i < _excluded.length; i++) {
916   if (_excluded[i] == account) {
917   _excluded[i] = _excluded[_excluded.length - 1];
918   _tOwned[account] = 0;
919   _isExcluded[account] = false;
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 917

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
916   if (_excluded[i] == account) {
917   _excluded[i] = _excluded[_excluded.length - 1];
918   _tOwned[account] = 0;
919   _isExcluded[account] = false;
920   _excluded.pop();
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 996

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
995   for (uint256 i = 0; i < _excluded.length; i++) {
996   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
997   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
998   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
999   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 997

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
996   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
997   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
998   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
999   }
1000   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 998

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
997   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
998   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
999   }
1000   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1001   return (rSupply, tSupply);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1152

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
1151   address[] memory path = new address[](2);
1152   path[0] = address(this);
1153   path[1] = uniswapV2Router.WETH();
1154
1155   _approve(address(this), address(uniswapV2Router), tokenAmount);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1153

low SEVERITY

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

Source File

- aspirant.sol

Locations

```
1152   path[0] = address(this);
1153   path[1] = uniswapV2Router.WETH();
1154
1155   _approve(address(this), address(uniswapV2Router), tokenAmount);
1156
```

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.