Quadrium

Smart Contract Audit Report

SYSFIXED

16 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Quadrium	QUAD	Binance Smart Chain

Addresses

Contract address	0x3913c87296CdE0222634a7732Ff5Cda77cCF386A
Contract deployer address	0x96b516815919977C75B095953c62c0B4F40fBe4F

Project Website

https://quadriumcompany.com/

Codebase

https://bscscan.com/address/0x3913c87296CdE0222634a7732Ff5Cda77cCF386A#code

SUMMARY

Quadrium Foundation is a Singapore-India-Uzbekistan joint company operating in all areas of Blockchain and Web 3.0. The company already has an official license to operate in the market of Uzbekistan!

Contract Summary

Documentation Quality

Quadrium 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 Quadrium 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 line 431.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 31, 43, 54, 55, 66, 78, 411, 412, 434, 564, 566, 625, 644, 650 and 566.
- SWC-110 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 565, 566, 626, 627, 628, 730 and 731.

CONCLUSION

We have audited the Quadrium project released on January 2023 to discover issues and identify potential security vulnerabilities in Quadrium 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 code on Quadrium smart contract 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 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.

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
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegate calls 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	Sunday Jan 15 2023 06:42:32 GMT+0000 (Coordinated Universal Time)
Finished	Monday Jan 16 2023 17:00:44 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Quadrium.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	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 31

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
30   function add(uint256 a, uint256 b) internal pure returns (uint256) {
31   uint256 c = a + b;
32   require(c >= a, "SafeMath: addition overflow");
33
34   return c;
35
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 43

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
42   require(b <= a, errorMessage);
43   uint256 c = a - b;
44
45   return c;
46   }
47
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 54

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
53
54   uint256 c = a * b;
55   require(c / a == b, "SafeMath: multiplication overflow");
56
57   return c;
58
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 55

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
54   uint256 c = a * b;
55   require(c / a == b, "SafeMath: multiplication overflow");
56
57   return c;
58   }
59
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 66

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
65   require(b > 0, errorMessage);
66   uint256 c = a / b;
67   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
68
69   return c;
70
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 78

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
77   require(b != 0, errorMessage);
78   return a % b;
79   }
80   }
81
82
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 411

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
410   uint256 private constant MAX = ~uint256(0);
411   uint256 private _tTotal = 10000000000 * 10**18;
412   uint256 private _rTotal = (MAX - (MAX % _tTotal));
413   uint256 private _tFeeTotal;
414
415
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 412

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
411   uint256 private _tTotal = 10000000000 * 10**18;
412   uint256 private _rTotal = (MAX - (MAX % _tTotal));
413   uint256 private _tFeeTotal;
414
415   string private _name = "Quadrium";
416
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 434

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
433
434   uint256 private numTokensSellToAddToLiquidity = 8000 * 10**18;
435
436   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
437   event SwapAndLiquifyEnabledUpdated(bool enabled);
438
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 564

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
563   require(_isExcluded[account], "Account is already excluded");
564   for (uint256 i = 0; i < _excluded.length; i++) {
565   if (_excluded[i] == account) {
566   _excluded[i] = _excluded[_excluded.length - 1];
567   _tOwned[account] = 0;
568
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 566

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
565   if (_excluded[i] == account) {
566   _excluded[i] = _excluded[_excluded.length - 1];
567   _tOwned[account] = 0;
568   _isExcluded[account] = false;
569   _excluded.pop();
570
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 625

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
624   uint256 tSupply = _tTotal;
625   for (uint256 i = 0; i < _excluded.length; i++) {
626   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
627   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
628   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
629
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 644

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
643   return _amount.mul(_RewardFee).div(
644   10**2
645   );
646   }
647
648
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 650

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
649   return _amount.mul(_liquidityFee).div(
650   10**2
651   );
652   }
653
654
```

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

LINE 566

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
565   if (_excluded[i] == account) {
566   _excluded[i] = _excluded[_excluded.length - 1];
567   _tOwned[account] = 0;
568   _isExcluded[account] = false;
569   _excluded.pop();
570
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 431

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

- Quadrium.sol

Locations

```
430
431   bool inSwapAndLiquify;
432   bool public swapAndLiquifyEnabled = false;
433
434   uint256 private numTokensSellToAddToLiquidity = 8000 * 10**18;
435
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 565

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
564   for (uint256 i = 0; i < _excluded.length; i++) {
565   if (_excluded[i] == account) {
566   _excluded[i] = _excluded[_excluded.length - 1];
567   _tOwned[account] = 0;
568   _isExcluded[account] = false;
569
```

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

- Quadrium.sol

Locations

```
565   if (_excluded[i] == account) {
566   _excluded[i] = _excluded[_excluded.length - 1];
567   _tOwned[account] = 0;
568   _isExcluded[account] = false;
569   _excluded.pop();
570
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 626

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
625   for (uint256 i = 0; i < _excluded.length; i++) {
626   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
627   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
628   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
629   }
630
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 627

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
626   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
627   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
628   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
629   }
630   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
631
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 628

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
627   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
628   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
629   }
630   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
631   return (rSupply, tSupply);
632
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 730

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
729   address[] memory path = new address[](2);
730   path[0] = address(this);
731   path[1] = uniswapV2Router.WETH();
732
733   _approve(address(this), address(uniswapV2Router), tokenAmount);
734
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 731

low SEVERITY

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

Source File

- Quadrium.sol

Locations

```
730   path[0] = address(this);
731   path[1] = uniswapV2Router.WETH();
732
733   _approve(address(this), address(uniswapV2Router), tokenAmount);
734
735
```

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.